Towards a Practical Cluster Analysis over Encrypted Data

Jung Hee Cheon, Duhyeong Kim(✉), and Jai Hyun Park

Department of Mathematical Sciences,
Seoul National University, Seoul, South Korea
{jhcheon,doodoo1204,jhyunp}@snu.ac.kr

Abstract. Cluster analysis is one of the most significant unsupervised machine learning methods, and it is being utilized in various fields associated with privacy issues including bioinformatics, finance and image processing. In this paper, we propose a practical solution for privacy-preserving cluster analysis based on homomorphic encryption (HE). Our work is the first HE solution for the mean-shift clustering algorithm. To reduce the super-linear complexity of the original mean-shift algorithm, we adopt a novel random sampling method called dust sampling approach, which perfectly suits with HE and achieves the linear complexity. We also substitute non-polynomial kernels by a new polynomial kernel so that it can be efficiently computed in HE.

The HE implementation of our modified mean-shift clustering algorithm based on the approximate HE scheme HEAAN shows prominent performance in terms of speed and accuracy. It takes approx. 30 min with 99% accuracy over several public datasets with hundreds of data, and even for the dataset with 262,144 data, it takes 82 min only when SIMD operations in HEAAN is applied. Our results outperform the previously best known result (SAC 2018) by over 400 times.

Keywords: Clustering · Mean-shift · Homomorphic encryption · Privacy

1 Introduction

For a decade, machine learning has garnered much attention globally in various fields due to its strong ability to resolve various real world problems. Since many fields of frequently-used data such as financial and biomedical data including personal or sensitive information, privacy-related issues are inevitable in the use of machine learning in such fields. There have been several non-cryptographic approaches for privacy-preserving machine learning including anonymization, perturbation, randomization and condensation [34,44]; however, these methods commonly accompany a potential loss of information which might degrade the utility of data.

On the other hand, Homomorphic Encryption (HE), which allows *computations over encrypted data* without any decryption process, is theoretically one

K. G. Paterson and D. Stebila (Eds.): SAC 2019, LNCS 11959, pp. 227–249, 2020.
https://doi.org/10.1007/978-3-030-38471-5_10

of the most ideal cryptographic primitives for privacy protection without the potential leakage of any information related to relevant data. There have been a number of studies on privacy-preserving machine learning based on HE, particularly supervised machine learning tasks such as classification and regression; including logistic regression [5,9,15,19,27,30,31,45] and (the prediction phase of) deep neural networks [6,25].

cluster analysis is one of the most significant unsupervised machine learning tasks, which aims to split a set of given data into several subgroups, called clusters, in which such data in the same cluster are "similar" to each other. As well as classification and regression, clustering is also widely used in various fields that engage the use of private information, including bioinformatics, image segmentation, finance, customer behavior analysis and forensics [20,22,36].

Contrary to classification and regression, there are only a few works [4,29] on privacy-preserving clustering based on HE, and even only one of these works provides a full HE solution, i.e., the whole procedure is done by HE operations without any decryption process or a trusted third-party setting. The main reason for the slow progress of the research on HE-based clustering is that there are many HE-unfriendly operations such as division and comparison. Recently, Cheon et al. [14] proposed efficient HE algorithms for division and comparison of numbers which are encrypted word-wise, and this work surely has its significance as it has initiated and called for active research on HE-based clustering.

1.1 This Work

In this paper, we propose a practical solution of privacy-preserving cluster analysis based on HE. Our solution is the *first* HE algorithm for *mean-shift clustering*, which is one of the representative algorithms for cluster analysis (Fig. 1). For given n-dimensional points $P_1, P_2, \ldots, P_p$ and a function called *kernel* $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$, the mean-shift clustering utilizes the gradient descent algorithm which finds local maxima (called *modes*) of the kernel density estimator $F(\boldsymbol{x}) = \frac{1}{p} \cdot \sum_{i=1}^{p} K(\boldsymbol{x}, P_i)$, in which $K(\boldsymbol{x}, P_i)$ outputs a value close to 0 when $\boldsymbol{x}$ and P_i are far from each other.

Core Ideas. The major challenges for the original mean-shift algorithm to be applied on HE are (1) super-linear computational complexity $O(p^2)$ for each mean-shift process and (2) non-polynomial operations in kernel which are hard to be efficiently computed in HE. In order to overcome these challenges, we suggest several novel techniques to modify the original mean-shift algorithm into an *HE-friendly* form:

- Rather than mean-shifting every given point, we randomly sample several points called *dusts* and the mean-shift process will only be conducted for the *dusts*. As a result, the computational cost to seek the modes is reduced from $O(p^2)$ to $O(d \cdot p)$ where d is the number of dusts, which is much smaller than p.
- After the mode-seeking phase, one should match given points to the closest mode, which we call *point-labeling*. We suggest a carefully devised algorithm

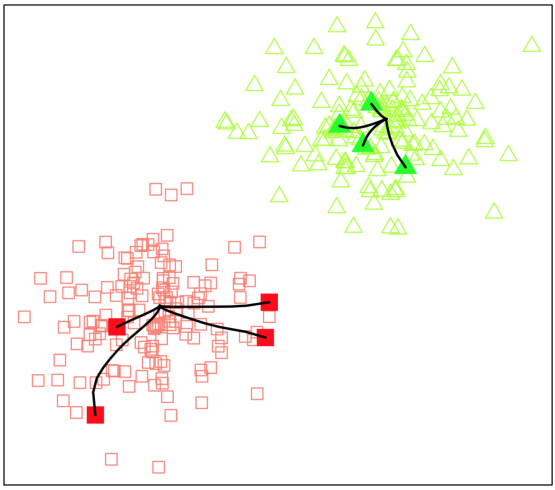

Fig. 1. Illustration of the mean-shift algorithm

for labeling points with the modes, which only consists of polynomial operations so that it can be implemented by HE efficiently.

- We propose a new *HE-friendly kernel* $K(\boldsymbol{x}, \boldsymbol{y}) = (1 - ||\boldsymbol{x} - \boldsymbol{y}||^2)^{2^r+1}$. The most commonly used kernel functions in clustering are Gaussian kernel and Epanechnikov kernel. However, the derivatives of those functions, which we should compute for each mean-shift process, are either exponential or discontinuous. Our new kernel is a simple polynomial which only requires $\log(degree)$ complexity to compute its derivative, and the cluster analysis based on this HE-friendly kernel is very accurate in practice.

Practical Performance: Fast and Accurate. To the best of our knowledge, the work in [29] has been a unique full HE solution to privacy-preserving clustering so far. While their implementation takes as much as 619 h for 400 2-dimensional data, our algorithm takes only approx. 1.4 h for the same dataset, which is over 400 times faster than the previous result. Using a multi-threading option with 8 threads, its running time is even reduced to *half an hour*. The fast and accurate performance of our algorithm implies that the research on HE-based privacy-preserving clustering is approaching to a stage of practical application.

Why Mean-shift Clustering? K-means clustering is another representative algorithm for clustering, and many of the previous works on privacy-preserving clustering used the K-means clustering algorithm. However, there are some critical drawbacks in K-means clustering in the perspective of HE applications. Firstly, K-means clustering requires a user to pre-determine the exact number of clusters. However, there is no way to determine the number of clusters when the encrypted data are given only. Therefore, a data owner should additionally provide the number of clusters, but determining the exact number of clusters

from a given dataset also requires a costly process even in an unencrypted state [41]. Secondly, K-means clustering often does not work when the shape of clusters is non-convex, but the shape of clusters is also non-predictable information from encrypted data.

1.2 Related Works

In the case of HE-based privacy-preserving clustering, to the best of our knowledge, there has been proposed only a single solution [29] which does not requires any decryption process during the analysis. They transform the K-means clustering algorithm into an HE algorithm based on the HE scheme TFHE [16,17], which encrypts data bit-wisely. One of their core ideas is to modify the original K-means clustering algorithm by substituting a homomorphic division of a ciphertext, which is very expensive, with a simple constant division. As a result, to run their modified algorithm with TFHE over 400 2-dimensional data, it takes approx. 619 h (≈.26 days) on a virtual machine with an Intel i7-3770 processor with 3.4 GHz without parallelization options. Before this work, there has been an attempt [4] to perform K-means clustering based on HE with trusted third party; however, the HE scheme they used [32] was proved to be insecure [46].

