Privacy Preserving Approximate K-means Clustering

Chandan Biswas
Indian Statistical Institute, Kolkata
chandanbiswas08_r@isical.ac.in

Debasis Ganguly
IBM Research Lab, Dublin, Ireland
debasis.ganguly1@ie.ibm.com

Dwaipayan Roy*
GESIS - Leibniz Institute for the Social Sciences, Germany
dwaipayan.roy@gesis.org

Ujjwal Bhattacharya
Indian Statistical Institute, Kolkata, India
ujjwal@isical.ac.in

ABSTRACT

Privacy preserving computation is of utmost importance in a cloud computing environment where a client often requires to send sensitive data to servers offering computing services over untrusted networks. Eavesdropping over the network or malware at the server may lead to leaking sensitive information from the data. To prevent information leakage, we propose to encode the input data in such a way that, firstly, it should be difficult to decode it back to the true data, and secondly, the computational results obtained with the encoded data should not be considerably different from those obtained with the true data. Specifically, the computational activity that we focus on is the K-means clustering, which is widely used for many data mining tasks. Our proposed variant of the K-means algorithm is capable of privacy preservation in the sense that it requires as input only binary encoded data, and is not allowed to access the true data vectors at any stage of the computation. During intermediate stages of K-means computation, our algorithm is able to effectively process the inputs with incomplete information seeking to yield outputs relatively close to the complete information (non-encoded) case. Specifically, we propose a mixture of Gaussians based approximation method to estimate the centroid vectors effectively at each stage of K-means computation with information only from the encoded data vectors. Evaluation on real datasets show that the proposed methods yields comparable clustering effectiveness in comparison to the standard K-means algorithm on image clustering (MNIST-8M dataset), and in fact outperforms the standard K-means on text clustering (ODPtweets dataset).

CCS CONCEPTS

• **Information systems → Clustering and classification**; • **Security and privacy → Management and querying of encrypted data**;

KEYWORDS

Privacy Preservation, K-means Clustering, Centroid Estimation.

1 INTRODUCTION

Modern advances in software engineering have led to deploying software as services (known as SaaS), which provides an important advantage to organizations to focus on their core businesses instead of expending resources on computer infrastructure and maintenance. Consider for example, a 'big-data' clustering SaaS, which takes as input a set of data instances, performs the computations for data clustering on the server side, and returns as output a partitioning of the data to the client.

However, this ubiquitous use of service oriented computational architecture may lead to leakage of information from the input data that a client needs to send to a SaaS component. This information leakage may happen either due to eavesdropping activities in the network or due to malware executed on the servers with intentions of stealing information from the input data. Even when the data appears to be seemingly anonymous with suppressed sensitive information, intelligent processing of the data can reveal sensitive information, such as the infamous *AOL search query data scandal* [1] which exposed the personal identity, or the case of revealing the identities of authors with the help of stylometric features [2].

A solution to preserve data integrity is to *encode* the data in a way that it becomes difficult for any information stealing malware to detect the sensitive information from it. For example, existing literature in differential privacy has proposed a range of approaches for data protection, ranging from pseudo-anonymization of data [3], to adding noise to the data for protecting author information [2] (see [4] for a survey). Each such data-protection initiated transformation needs to achieve a trade-off between two objectives - i) ensure that attacks on the encoded data have low likelihood of success, and ii) the quality of the final output does not change significantly as a result of the transformation.

In this paper, we focus our attention on the latter objective, i.e., ensuring that the output obtained on processing the non-encoded data is not significantly different than the one obtained after encoding the input. The problem, that we particularly focus on, is that of clustering a given set of input vectors. In contrast to assuming a structured form of the input in terms of a database of attribute value lists, as common in existing research on differential privacy focusing on the effectiveness of data protection approaches against deanonymization attacks (see e.g. [3, 5, 6]), we rather focus on a general form of input (real-valued vectors), similar to [2].

In our work, we employ a binary transformation of the real-valued data, i.e. we apply a function $\phi : \mathbb{R}^p \mapsto \{0, 1\}^m$ to transform every p dimensional real-valued input data vector to a binary vector of m bits. The main advantage of the binary transformation, in

[*] Research conducted during the author's PhD studentship at Indian Statistical Institute.