Contrary to HE, there have been a number of works [7,21,28,33,38–40,43] on privacy-preserving clustering based on another cryptographic tool called Multiparty Computation (MPC), which is a protocol between several parties to jointly compute a function without revealing any information of their inputs. For more details on MPC-based privacy-preserving clustering algorithms, we refer the readers to a survey paper written by Meskine and Nait-Bahloul [35]. MPC is normally known to be much faster than HE; however, MPC requires *online* computation of data owners and it yields significantly large bandwidth. On the other hand, HE computation can be totally done *offline* after encrypted data are sent to a computing service provider. Since data owners do not need to participate in the computation phase, HE-based solutions can be regarded to be much more convenient and economic to data owners than MPC.

2 Backgrounds

2.1 Notations

We call each given datum of the clustering problem *a point*. Let n be the dimension of each point, and $P = \{P_1, P_2, \dots, P_p\}$ be the set of given points where p is the number of elements in P. We denote the set of dusts, which will be defined in Sect. 3, by $D = \{D_1, D_2, \dots, D_d\}$ where d is the number of dusts. There are several auxiliary parameters for our new algorithms in Sects. 2 and 3: ζ, t, Γ and T denote the number of iterations for `Inv`, `MinIdx`, `Kernel` and `Mode-seeking`, respectively. $\mathbb{R}$ denotes the real number field, and $\mathbb{R}_{\geq 0}$ is a subset of $\mathbb{R}$ which consists of non-negative real numbers. The set $\mathbb{B}_n(1/2)$ denotes the n-dimensional ball of the radius $1/2$ with center 0. For an n-dimensional vector $\boldsymbol{x} \in \mathbb{R}^n$, the

L_2-norm of $\boldsymbol{x}$ is denoted by $||\boldsymbol{x}||$. For a finite set X, $x \leftarrow U(X)$ means that x is sampled uniformly at random from X, and $|X|$ denotes the number of elements in X. For (row) vectors $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^m$, the concatenation of the two vectors is denoted by $(\boldsymbol{x}||\boldsymbol{y}) \in \mathbb{R}^{n+m}$. For a positive integer q, $[\cdot]_q$ denotes a residue modulo q in $[-q/2, q/2)$.

2.2 Approximate Homomorphic Encryption HEAAN

For privacy-preserving clustering, we apply an HE scheme called HEAAN proposed by Cheon et al. [12,13], which supports *approximate computation* of real numbers in encrypted state. Efficiency of HEAAN in the real world has been proved by showing its applications in various fields including machine learning [15,30,31] and cyber-physical systems [11]. After the solution [30] based on HEAAN won the first place in privacy-preserving genome data analysis competition called IDash in 2017, all the solutions for the next-year competition which aimed to develop a privacy-preserving solution for Genome-wide Association Study (GWAS) computation were constructed based on HEAAN.

In detail, let ct be a HEAAN ciphertext of a plaintext vector $\boldsymbol{m} \in \mathbb{C}^{N/2}$. Then, the decryption process with a secret key sk is done as

$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}) = \boldsymbol{m} + e \approx \boldsymbol{m}$$

where e is a small error attached to the plaintext vector $\mathtt{m}$. For formal definitions, let L be a level parameter and $q_\ell := 2^\ell$ for $1 \leq \ell \leq L$. Let $R := \mathbb{Z}[X]/(X^N + 1)$ for a power-of-two N and R_q be a modulo-q quotient ring of R, i.e., $R_q := R/qR$. The distribution $\chi_{\mathtt{key}} := \mathsf{HW}(\mathsf{h})$ over R_q outputs a polynomial of $\{-1, 0, 1\}$-coefficients having h number of non-zero coefficients, and $\chi_{\mathtt{enc}}$ and $\chi_{\mathtt{err}}$ denote the discrete Gaussian distribution with some prefixed standard deviation. Finally, $[\cdot]_q$ denotes a component-wise modulo q operation on each element of R_q. Note that whether those parameters N, L and h are satisfying a certain security level can be determined by Albrecht's security estimator [2,3].

A plaintext vector $\boldsymbol{m} \in \mathbb{C}^{n/2}$ is firstly encoded as a polynomial in R by applying a (field) isomorphism τ from $\mathbb{R}[X]/(X^N + 1)$ to $\mathbb{C}^{N/2}$ called canonical embedding. A naive approach is to transform the plaintext vector as $\tau^{-1}(\boldsymbol{m}) \in \mathbb{R}[X]/(X^N + 1)$; however, the naive rounding-off can derive quite a large relative error on the plaintext. In order to control the error, we round the plaintext off after scaling up by p bits for some integer p, i.e., $\lfloor 2^p \cdot \tau^{-1}(\boldsymbol{m}) \rceil$, so that the relative error can be reduced. The full scheme description of HEAAN is as following:

- $\mathtt{KeyGen}$.
  - Sample $s \leftarrow \chi_{\mathtt{key}}$. Set the secret key as $\mathsf{sk} \leftarrow (1, s)$.
  - Sample $a \leftarrow U(R_{q_L})$ and $e \leftarrow \chi_{\mathtt{err}}$. Set the public key as $\mathsf{pk} \leftarrow (b, a) \in R_{q_L}^2$ where $b \leftarrow [-a \cdot s + e]_{q_L}$.
  - Sample $a' \leftarrow U(R_{q_L^2})$ and $e' \leftarrow \chi_{\mathtt{err}}$. Set the evaluation key as $\mathsf{evk} \leftarrow (b', a') \in R_{q_L^2}^2$ where $b' \leftarrow [-a's + e' + q_L \cdot s^2]_{q_L^2}$.

- $\underline{\mathtt{Enc}_{\mathtt{pk}}(\boldsymbol{m})}$.
  - For a plaintext $\boldsymbol{m} = (m_0, ..., m_{N/2-1})$ in $\mathbb{C}^{N/2}$ and a scaling factor $p > 0$, compute a polynomial $\mathtt{m} \leftarrow \lfloor 2^p \cdot \tau^{-1}(\boldsymbol{m}) \rceil \in R$
  - Sample $v \leftarrow \chi_{\mathtt{enc}}$ and $e_0, e_1 \leftarrow \chi_{\mathtt{err}}$. Output $\mathtt{ct} = [v \cdot \mathtt{pk} + (\mathtt{m} + e_0, e_1)]_{q_L}$.
- $\underline{\mathtt{Dec}_{\mathtt{sk}}(\mathtt{ct})}$.
  - For a ciphertext $\mathtt{ct} = (c_0, c_1) \in R^2_{q_\ell}$, compute $\mathtt{m}' = [c_0 + c_1 \cdot s]_{q_\ell}$.
  - Output a plaintext vector $\boldsymbol{m}' = 2^{-p} \cdot \tau(\mathtt{m}') \in \mathbb{C}^{N/2}$.
- $\underline{\mathtt{Add}(\mathtt{ct}, \mathtt{ct}')}$. For $\mathtt{ct}, \mathtt{ct}' \in R^2_{q_\ell}$, output $\mathtt{ct}_{\mathtt{add}} \leftarrow [\mathtt{ct} + \mathtt{ct}']_{q_\ell}$.
- $\underline{\mathtt{Sub}(\mathtt{ct}, \mathtt{ct}')}$. For $\mathtt{ct}, \mathtt{ct}' \in R^2_{q_\ell}$, output $\mathtt{ct}_{\mathtt{sub}} \leftarrow [\mathtt{ct} - \mathtt{ct}']_{q_\ell}$.
- $\underline{\mathtt{Mult}_{\mathtt{evk}}(\mathtt{ct}, \mathtt{ct}')}$. For $\mathtt{ct} = (c_0, c_1), \mathtt{ct}' = (c'_0, c'_1) \in \mathcal{R}^2_{q_\ell}$, let $(d_0, d_1, d_2) = (c_0 c'_0, c_0 c'_1 + c_1 c'_0, c_1 c'_1)$. Compute $\mathtt{ct}'_{\mathtt{mult}} \leftarrow [(d_0, d_1) + \lfloor q_L^{-1} \cdot d_2 \cdot \mathtt{evk} \rceil]_{q_\ell}$, and output $\mathtt{ct}_{\mathtt{mult}} \leftarrow [\lfloor (1/p) \cdot \mathtt{ct}'_{\mathtt{mult}} \rceil]_{q_{\ell-1}}$.

We omitted the parameters $(N, L, \mathtt{h}, p)$ as an input of the above algorithms for convenience. Let $\mathtt{ct}_1$ and $\mathtt{ct}_2$ be ciphertexts of plaintext vectors $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$. Then, the homomorphic evaluation algorithms $\mathtt{Add}$ and $\mathtt{Mult}$ satisfy

$$\mathtt{Dec}_{\mathtt{sk}}(\mathtt{Add}(\mathtt{ct}_1, \mathtt{ct}_2)) \approx \boldsymbol{m}_1 + \boldsymbol{m}_2,$$
$$\mathtt{Dec}_{\mathtt{sk}}(\mathtt{Mult}_{\mathtt{evk}}(\mathtt{ct}_1, \mathtt{ct}_2)) \approx \boldsymbol{m}_1 \odot \boldsymbol{m}_2$$

where $\odot$ denotes the Hadamard (component-wise) multiplication, i.e., addition and multiplication can be *internally* done in a Single Instruction Multi Data (SIMD) manner even in encrypted state. For more details of the scheme including the correctness and security analysis, we refer the readers to [13].

In order to manage a plaintext vector of the form $\boldsymbol{m} \in \mathbb{C}^K$ having length $K \leq N/2$ for some power-of-two divisor K of $N/2$, HEAAN encrypts $\boldsymbol{m}$ into a ciphertext of an $N/2$-dimensional vector $(\boldsymbol{m}||\cdots||\boldsymbol{m}) \in \mathbb{C}^{N/2}$. This implies that a ciphertext of $\boldsymbol{m} \in \mathbb{C}^K$ can be understood as a ciphertext of $(\boldsymbol{m}||\cdots||\boldsymbol{m}) \in \mathbb{C}^{K'}$ for powers-of-two K and K' satisfying $K \leq K' \leq N/2$.

Bootstrapping of HEAAN. Since the output ciphertext of a homomorphic multiplication has a reduced modulus by the scaling factor p compared to the input ciphertexts, the homomorphic operation should be stopped when the ciphertext modulus becomes so small that no more modulus reduction can be done. In other words, without some additional procedures, the HE scheme only supports polynomial operations with a bounded degree pre-determined by HEAAN parameters.

A *bootstrapping* algorithm, of which the concept was firstly proposed by Gentry [24], enables us to overcome the limitation on the depth of computation. The bootstrapping algorithm gets a ciphertext with the lowest modulus $\mathtt{ct} \in R^2_{q_1}$ as an input, and outputs a refreshed ciphertext $\mathtt{ct}' \in R^2_{q_{L'}}$ where L' is a pre-determined parameter smaller than L. The important fact is that the bootstrapping preserves the most significant bits of a plaintext, i.e., $\mathtt{Dec}_{\mathtt{sk}}(\mathtt{ct}) \approx \mathtt{Dec}_{\mathtt{sk}}(\mathtt{ct}')$. In 2018, a first bootstrapping algorithm for HEAAN was proposed by Cheon et al. [12], and later it was improved by several works concurrently [8,10].

Even though the performance of bootstrapping has been improved by active studies, the bootstrapping algorithm is still regarded as the most expensive part of HE. In the case of HEAAN, the performance of bootstrapping depends on the number of plaintext slots K; roughly the computational complexity is $O(\log K)$ considering SIMD operations of HEAAN.

2.3 Non-polynomial Operations in HEAAN

Since HEAAN basically supports homomorphic addition and multiplication, performing non-polynomial operations in HEAAN is clearly non-trivial. In this section we introduce how to perform the *division* and a comparison-related operation called *min-index* in word-wise HE including HEAAN, which are required for our mean-shift clustering algorithm. Note that the following methods are essentially efficient polynomial approximations for the target operations.

Division. The Goldschmidt's division algorithm [26] is an approximate algorithm to compute the inversion of a positive real number in $(0, 2)$, and has been used in various cryptographic applications [14,18] to deal with inversion and division operations through a polynomial evaluation. The algorithm approximates the inversion of $x \in (0, 2)$ by

$$\frac{1}{x} = \prod_{i=0}^{\infty} \left(1 + (1-x)^{2^i}\right) \approx \prod_{i=0}^{\zeta-1} \left(1 + (1-x)^{2^i}\right)$$

where ζ is a parameter we choose considering the approximation error. If the range of an input is $(0, m)$ for large $m > 0$ which is known, then the Goldschmidt's division algorithm can be easily generalized by simply scaling down the input into the range $(0, 2)$ and scaling up the output after the whole process.

Algorithm 1. $\texttt{Inv}(x; m, \zeta)$

Input: $0 < x < m$, $\zeta \in \mathbb{N}$
Output: an approximate value of $1/x$

1: $a_0 \leftarrow 2 - (2/m) \cdot x$
2: $b_0 \leftarrow 1 - (2/m) \cdot x$
3: **for** $i \leftarrow 0$ **to** $\zeta - 1$ **do**
4: $\quad b_{i+1} \leftarrow b_i^2$
5: $\quad a_{i+1} \leftarrow a_i \cdot (1 + b_{i+1})$
6: **end for**
7: **return** $(2/m) \cdot a_\zeta$

Min Index. In [14], Cheon et al. proposed the iterative algorithm `MaxIdx` to compute the max-index of an array of positive numbers that can be homomorphically computed by HEAAN efficiently. More precisely, for an input vector $\boldsymbol{x} = (x_1, x_2, .., x_m)$ where $x_i \in (0, 1)$ are distinct numbers, the output of the

max-index algorithm is a vector $\left(x_i^{2^t}/(\sum_{j=1}^m x_j^{2^t})\right)_{1\le i\le m}$ for sufficiently large $t > 0$, in which i-th component is close to 1 if x_i is the maximal element and is approximately 0 otherwise. If there are several maximal numbers, say $x_1, ..., x_\ell$ for $1 \le \ell \le m$ without loss of generality, the output vector is approximately $(1/\ell, 1/\ell, ..., 1/\ell, 0..., 0)$.

As a simple application of max-index, one can also compute the *min-index* of an array of positive numbers in $(0, 1)$ by running the `MaxIdx` algorithm for input $(1-x_1, 1-x_2, ..., 1-x_m)$. The following algorithm describes the min-index algorithm denoted by `MinIdx`.

Algorithm 2. `MinIdx`$((x_i)_{i=1}^m; t, \zeta)$

Input: $(x_1, ..., x_m) \in (0, 1)^m$ where $\ell \ge 1$ elements are minimal, $t \in \mathbb{N}$
Output: $(y_1, ..., y_m)$ where $y_i \approx 1/\ell$ if x_i is a minimal element and $y_i \approx 0$ otherwise;

$sum \leftarrow 0$
for $i \leftarrow 1$ **to** m **do**
  $y_i \leftarrow 1 - x_i$
  for $j \leftarrow 1$ **to** t **do**
    $y_i \leftarrow y_i \cdot y_i$
  end for
  $sum \leftarrow sum + y_i$
end for
$inv \leftarrow$ `Inv`$(sum; m, \zeta)$
for $i \leftarrow 1$ **to** m **do**
  $y_i \leftarrow y_i \cdot inv$ // $y_i \simeq (1 - x_i)^{2^t} / \sum_{j=1}^m (1 - x_j)^{2^t}$
end for
return $(y_1, ..., y_m)$

2.4 Mean-Shift Clustering

The mean-shift clustering algorithm is a *non-parametric* clustering technique which does not restrict *the shape of the clusters* and not require prior knowledge of *the number of clusters*. The goal of the algorithm is to cluster the given points by finding the local maxima (called *modes*) of a density function called *Kernel Density Estimator* (KDE), and this process is essentially done by the gradient descent algorithm. For given n-dimensional points $P_1, P_2, ..., P_p$ and a function $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\ge 0}$ so-called *kernel*, the KDE map $F : \mathbb{R}^n \to \mathbb{R}^n$ is defined as

$$F(\boldsymbol{x}) = \frac{1}{p} \cdot \sum_{i=1}^{p} K(\boldsymbol{x}, P_i).$$

The kernel K is defined by a profile $k : \mathbb{R} \to \mathbb{R}_{\ge 0}$ as $K(\boldsymbol{x}, \boldsymbol{y}) = c_k \cdot k(||\boldsymbol{x} - \boldsymbol{y}||^2)$ for some constant $c > 0$. Through a simple computation, one can check that

$\nabla F(\boldsymbol{x})$ is parallel to $\sum_{i=1}^{p} \frac{k'(||\boldsymbol{x}-P_i||^2)\cdot P_i}{\sum_{i=1}^{p} k'(||\boldsymbol{x}-P_i||^2)} - \boldsymbol{x}$ where k' is the derivative of k. As a result, the mean-shift process is to update the point $\boldsymbol{x}$ with