CIKM '19, November 3–7, 2019, Beijing, China
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357969

particular, is that it enables much faster transmission of the data over the network and processing of the data on the server side. This is because it requires only $m/8$ bytes to store a binary vector of m bits, whereas storing a p dimensional real vector requires at least $p \times 4$ bytes of memory.

Next, after encoding the data, we focus on the problem of K-means clustering on this encoded data. Since it is known that a general class of binary transformation functions of real-valued data is a lossy transformation [7, 8], it is important to modify the K-means clustering algorithm with an objective to make it work well with incomplete information. Indeed, this forms the core of our research in this paper, where we propose a modified K-means algorithm which works under an imposed privacy preservation constraint that it can access only the encoded input. This is in contrast to existing research on fast approximate K-means approaches (see e.g. [9, 10]) which make use of the encoded data vectors in addition to the original ones during different stages of K-means execution.

Our main contribution in this paper is a modified K-means algorithm that respects the privacy preservation constraint, which we call *PPK-means* (privacy preserving K-means). The constraint makes it imperative to devise an effective method to estimate the centroid vectors during K-means iterations with the incomplete information from the binary encoded input data vectors. Informally speaking, the closer the estimated centroid vectors will be to the true centroids (computed with the complete information from the non-encoded data vectors without the privacy preservation constraint), potentially better will be the output of the clustering algorithm. To this end, we propose a Gaussian mixture model based solution to estimate the bit values of the centroid vectors during the intermediate computational steps. For more reliable estimation of the centroid vectors, we make available for the purpose of computation additional information in the form of aggregated statistics of projected values of the data vectors along a set of randomly chosen basis vectors.

We evaluate our proposed method on a set of both synthetic and real datasets. In comparison to standard K-means, our proposed method, PPK-means, shows significant improvements in terms of latency, without significantly decreasing the clustering effectiveness. Further, our proposed method outperforms the standard K-Means algorithm for clustering a large collection of short documents (tweets).

The rest of the paper is organized as follows. Section 2 provides an overview of the literature on fast approximate K-means and privacy preservation based computation. In Section 3, we briefly overview the characteristics of the two specific vector spaces that we consider in our work, namely Euclidean and Hamming. Section 4 formally introduces the concepts that are used to estimate the centroids during K-means iterations under the privacy preservation constraint. Section 5 describes our proposed method that uses mixture of Gaussians based centroid estimation from a set of encoded vectors and global statistics on the input data. In Section 6, we generalize the Gaussian estimation for multiple components. Section 7 presents the experiment setup, namely baselines, parameter tuning, and evaluation metrics. This is followed Section 8, where we present the results of our experiments on synthetic and real datasets (images and text). Finally, Section 9 concludes the paper with directions for future work.

2 RELATED WORK

Privacy Preserving Computing. Existing studies in machine learning have attempted to achieve the dual objective of privacy preservation (minimizing leakage of sensitive information) and model preservation (maximizing the performance of an algorithm on the encoded data), e.g. the work in [11] learns a transformation function to simultaneously minimize the likelihood of predicting missing values from the data and also minimizing a linear regression loss. Two major differences of our work with respect to [11] are that, firstly, we focus on a different objective, namely that of *clustering*, which in contrast to the objective of linear regression in [11], is unsupervised in nature, and secondly, the transformation function in our case is a binary one instead of a low rank approximation of [11], thus ensuring much faster execution.

In contrast to our client-server setting of K-means computation, the authors in [12] address the distributed computing case where the K-means computation is securely distributed over computing resources before employing secure key exchange protocols for computing the centroids and the closest cluster centres. Researchers in [13] proposed an attribute generalization based algorithm to abstract out specific instances of values of attributes, e.g. replacing attribute values such as ‘dancers’ and ‘writers’ with the more general value ‘artists’.

Similar to our approach of binary encoding the data with additional information about the averages and the variances of values projected along basis vectors, the work in [14] shares additional information of the form $f(x) = g(x_i)$, where x_i denotes the i^{th} row of a database and g maps database rows to $[0, 1]$.