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + \left(\sum_{i=1}^{p} \frac{k'(||\boldsymbol{x} - P_i||^2)}{\sum_{j=1}^{p} k'(||\boldsymbol{x} - P_j||^2)} \cdot P_i - \boldsymbol{x} \right) = \sum_{i=1}^{p} \frac{k'(||\boldsymbol{x} - P_i||^2)}{\sum_{j=1}^{p} k'(||\boldsymbol{x} - P_j||^2)} \cdot P_i,$$

which is the weighted mean of given points. The most usual choices of the kernel function are the Gaussian kernel $K_G(\boldsymbol{x}, \boldsymbol{y}) = c_{k_G} \cdot \exp\left(-||\boldsymbol{x} - \boldsymbol{y}||^2/\sigma^2\right)$ and the Epanechnikov kernel $K_E(\boldsymbol{x}, \boldsymbol{y}) = c_{k_E} \cdot \max(0, 1 - ||\boldsymbol{x} - \boldsymbol{y}||^2/\sigma^2)$ for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ with an appropriate parameter $\sigma > 0$ and constants c_{k_G} and c_{k_E}. Algorithm 3 is a full description of the original mean-shift clustering algorithm with Gaussian kernel.

Algorithm 3. `MS-clustering-original`$(P = \{P_1, ..., P_p\}, T; \sigma)$

Input: $P_1, P_2, \cdots, P_p \in \mathbb{R}^n$, the number of iterations $T \in \mathbb{N}$
Output: Label vector M of given points $P_1, ..., P_p$

for $i \leftarrow 1$ **to** p **do**
  $M_i \leftarrow P_i$
end for
for $i \leftarrow 1$ **to** T **do**
  for $j \leftarrow 1$ **to** p **do**
    $sum \leftarrow 0$
    $A \leftarrow 0^d$
    for $k \leftarrow 1$ **to** p **do**
      $a \leftarrow \exp(-||P_k - M_j||^2/\sigma^2)$
      $A \leftarrow A + a \cdot P_k$
      $sum \leftarrow sum + a$
    end for
    $M_j \leftarrow (1/sum) \cdot A$
  end for
end for
return $M = (M_1, ..., M_p)$

Freedman-Kisilev Mean-Shift. A decade ago, Freedman and Kisilev [23] proposed a novel fast mean-shifting algorithm based on the random sampling. As the first step, for the given set $P = \{P_1, P_2, ..., P_p\}$ which consists of n-dimensional points, they randomly choose a subset $P' \subset P$ of the cardinality p'. Here the cardinality p' is indeed smaller than p but *should not be too small* so that the subset P' approximately conserves the distribution of the points. For example, if the random sampling factor p/p' is too high, then Freedman-Kisilev mean-shift algorithm shows a quite different result compared to that of the original mean-shift algorithm. After the random sampling phase, the second step is to run the original mean-shift algorithm only on the randomly chosen subset P' and obtain

the modes of KDE constructed by P', not P. Since only p' points are used for mean-shifting process, the computational complexity of this phase is $O(p'^2)$, not $O(p^2)$. The last step so-called "map-backwards" is to find the closest point in $P'_j \in P'$ for each point in $P_i \in P$ and then output the mode mean-shifted from P'_j. The last step takes $O(p' \cdot p)$ computational complexity, which is still smaller than $O(p^2)$. Note that the map-backwards, the last step in Freedman-Kisilev mean-shift algorithm, is not required in the original mean-shift algorithm, since every point converges to some mode which takes a role of the label in the original mean-shift algorithm.

2.5 Clustering Quality Evaluation Criteria

To evaluate the quality of our cluster analysis results, we bring two measures: accuracy and silhouette coefficient. The accuracy is measured by comparing the cluster analysis result and the given true label information. Let L_i and $C(P_i)$ be the true label and the label obtained by cluster analysis of the point P_i, respectively; then, the accuracy is calculated as

$$\mathsf{Accuracy} = \frac{|\{1 \le i \le p : L_i = C(P_i)\}|}{p}.$$

Note that the measure is valid only if the number of clusters of the given true label is equal to that of the cluster analysis result.

The silhouette coefficient [37] is another measure which evaluates the quality of cluster analysis, which does not require true label information to be given. Let Q_1,...,Q_k be the clusters of the given dataset P obtained by cluster analysis. For each point P_i which belongs to the cluster Q_{k_i}, we first define two functions A and B as

$$A(P_i) = \frac{1}{|Q_{k_i}| - 1} \cdot \sum_{\substack{P_\ell \in Q_{k_i} \\ \ell \neq i}} \mathrm{dist}(P_i, P_\ell), \quad B(P_i) = \min_{j \neq i} \frac{1}{|Q_{k_j}|} \cdot \sum_{P_\ell \in Q_{k_j}} \mathrm{dist}(P_i, P_\ell).$$

Then, the silhouette coefficient is defined as

$$\mathsf{SilhCoeff} = \frac{1}{p} \cdot \sum_{i=1}^{p} \frac{B(P_i) - A(P_i)}{\max(B(P_i), A(P_i))}$$

which indicates how well the points are clustered. It is clear that $-1 \le \mathsf{SilhCoeff} \le 1$, and the silhouette coefficient closer to 1 implies the better result of clustering.

3 HE-Friendly Modified Mean-Shift Clustering

In this section, we introduce several modifications on the mean-shift algorithm which can be efficiently performed by HE. One big drawback of the original

mean-shift algorithm to be implemented by HE is the evaluation of kernel functions. They usually contain non-polynomial operations, but these operations cannot be easily computed with HE algorithms. In order to overcome the problem, we suggest a new HE-friendly kernel function in Sect. 3.1 which is computationally efficient and shows a good performance.

Another big drawback of the original mean-shift algorithm to be implemented by HE is its high computational cost. The usual mean-shift process classifies data by seeking modes and mapping points to its corresponding mode at the same time. This strategy eventually forces us to perform mean-shift process on all data, so it is computationally inefficient to be implemented by HE which possibly accompanies more than hundreds or thousands times of overhead. In order to address this issue, we adopt a random sampling method called dust sampling and separate the total mean-shift clustering process into two phases: *mode-seeking phase* and *point-labeling phase*. One can check the details on these two phases in Sects. 3.2 and 3.3 respectively, and the full description of our modified mean-shift clustering algorithm is described in Sect. 3.4.

3.1 HE-Friendly Kernel

As described in Sect. 2.4, the most popular kernel functions for mean-shift algorithm are Gaussian kernel and Epanechnikov kernel. However, the derivatives of both kernel functions, which should be computed in the mean-shift clustering algorithm, are either exponential or discontinuous that cannot be directly computed with HE.

In order to overcome those drawbacks, we propose a new HE-friendly kernel function which is a polynomial. We aim to construct a kernel function that vanishes rapidly as its input goes far from the origin. Moreover, we also consider about reducing the number of multiplications during the computation of the kernel. For each $x \in [0, 1]$, our new profile k is calculated as following:

$$k(x) = (1 - x)^{2^{\Gamma}+1}. \tag{1}$$

The degree was set $2^{\Gamma} + 1$ to reduce the computational complexity of the derivative function k', which should be computed for mean-shift. Using this profile, a new HE-friendly kernel is defined as following: For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{B}_n(1/2)$, the kernel function K based on the profile k is

$$K(\boldsymbol{x}, \boldsymbol{y}) = c \cdot \left(1 - ||\boldsymbol{x} - \boldsymbol{y}||^2\right)^{2^{\Gamma}+1} \tag{2}$$

for some constant $c > 0$. The following algorithm, denoted by `Kernel`, shows a very simple computation of $k'(||\boldsymbol{x} - \boldsymbol{y}||^2)$ up to constant $-1/(2^{\Gamma} + 1)$. If one chooses bigger Γ, the kernel function will decrease more rapidly, so the mean-shift process will focus more on closer points. Conversely, if one chooses smaller Γ, the kernel function will decrease more slowly so that the mean-shift process references wider area.

Our new kernel function is composed of $(\Gamma + 1)$ multiplications and one constant addition, while Γ is relatively minute compared to the degree of the

Algorithm 4. $\texttt{Kernel}(\boldsymbol{x}, \boldsymbol{y}; \Gamma)$

Input: $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{B}_n(1/2)$, $\Gamma \in \mathbb{N}$
Output: HE-friendly kernel value between A and B
1: $a \leftarrow 1 - ||\boldsymbol{x} - \boldsymbol{y}||^2$
2: **for** $i \leftarrow 1$ **to** Γ **do**
3: $\quad a \leftarrow a^2$
4: **end for**
5: **return** a

kernel polynomial ($\Gamma = \log(degree)$). Thus, our new kernel function is very HE-friendly. At the same time, it is non-negative and strictly decreasing, so it satisfies the core conditions of a kernel function for mean-shift algorithm. Moreover, its rapid decreasing property provides a high performance for mean-shift algorithm. The performance of our new kernel function is experimentally proved under various datasets (See Sect. 4). In an unencrypted state, the mean-shift clustering with our kernel function shows almost same performance with that with the Gaussian kernel function on same datasets described in Sect. 4.1.

3.2 Mode-Seeking Phase

The biggest drawback of the original mean-shift clustering algorithm is its high time complexity. It requires super-linear operations in the number of data points. Since HE consumes considerably long time to compute each operation, it is strongly demanded to modify mean-shift algorithm for practical implementation with HE.

In order to overcome those drawbacks, we use random sampling to reduce the total number of operations for each mean-shift process. Instead of performing mean-shift on every point, we perform the mean-shift process only on selected points, which we shall call *dusts*. Each mean-shift process references all the data so that dusts move to proper modes of the KDE map generated by given data. After sufficiently many iterations, each dust converges to a mode, so we can seek all modes if we selected enough number of dusts.

Advantage of the Dust Sampling Method. Our modification has a great advantage on the number of operations. In the original mean-shift clustering algorithm, every point shifts its position by referencing all of the other points. Hence, it needs $O(p^2)$ operations for each loop where p is the number of given points. However, in our approach, only selected dusts shift their positions, so we can complete each mean-shift iteration with $O(p \cdot d)$ operations, where d is the number of selected dusts. This decreases the total number of operations, because we select relatively negligible number of dusts among numerous points.

Even though our approach requires less operations, its performance is acceptable. Since we use the KDE map over all given points, the dusts converge to modes exactly in the same way with the original mean-shift algorithm. Consequently, we can seek all modes by selecting sufficiently many dusts.

How to Sample Dusts? There are many possible ways to set the initial position of dusts. We consider two candidates of initialization strategy of the dusts. One is to uniformly select dusts from the space (so that can form a *grid*), and the other is to select dusts among the given points. The first strategy is tempting because it guarantees high probability to seek all the modes. However, as the dimension of the data set becomes higher, it requires too many number of dusts, which directly increases the total time complexity. On the other hand, the second strategy provides a stable performance with less number of dusts even if the dimension and shape of the data vary. Moreover, it chooses more dusts from the denser regions, so we can expect that it succeeds in detecting all centers of clusters. Thus, we use the second strategy, selecting dusts among given points as described in Algorithm 5.

Comparison to Freedman-Kisilev's Method. At first glance, our approach looks similar to that of Freedman and Kisilev [23]. Remark that they pick p' random samples among the data, and run the mean-shift algorithm only on the sampled points by referencing the KDE map generated by the sampled points.

Compared to Freedman-Kisilev mean-shift, the number of selected dusts d in our mean-shift can be set smaller than the number of randomly sampled points p'. While our sampling method uses the original KDE map, Freedman-Kisilev algorithm uses the KDE map generated by the sampled points. As a consequence, Freedman and Kisilev have to select substantially many samples to preserve the original KDE structure, while we do not have such restriction on the number of dusts.

Algorithm 5. `Mode-seeking`$(P = \{P_1, ..., P_p\}, d, T; \Gamma, \zeta)$

Input: Points $P_1, P_2, \cdots, P_p \in \mathbb{B}_n(1/2)$, the number of dusts $d \in \mathbb{N}$, the number of mean-shift iterations $T \in \mathbb{N}$

Output: Mean-shifted dusts $D_i \in \mathbb{B}_n(1/2)$ close to modes for $1 \leq i \leq d$

for $i \leftarrow 1$ **to** d **do**
  $D_i \leftarrow U(P)$ // selecting dusts among P_i's
end for
for $i \leftarrow 1$ **to** T **do**
  for $j \leftarrow 1$ **to** d **do**
    $sum \leftarrow 0$
    $A \leftarrow 0^d$
    for $k \leftarrow 1$ **to** p **do**
      $a \leftarrow \texttt{Kernel}(P_k, D_j; \Gamma)$
      $A \leftarrow A + a \cdot P_k$
      $sum \leftarrow sum + a$
    end for
    $D_j \leftarrow \texttt{Inv}(sum; p, \zeta) \cdot A$ // $D_j \leftarrow \sum_{i=1}^{p} \frac{k'(||D_j - P_i||^2)}{\sum_{\ell=1}^{p} k'(||D_j - P_\ell||^2)} \cdot P_i$
  end for
end for
return D

The computational complexity of each mean-shift process in Freedman and Kisilev's algorithm is $O(p'^2)$, while ours is $O(d \cdot p)$. If p' is large enough so that $d \cdot p < p'^2$, our mean-shift process might require even less computations. And even if p' has been set small enough so that $p'^2 < p \cdot d$, the computational complexity of the map-backwards process in Freedman-Kisilev mean-shift $O(p \cdot p')$ is still larger than corresponding point-labeling process in our mean-shift $O(p \cdot d)$ since $p' > d$. More importantly, the less number of selected dusts in our approach has a huge advantage on HE implementation. Bootstrapping is the most expensive part in HE, so minimizing the cost of bootstrapping, by reducing the number of bootstrappings or setting the number of plaintext slots as small as possible, is very important to optimize HE implementations. Since the mean-shift clustering algorithm requires very large amount of computations, we have to repeatedly execute bootstrapping on d dusts in the case of our algorithm and p' samples in the case of Freedman-Kisilev. Since $d < p'$, the total bootstrapping procedure takes much less time in our mean-shift algorithm than the Freedman-Kisilev mean-shift algorithm.

3.3 Point-Labeling Phase

Let us move on to the second phase, point-labeling. After finding all the modes, we should label each point by mapping it to its closest mode. A naive way to label a point P_i is as followings:

$$C_{\texttt{naive}}(P_i) = \texttt{argmin}_{1 \le j \le d} \texttt{dist}(D_j, P_i)$$

where each D_i denotes the mean-shifted dust after the mode-seeking phase. However, the `argmin` function is very hard to compute in HE, and furthermore this naive approach would label the points in the same cluster with different indices. For example, let two dusts D_1 and D_2 converge to a same mode M after the mean-shift process, and P_1 and P_2 are unselected points of which the closed dusts are D_1 and D_2 respectively. We expect P_1 and P_2 to be classified as a same cluster because both points are close to the same mode M. However, with the naive way of point-labeling above, $C_{\texttt{naive}}(P_1) = 1$ does not match with $C_{\texttt{naive}}(P_2) = 2$ due to the slight difference between D_1 and D_2.

Fortunately, utilizing `MinIdx` algorithm in Sect. 2.3 resolves both problems of the naive approach. Let us define a modified point-labeling function C' as

$$C'(P_i) = \texttt{MinIdx}\left((||P_i - D_k||^2)_{1 \le k \le d}; t, \zeta\right).$$

Since `MinIdx` algorithm consists of polynomial operations, it can be evaluated by HE for sure. Moreover, with appropriate parameters t and ζ, $\texttt{MinIdx}((x_1, ..., x_m); t, \zeta)$ outputs a vector close to $\left(\frac{1}{2}, \frac{1}{2}, 0, ..., 0\right)$ when x_1 and x_2 are (approximately) minimal among x_i's, rather than $(1, 0, ..., 0)$ or $(0, 1, ..., 0)$. Therefore, in the same setting to above, we get $C'(P_1) \simeq C'(P_2) \simeq \left(\frac{1}{2}, \frac{1}{2}, 0, ..., 0\right)$.