K-means on Hamming Space. Since our proposed privacy preservation based K-means is based on binary encoding of data, we now review some existing K-means clustering variations that work with binary data. For example, the work in [9, 15] represented data vectors as binary codes to perform clustering. While the study in [15] defined a cluster centroid as the component-wise median of constituent vectors of a cluster, the authors of [16] obtained sparse cluster centers by applying $L1$-ball projection on each cluster center during each mini-batch iteration of K-means, which contributed to reduction in computational cost.

The idea of PQK-means involves representing input real-valued vectors as short codes by applying product quantization [17] and then clustering them by making use of hashing on the PQ codes during the cluster assignment step and sparse voting during updating the centroids [18]. The main limitation of PQ codes is that it has to rely on fixed subspaces of the data. In contrast, the JL transformation [7, 19] used to encode the data vectors in our method is able to preserve more information about the data by taking projections along orthogonal basis vectors. Similar to our method, the study in [9] uses random basis vectors to encode the input data in binary. However, during intermediate steps, the algorithm makes use of the original data vectors to modify the basis vectors, which leads to violating the privacy preservation constraint.

3 BACKGROUND

3.1 Euclidean and Hamming Spaces

A vector space V is represented by $\mathcal{V}$: a set of vectors with each component belonging to a specific domain (e.g. real numbers) and d: a distance metric, which takes two vectors as input and outputs a non-negative real number. More formally,

$$V : (\mathcal{V}, d);\ d : (\mathbf{x}, \mathbf{y}) \mapsto \mathbb{R},\ \mathbf{x}, \mathbf{y} \in \mathcal{V}. \tag{1}$$

The two vector spaces that are relevant in the context of our problem of privacy preserving clustering are i) $\mathbb{R}^p$: a p dimensional real vector space with L_2 (Euclidean) distance metric, and ii) $\mathbb{H}^m$: an m dimensional Hamming space of vectors with binary (0/1) components with L_1 distance metric, commonly known as the Hamming distance (d_H). For different applications, it is useful to transform points from one space to another. A transformation is usually meaningful if it is *distance preserving*, i.e., two nearby points continue to remain in close proximity post transformation.

In the context of our problem, let ϕ be the transformation function which maps points from a p dimensional Euclidean space to points in a Hamming space of m dimensions, i.e.,

$$\phi : \mathbf{w} \in \mathbb{R}^p \mapsto \mathbf{x} \in \mathbb{H}^m \tag{2}$$

The most common function for transforming points in an Euclidean vector space to those in the Hamming space, is the locality sensitive hash function (LSH) proposed in [8]. The LSH based approach involves randomly choosing a set of m basis vectors $\mathfrak{B}$; each point is then transformed by computing signs of projections of the point along these m basis vectors yielding the m components of the transformed point in the Hamming space (the transformed point is often called a *signature* of the original data point). Formally speaking, the i^{th} component of the transformed vector in Hamming space $\mathbb{H}^m$ is given by

$$x_i = \text{sgn}(\mathbf{w}.\mathbf{b}_i), \tag{3}$$

where $\mathbf{w}$ is an input data vector in the space $\mathbb{R}^p$, $\mathbf{b}_i \in \mathfrak{B}$ is the i^{th} basis vector, and sgn(z) is the *sign* function.

As per Johnson-Lindenstrauss (JL) lemma, it is known that this transformation is distance preserving [20]. The superbit algorithm, proposed in [19], improves on the likelihood of semantic hashing (similar signatures corresponding to similar points and dissimilar signatures otherwise) by applying orthogonalization on the randomly chosen basis vectors with the help of Gram-Schmidt algorithm, which we specifically use in this paper as a definition of the transformation function, ϕ.

At this point, we mention that in contrast to the standard notion of *differential privacy*, which applies for relational data comprising a set of attribute-value pairs, in our work, we consider privacy preservation (specifically during K-means computation) of real-valued data only. This means that instead of enforcing differential privacy, all we need to ensure in the context of our problem is that it should be difficult for an adversary to compute the true data vectors from the encoded vectors sent to a server for K-means computation. In fact, the notion of privacy preserving computing, which we use in this paper, relies on the observation in [8] that without knowing the set of basis vectors, it is computationally difficult to find an inverse function of the JL transformation transformation ϕ, that we use (Equation 2) to encode the vectors before sending them to the server.