However, C' cannot be a complete solution if we consider the case that a lot of D_i's converge to a same mode. Let $D_1, ..., D_\ell$ converged to the same mode M after the mean-shifting process. Then for a point P_i that is close to the mode

Algorithm 6. $\texttt{Point-labeling}(P = \{P_1, ..., P_p\}, D = \{D_1, ..., D_d\}; \Gamma, \zeta, t)$

Input: $P_1, .., P_p \in \mathbb{B}_n(1/2)$, $D_1, ..., D_d \in \mathbb{B}_n(1/2)$, $\Gamma \in \mathbb{N}$
Output: Cluster index $C_i \in [0, 1]^d$ of each P_i for $1 \leq i \leq p$

for $i \leftarrow 1$ **to** d **do**
  $\texttt{NHBD}_i \leftarrow 0$
  for $j \leftarrow 1$ **to** d **do**
    $\texttt{NBHD}_i \leftarrow \texttt{NBHD}_i + \texttt{Kernel}(D_i, D_j; \Gamma)$
  end for
end for
$\texttt{NBHD} \leftarrow (\texttt{NBHD}_i)_{1 \leq i \leq d}$ // $\texttt{NBHD}_i = \sum_{j=1}^{d} \texttt{Kernel}(D_i, D_j; \Gamma)$
for $i \leftarrow 1$ **to** p **do**
  $C'_i \leftarrow \texttt{MinIdx}\left((||P_i - D_k||^2)_{1 \leq k \leq d}; t, \zeta\right)$
  $C_i \leftarrow C'_i \odot \texttt{NBHD}$
end for
return $C = (C_i)_{1 \leq i \leq p}$

M, it holds that $C'(P_i) \simeq \left(\frac{1}{\ell}, \frac{1}{\ell}, ..., \frac{1}{\ell}, 0, ..., 0\right)$. When ℓ is sufficiently large, we may not be able to distinguish between $\frac{1}{\ell}$ and an approximation error of $\texttt{MinIdx}$ attached to 0. We refine this problem by adopting a vector $\texttt{NBHD} \in \mathbb{R}^d$ of which i-th component indicates the number of D_j's very close to D_i:

$$\texttt{NBHD} = \left(\sum_{k=1}^{d} \texttt{Kernel}(D_j, D_k; \Gamma)\right)_{1 \leq j \leq d}$$

for proper parameter $\Gamma \geq 1$, and we define our final point-labeling function C as

$$C(P_i) = C'(P_i) \odot \texttt{NBHD}.$$

Since $\texttt{Kernel}(D_j, D_k; \Gamma)$ outputs approximately 1 if $D_j \simeq D_k$ and 0 otherwise, the j-th component $\texttt{NBHD}_i$ an approximate value of the number of dusts close to D_j. Therefore, each component of $C(P_i)$ is approximately 0 or 1 for $1 \leq i \leq p$. More precisely, for $1 \leq j \leq d$, $C(P_i)_j \simeq 1$ if and only if D_j is one of the closest dusts to P_i.

To sum up, with mean-shifted dusts $D = \{D_1, ..., D_d\}$, we label each point P_i by

$$C(P_i) = \texttt{MinIdx}\left((||P_i - D_k||^2)_{1 \leq k \leq d}; t, \zeta\right) \odot \left(\sum_{k=1}^{d} \texttt{Kernel}(D_j, D_k; \zeta)\right)_{1 \leq j \leq d}.$$

Parameters t and ζ control the accuracy of $\texttt{MinIdx}$, and the parameter ζ control the accuracy of counting the number of converged dusts in each mode. Note that the return type of C is a d-dimensional vector, in which the i-th component C_i denotes $C(P_i)$.

Other Approaches of Point-Labeling. Another possible choice of the point-labeling function is *coordinate-of-dust* function that simply returns the dust

closest to the input point, i.e., $C_{\texttt{coord}}(P_i) = D_{\texttt{argmin}_{1\le j\le d}\texttt{dist}(D_j,P_i)}$. However, the minimum distance between $C_{\texttt{coord}}(P_i)$'s cannot be bounded by any constant. This limitation makes it unclear to determine whether two points P_i and P_j satisfying $C_{\texttt{coord}}(P_i) \simeq C_{\texttt{coord}}(P_i)$ in some sense belong to the same cluster or not. Since we are using several approximate algorithms including `Mode-seeking`, this obscure situation occurs quite often. Therefore, $C_{\texttt{coord}}$ is not the best choice for point labeling.

Freedman and Kisilev [23] uses another strategy called the map-backwards strategy. In this strategy, we label points by referencing the initial position of dusts instead of their final position. For example, we can compute the label of each point $P_i \in P$ by a vector-matrix multiplication as followings:

$$C_{\texttt{back}}(P_i) = \texttt{MinIdx}\left((||P_i - D_j^0||^2)_{1\le j\le d}; t, \zeta\right) \cdot (\texttt{Kernel}(D_j, D_k))_{1\le j,k\le d}$$

where D_j^0 is the initial position of each $D_j \in D$. Note that we treat the first term as a $1 \times d$ matrix and the second term as $d \times d$ matrix, and multiply them by a matrix multiplication. As a result, the j-th entry of $C_{\texttt{back}}(P_i)$ would designate the set of dust-neighborhood of the dust closest to P_i at the initial state.

This strategy is also reasonable since the points close to the initial position of each dust are generally expected to move close to the same mode through the mean-shift process. We may regard this strategy as partitioning the points as several regions through the initial position of dusts. However, the map-backwards strategy shall be relatively inefficient compared to our point-labeling strategy in the perspective of HE implementation. In the map-backwards strategy with only small number of dusts, the sampled point in each partitioned regions might not completely represent the region. Thus, the map-backwards strategy essentially requires substantially many number of dusts. As we explained in Sect. 3.2, a less number of dusts is better for HE implementation. Furthermore, a vector-matrix multiplication in the map-backwards strategy is more expensive in HE compared to a Hadamard multiplication of two vectors in our point-labeling strategy.

3.4 Our Modified Mean-Shift Clustering Algorithm

In summary, our modified mean-shift clustering procedure is done by two phases: mode-seeking phase and point-labeling phase. In the first phase, we seek all the modes which are candidates for the centers of clusters, and in the second phase, we map each point to its closest mode with well-devised point-labeling function. Algorithm 7 describes our HE-friendly modified mean-shift clustering algorithm:

Complexity Analysis. In the mode-seeking phase, the mean-shift process is iterated for T times. For each iteration, we calculate the kernel value between all pairs of points and dusts. Note that the computational complexity of `Kernel` between two n-dimensional points is $O(n)$, so each mean-shift iteration takes $O(n \cdot d \cdot p)$; hence, the computational cost of `Mode-seeking` is $O(n \cdot d \cdot p \cdot T)$.

The point-labeling phase consists of calculating vectors `NBHD` and C_i', and Hadamard multiplications $\texttt{NBHD} \odot C_i'$ for $1 \le i \le p$. In order to obtain `NBHD`,

Algorithm 7. `Mean-shift-clustering`$(P = \{P_1, ..., P_p\}, d, T; \Gamma_1, \Gamma_2, \zeta_1, \zeta_2, t)$

Input: $P_1, P_2, \cdots, P_p \in \mathbb{B}_n(1/2)$, $\Gamma_1\Gamma_2, d, T \in \mathbb{N}$
Output: A label vector of $P_1, P_2, ..., P_p$
1: $D \leftarrow$ `Mode-seeking`$(P, d, T; \Gamma_1, \zeta_1)$
2: $C \leftarrow$ `Point-labeling`$(P, D; \Gamma_2, \zeta_2, t)$
3: **return** $C = (C_1, ..., C_p)$

we calculate the kernel values between all pairs of dusts, so it takes $O(n \cdot d^2)$ computations. Also, to calculate C'_i, we measure the distances from the given point to dusts, so it requires $O(n \cdot d)$ computations. Note that the cost $O(n)$ of a Hadamard multiplication is negligible. As a result, the computational cost of `Point-labeling` is $O(n \cdot d \cdot p)$ because d is always strictly smaller than p. To sum up, the cost of mode-seeking phase is $O(n \cdot d \cdot p \cdot T)$, and that of point-labeling phase is $O(n \cdot d \cdot p)$. Consequently, the computational cost of our algorithm is $O(n \cdot d \cdot p \cdot T)$.