4 COMPUTATION OF CLUSTER CENTROIDS

4.1 Vector Sum for Centroid Computation

The K centroid vectors during an iteration of K-means algorithm in the Euclidean space of data vectors is given by

$$\mathbf{c}^{\text{k}} = \frac{1}{|W^k|} \sum_{\mathbf{w} \in W^k} \mathbf{w},\ \ k = 1, \ldots, K \tag{4}$$

where W^k denotes the set of vectors in the k^{th} partition. Note that the true computation of the centroid vectors involves making use of the true data points $\mathbf{w}$'s.

Under privacy preservation constraints, the true data vectors $\mathbf{w}$'s are not available. A way to compute the centroid vectors under privacy preservation constraints is thus to compute centroids in the Hamming space. Formally speaking, the Hamming space represents a modulo 2 finite field (commonly denoted as $GF(2)$), where the (closure ensuring) sum operation is defined as

$$\mathbf{x} \oplus \mathbf{y} = \mathbf{z},\ \text{ where } z_i = (x_i + y_i) \bmod 2\ \in \{0, 1\}. \tag{5}$$

With this definition of the sum operator, the centroid vector in the Hamming space can be computed as

$$\mathbf{h}^k = \bigoplus_{\mathbf{x} \in X^k} \mathbf{x}, \tag{6}$$

where the i^{th} component of the vector $\mathbf{h}^k$, denoted by $\mathbf{h}_i^k \in \{0, 1\}$, is given as

$$\mathbf{h}_i^k = (\sum_{\mathbf{x} \in X^k} \mathbf{x}_i) \bmod 2, \tag{7}$$

where X^k denotes the set of vectors in the k^{th} partition of the Hamming space of encoded data vectors.

This way of computing the centroids of real-valued vectors, transformed (encoded) in the Hamming space is not optimal because of the apparent inconsistencies in the properties of the transformation function (Equation 3) and the modulo 2 addition. To illustrate with an example, consider adding the i^{th} components of two binary vectors both of which are 1, i.e. in other words, the corresponding true data vectors in the Euclidean space yield positive projection values over the i^{th} basis. The projection of the sum vector (in the true data space $\mathbb{R}^p$) over the i^{th} basis must then also be positive, and indeed the i^{th} component of the binary vector (in the JL transformed Hamming space) for the sum must also be encoded as '1' (as per Equation 3). More formally, due to the distributional property of the vector addition operation in Euclidean space,

$$\begin{aligned}(\mathbf{w} + \mathbf{v}).\mathbf{b}_i &= \mathbf{w}.\mathbf{b}_i + \mathbf{v}.\mathbf{b}_i \\ &> 0 \text{ if } \mathbf{w}.\mathbf{b}_i > 0\ \wedge\ \mathbf{v}.\mathbf{b}_i > 0.\end{aligned} \tag{8}$$

However, since the value of (1 + 1) mod 2 is 0, the vector sum of the encoded representations of $\mathbf{w}$ and $\mathbf{v}$ in the Hamming space produces an output of 0 in the i^{th} component.

4.2 Estimating Optimal Centroids

Given that modulo 2 addition in the Hamming space is problematic, there needs to be an alternate aggregation function to compute the centroid vector in the Hamming space. Moreover, due to the privacy preservation settings, it is not feasible to compute the centroid in the Euclidean space and then transform it to a point in the Hamming space. Therefore, under privacy preservation settings, the only way to compute the Hamming space centroid vectors would be to estimate these values probabilstically with incomplete information rather than computing them deterministically.

Considering the transformation function ϕ, this aggregate function equates to a sum of the signs of the projected values.

$$\begin{aligned} \mathbf{h}_i^k &= 1 \text{ if } \sum_{\mathbf{w} \in W^k} \operatorname{sgn}(\mathbf{w}.\mathbf{b}_i) \geq 0 \\ &= 0 \text{ otherwise} \end{aligned} \tag{9}$$

Although the vectors $\mathbf{w}$'s in Equation 9 are not known due to privacy constraints, the projected values themselves or the signs of these values may be considered to be made available to the server for the purpose of computation without posing a major security threat. Privacy in this case is preserved from the well-known property of locality preserving property of JL lemma that devising an inverse function of ϕ is computationally intractable [7, 8].

The intuition behind estimating the value at i^{th} signature bit of sum vector is that the sum of a large number of positive projected values with a relatively smaller number of negative values is likely to yield a positive result due to the outweighing effect.