We can reduce the computational cost of `Mean-shift-clustering` by at most $N/2$, since HEAAN supports $N/2$ parallel computations in a SIMD manner where N is a HEAAN parameter. Fortunately, we can apply SIMD efficiently to our algorithm. The most heaviest parts of our algorithm are mean-shift process and `MinIdx`, both of which require $O(n \cdot p \cdot d)$ computations. For mean-shift process, we compute kernel values between all pairs of points and dusts. When we have one ciphertext of

$$(P_1 \,||\, P_2 \,||\, \cdots \,||\, P_p \,||\, P_1 \,||\, P_2 \,||\, \cdots \,||\, P_p \,||\, \cdots \,||\, P_1 \,||\, P_2 \,||\, \cdots \,||\, P_p)$$

and another ciphertext of

$$(D_1 \,||\, D_1 \,||\, \cdots \,||\, D_1 \,||\, D_2 \,||\, D_2 \,||\, \cdots \,||\, D_2 \,||\, \cdots \,||\, D_k \,||\, D_k \,||\, \cdots \,||\, D_k)$$

with $k = \frac{N}{2np}$, then we can compute $k \cdot p = \frac{N}{2n}$ kernel computations simultaneously, and the computational cost of each kernel reduces to $O(\log n)$. As a result, we can run `Mode-seeking` with $O\left(\frac{n^2 \cdot d \cdot p \cdot T}{\log n \cdot N}\right)$ computations in HEAAN. Similarly we can reduce the number of computations for `Point-labeling` as well. Thereby the total computational cost of our algorithm would be $O\left(\frac{n^2 \cdot d \cdot p \cdot T}{\log n \cdot N}\right)$.

4 Experimental Results

4.1 Dataset Description

In order to monitor the performance, we implement our algorithm over four datasets (Hepta, Tetra, TwoDiamonds, Lsun) with true labels which are publicly accessible from fundamental clustering problems suite (FCPS) [42] and one large-scale dataset (LargeScale) randomly generated by ourselves. LargeScale dataset consists of four clusters following Gaussian distributions with small variance and distinct centers. Table 1 describes the properties of each dataset (Fig. 2):

Table 1. Short descriptions of the datasets

Dataset	Dimension	# Data	# Clusters	Property
Hepta	3	212	7	`Different densities`
Tetra	3	400	4	`Big and touching clusters`
TwoDiamonds	2	800	2	`Touching clusters`
Lsun	2	400	3	`Different shapes`
LargeScale	4	262,144	4	`Numerous points`

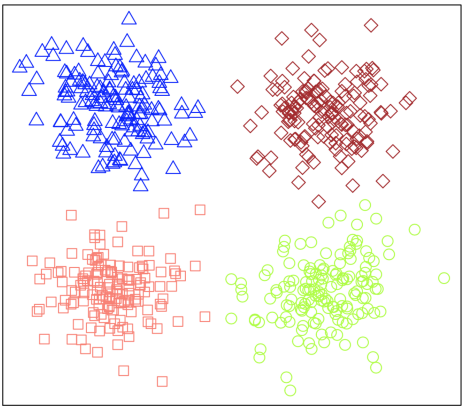

Fig. 2. A visualization of LargeScale dataset

4.2 Parameter Selection

Our implementation is based on the approximate HE library HEAAN [1,13]. We set HEAAN parameters $(N, q_L, \mathsf{h}, \chi_{\mathtt{err}})$ to satisfy 128-bit security, where N is the ring dimension, q_L is the initial modulus of a ciphertext, h is a hamming weight of a secret polynomial, and $\chi_{\mathtt{err}}$ is the error distribution. As mentioned in Sect. 2.2, we used Albrecht's security estimator [2,3] to estimate the bit security of those HEAAN parameters. Note that since the modulus of the evaluation key evk is q_L^2, the input on the security estimator is a tuple $(N, Q = q_L^2, \mathsf{h}, \chi_{\mathtt{err}})$. As a result, we set HEAAN parameters $N = 2^{17}$ and $\log q_L = 1480$, and we followed the default setting of HEAAN library [1] for error and secret distributions $\chi_{\mathtt{err}}$, $\chi_{\mathtt{enc}}$ and$\chi_{\mathtt{key}}$.

We flexibly chose the clustering parameters T, Γ_1, Γ_2, ζ_1, ζ_2 and t for each dataset to optimize the implementation results. Let us denote the a tuple of parameters by $\mathsf{params} = (T, \Gamma_1, \Gamma_2, \zeta_1, \zeta_2, t)$. In the case of Hepta dataset, the best choice of parameters was $\mathsf{params} = (5, 6, 6, 4, 4, 6)$, while $\mathsf{params} = (7, 6, 6, 5, 6, 6)$ was the best for Tetra dataset, $\mathsf{params} = (8, 5, 5, 5, 6, 5)$ was the best for TwoDiamonds dataset, $\mathsf{params} = (5, 6, 5, 5, 8, 6)$ was the best for Lsun dataset, and $\mathsf{params} = (5, 5, 5, 3, 3, 5)$ was the best for LargeScale dataset. We set the number of dusts to be as small as possible (e.g., $d = 8$) to reduce the cost of bootstrapping.

Table 2. Experimental results for various datasets with 8 threads

Dataset	Comp. Time	Memory	Quality Evaluation	
			Accuracy	SilhCoeff
Hepta	25 min	10.7 GB	212/212	0.702 (0.702)
Tetra	36 min	10.7 GB	400/400	0.504 (0.504)
TwoDiamonds	38 min	9.6 GB	792/800	0.478 (0.485)
Lsun	24 min	9.4 GB	-	0.577 (0.443)
LargeScale	82 min	20.7 GB	262127/262144	0.781 (0.781)

4.3 Experimental Results

In this subsection, we present experimental results on our mean-shift clustering algorithm based on HEAAN. All experiments were performed on C++11 standard and implemented on Linux with Intel Xeon CPU E5-2620 v4 at 2.10 GHz processor.

In Table 2, we present the performance and quality of our algorithm on various datasets. We use 8 threads for all experiments here. We describe the accuracy value by presenting both the number of well-classified points and the total number of points. We present two silhouette coefficients; the one without bracket is the silhouette coefficient of our clustering results, and the other one with bracket is that of the true labels.

We complete the clustering on various datasets within a few dozens of minutes. In the case of FCPS datasets, their sizes are much smaller than the number of HEAAN plaintext slots we can manage. On the other hand, the size of LargeScale dataset is big enough so that we can use full slots; therefore, we can fully utilize SIMD of HEAAN for the LargeScale dataset. Consequently, the performance of our algorithm for LargeScale dataset is quite nice in spite of its huge size.

For all the five datasets, our algorithm achieves high accuracy. In the case of Hepta, Tetra and LargeScale datasets, we succeed in labeling all data points by its exact true label. For the TwoDiamonds dataset, we succeed in classifying 792 points out of 800 points properly. Even for the rest 8 points, the label vector of each point is close to its true label.

In the case of the Lsun dataset, our algorithm results in four clusters while there are only three clusters in the true labels. Thereby, it is impossible to measure the accuracy by comparing with the true labels, so one may doubt that

our algorithm is inadequate to the Lsun dataset. However, it is also reasonable to classify the Lsun dataset with 4 clusters. In fact, our result shows even a better quality in aspect of the silhouette coefficient. The silhouette coefficient for our clustering result is 0.577, and that for the true labels is 0.443.

We also checked the performance of our algorithm with several numbers of threads for the Lsun dataset as described in Table 3. With a single thread, it consumes 9.4 GB memory and takes 83 min. This result is much faster than the result of the previous work in [29], which takes 25.79 days to complete a clustering process for the same Lsun dataset. Obviously we can speed up the performance by using much more number of threads. For example, the running time can be reduced to 16 min with just small overhead of memory when we use 20 threads.

Table 3. Experimental results for various # threads with the Lsun dataset

1 Thread		8 Threads		20 Threads	
Time	Memory	Time	Memory	Time	Memory
83 min	9.4 GB	24 min	9.4 GB	16 min	10.0 GB

Comparison to Freedman-Kisilev's Method. The experimental results of Freedman-Kisilev mean-shift clustering on the Tetra dataset under various p', the number of sampled points (see Sect. 2.4), shows how marginal p' may contaminate the performance. Note that our sampling method achieves 400/400 accuracy on the Tetra dataset with only 8 dusts. In contrast, when p' is either 8 or 16, Freedman-Kisilev algorithm even fails to detect the correct modes from the original data. It detects only three clusters while there actually exist four clusters; it classifies two different clusters as a single cluster. Thus, the results on when p' is either 8 or 16 are not even comparable with the answer. This supports the argument that the KDE map of Freedman-Kisilev mean-shift may not fully represent the original KDE map unless p' is sufficiently big.

When p' is bigger than 16, Freedman-Kisilev algorithm succeed in detecting four clusters as expected. However, the accuracy under each $p' = 32, 64, 128, 256$ is $377/400, 368/400, 393/400, 399/400$ respectively, while our sampling method achieves 400/400 with only 8 dusts. This implies that the approximate KDE map of Freedman-Kisilev mean-shift may indicate modes with possible errors.

As a consequence, Freedman and Kisilev have to select substantially many samples that can preserve the original KDE structure in some sense, while we do not have such restriction on the number of dusts.

Acknowledgement. This work was supported in part by the Institute for Information & Communications Technology Promotion (IITP) Grant through the Korean

Government (MSIT), (Development of lattice-based post-quantum public-key cryptographic schemes), under Grant 2017-0-00616, and in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) (No. 2017R1A5A1015626). We also thank anonymous reviewers of SAC'19 for very usual comments.

References

1. HEAAN Library (2017). https://github.com/snucrypto/HEAAN
2. Albrecht, M.R.: A Sage Module for estimating the concrete security of Learning with Errors instances (2017). https://bitbucket.org/malb/lwe-estimator
3. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Math. Cryptol. **9**(3), 169–203 (2015)
4. Almutairi, N., Coenen, F., Dures, K.: K-means clustering using homomorphic encryption and an updatable distance matrix: secure third party data clustering with limited data owner interaction. In: Bellatreche, L., Chakravarthy, S. (eds.) DaWaK 2017. LNCS, vol. 10440, pp. 274–285. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64283-3_20
5. Bonte, C., Vercauteren, F.: Privacy-preserving logistic regression training. Cryptology ePrint Archive, Report 2018/233 (2018). https://eprint.iacr.org/2018/233
6. Bourse, F., Minelli, M., Minihold, M., Paillier, P.: Fast homomorphic evaluation of deep discretized neural networks. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10993, pp. 483–512. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96878-0_17
7. Bunn, P., Ostrovsky, R.: Secure two-party k-means clustering. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS 2007, New York, NY, USA, pp. 486–497. ACM (2007)
8. Chen, H., Chillotti, I., Song, Y.: Improved bootstrapping for approximate homomorphic encryption. Cryptology ePrint Archive, Report 2018/1043 (2018). http://eprint.iacr.org/2018/1043. To appear EUROCRYPT 2019
9. Chen, H., et al.: Logistic regression over encrypted data from fully homomorphic encryption. Cryptology ePrint Archive, Report 2018/462 (2018). https://eprint.iacr.org/2018/462
10. Cheon, J.H., Han, K., Hhan, M.: Faster homomorphic discrete fourier transforms and improved FHE bootstrapping. Cryptology ePrint Archive, Report 2018/1073 (2018). https://eprint.iacr.org/2018/1073. To appear IEEE Access
11. Cheon, J.H., et al.: Toward a secure drone system: flying with real-time homomorphic authenticated encryption. IEEE Access **6**, 24325–24339 (2018)
12. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 360–384. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_14
13. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 409–437. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_15
14. Cheon, J.H., Kim, D., Kim, D., Lee, H.H., Lee, K.: Numerical methods for comparison on homomorphically encrypted numbers. Cryptology ePrint Archive, Report 2019/417 (2019). https://eprint.iacr.org/2019/417, To appear ASIACRYPT 2019

15. Cheon, J.H., Kim, D., Kim, Y., Song, Y.: Ensemble method for privacy-preserving logistic regression based on homomorphic encryption. IEEE Access **6**, 46938–46948 (2018)
16. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: bootstrapping in less than 0.1 seconds. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 3–33. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_1
17. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 377–408. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_14
18. Cho, H., Wu, D.J., Berger, B.: Secure genome-wide association analysis using multiparty computation. Nat. Biotechnol. **36**(6), 547 (2018)
19. Crawford, J.L., Gentry, C., Halevi, S., Platt, D., Shoup, V.: Doing real work with FHE: the case of logistic regression (2018)
20. Dhillon, I.S., Marcotte, E.M., Roshan, U.: Diametrical clustering for identifying anti-correlated gene clusters. Bioinformatics **19**(13), 1612–1619 (2003)
21. Doganay, M.C., Pedersen, T.B., Saygin, Y., Savaş, E., Levi, A.: Distributed privacy preserving k-means clustering with additive secret sharing. In: Proceedings of the 2008 International Workshop on Privacy and Anonymity in Information Society, PAIS 2008, New York, NY, USA, pp. 3–11. ACM (2008)
22. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley, Hoboken (2012)
23. Freedman, D., Kisilev, P.: Fast mean shift by compact density representation. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1818–1825. IEEE (2009)
24. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009). http://crypto.stanford.edu/craig
25. Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., Wernsing, J.: CryptoNets: applying neural networks to encrypted data with high throughput and accuracy. In: International Conference on Machine Learning, pp. 201–210 (2016)
26. Goldschmidt, R.E.: Applications of division by convergence. Ph.D. thesis, Massachusetts Institute of Technology (1964)
27. Han, K., Hong, S., Cheon, J.H., Park, D.: Logistic regression on homomorphic encrypted data at scale (2019)
28. Jagannathan, G., Wright, R.N.: Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, KDD 2005, New York, NY, USA, pp. 593–599. ACM (2005)
29. Jäschke, A., Armknecht, F.: Unsupervised machine learning on encrypted data. In: Cid, C., Jacobson Jr., M. (eds.) SAC 2018. LNCS, vol. 11349, pp. 453–478. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-10970-7_21
30. Kim, A., Song, Y., Kim, M., Lee, K., Cheon, J.H.: Logistic regression model training based on the approximate homomorphic encryption. BMC Med. Genomics **11**(4), 83 (2018)
31. Kim, M., Song, Y., Wang, S., Xia, Y., Jiang, X.: Secure logistic regression based on homomorphic encryption: design and evaluation. JMIR Med. Inform. **6**(2), e19 (2018)
32. Liu, D.: Practical fully homomorphic encryption without noise reduction. Cryptology ePrint Archive, Report 2015/468 (2015). https://eprint.iacr.org/2015/468

33. Liu, X., et al.: Outsourcing two-party privacy preserving k-means clustering protocol in wireless sensor networks. In: 2015 11th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN), pp. 124–133. IEEE (2015)
34. Malik, M.B., Ghazi, M.A., Ali, R.: Privacy preserving data mining techniques: current scenario and future prospects. In: 2012 Third International Conference on Computer and Communication Technology (ICCCT), pp. 26–32. IEEE (2012)
35. Meskine, F., Nait-Bahloul, S.: Privacy preserving k-means clustering: a survey research. Int. Arab J. Inf. Technol. **9**, 03 (2012)
36. Pouget, F., Dacier, M., et al.: Honeypot-based forensics. In: AusCERT Asia Pacific Information Technology Security Conference (2004)
37. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**, 53–65 (1987)
38. Sakuma, J., Kobayashi, S.: Large-scale k-means clustering with user-centric privacy preservation. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 320–332. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68125-0_29
39. Samet, S., Miri, A., Orozco-Barbosa, L.: Privacy preserving k-means clustering in multi-party environment, January 2007
40. Su, C., Bao, F., Zhou, J., Takagi, T., Sakurai, K.: Privacy-preserving two-party k-means clustering via secure approximation. In: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops - Volume 01, AINAW 2007, Washington, DC, USA, pp. 385–391. IEEE Computer Society (2007)
41. Sugar, C.A., James, G.M.: Finding the number of clusters in a dataset: an information-theoretic approach. J. Am. Stat. Assoc. **98**(463), 750–763 (2003)
42. Ultsch, A.: Clustering with SOM: U*C. In: Proceedings of Workshop on Self-Organizing Maps, Paris, France, pp. 75–82 (2005). https://www.uni-marburg.de/fb12/arbeitsgruppen/datenbionik/data?language_sync=1
43. Vaidya, J., Clifton, C.: Privacy-preserving k-means clustering over vertically partitioned data. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2003, New York, NY, USA, pp. 206–215. ACM (2003)
44. Vinoth, K.J., Santhi, V.: A brief survey on privacy preserving techniques in data mining. IOSR J. Comput. Eng. (IOSR-JCE) **18**, 47–51 (2016)
45. Wang, S., et al.: HEALER: homomorphic computation of exact logistic regression for secure rare disease variants analysis in GWAS. Bioinformatics **32**(2), 211–218 (2016)
46. Wang, Y.: Notes on two fully homomorphic encryption schemes without bootstrapping. IACR Cryptology ePrint Archive, 2015:519 (2015)