In addition to the frequency of the positive projections, their average magnitude values and the skewness of these values can also affect the likelihood of the sum being positive. To model these factors formally, we make use of the Gaussian mixture model (GMM) to estimate the likelihood of the i^{th} bit of the sum vector to be 1.

5 CENTROID ESTIMATION BY GAUSSIANS

5.1 Global distribution of the projections

Let the set of projected values along the i^{th} basis vector be

$$\mathscr{B}_i = \bigcup_{\mathbf{w} \in W} \mathbf{w}.\mathbf{b}_i. \tag{10}$$

We split the set $\mathscr{B}_i$ in two parts according to whether the projection values are positive or negative and assume that the values in each set are generated by a normal distribution, i.e.,

$$\begin{aligned} &\mathscr{B}_i = \mathscr{P}_i \cup \mathscr{N}_i, \text{ such that} \\ &\mathscr{P}_i = \{\mathbf{w}.\mathbf{b}_i | \mathbf{w}.\mathbf{b}_i \geq 0\}, \ \mathbf{w}.\mathbf{b}_i \sim \mathcal{N}(\mu_i^+, \sigma_i^+) \\ &\mathscr{N}_i = \{\mathbf{w}.\mathbf{b}_i | \mathbf{w}.\mathbf{b}_i < 0\}, \ \mathbf{w}.\mathbf{b}_i \sim \mathcal{N}(\mu_i^-, \sigma_i^-), \end{aligned} \tag{11}$$

where μ_i^+ (μ_i^-) and σ_i^+ (σ_i^-) denote the mean and variance of the positive (negative) projections along the i^{th} basis vector respectively and $\mathcal{N}(\mu, \sigma)$ denotes the Normal distribution with mean μ and variance σ. The parameters of the normal distributions corresponding to each basis vector are computed from the observed projection values, e.g. μ_i^+ and σ_i^+ are computed from the $\mathscr{P}_i$ values.

5.2 Distribution of the sum

During each iteration, a privacy preserving K-means algorithm needs to assign the i^{th} component (bit) of the Hamming vector corresponding to the centroid (vector sum) of the k^{th} partition, $\mathbf{h}_i^k$, to the value of 1 or 0. This binary classification problem thus involves estimating the value of the sum of a set of projection variables (some positive and some negative). We assume that the positive and negative projections (encoded as 1's and 0's respectively) are drawn from two separate distributions. We are interested in the underlying distribution of the sum of these variables. In order to estimate the sum, we present the well known theorem (Theorem 5.1) that the sum of two normally distributed random variables is also normal.

Theorem 5.1. *If $Y_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $Y_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$ then the sum of these random variables $Y = Y_1 + Y_2 \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$.*

It is easy to prove Theorem 5.1 using the characteristic function of Normal distributions; for a proof the reader is referred to [21].

In the context of our problem, we assume that the sum of the projected values along i^{th} basis vector corresponding to an arbitrary partition is drawn from the sums of the $\mathscr{P}_i$ and the $\mathscr{N}_i$ values. This value, say x, according to Theorem 5.1, then follows the distribution

$$x \sim \mathcal{N}\Big(\mu_i^+ + \mu_i^-, (\sigma_i^+)^2 + (\sigma_i^-)^2\Big). \tag{12}$$

A value sampled from the distribution of Equation 12 is our best guess for the sum of an arbitrary number of reals representing the i^{th} component of the centroid in $\mathbb{H}^m$ belonging to a partition.

5.3 Estimating priors from a partition

Next, we need to classify the sampled value x into one of the classes (i.e. 1 or 0) for a current partition of the encoded vectors. We leverage the following two sources of information from the observed encoded vectors in each partition to estimate the likelihood of the i^{th} bit of the sum vector in each partition to be 1 (the likelihood of the bit to be set to 0 represents the complementary event).

1. **Hypothesis 1:** If the number of positive projections in a partition contributing to the sum (i.e. the number of vectors with the i^{th} bit observed to be 1) is considerably higher than the number of negative projections, then there is a considerable likelihood of the corresponding bit of the sum vector to be 1.
2. **Hypothesis 2:** If the average of positive projections (over the entire dataset) along the i^{th} basis vector is considerably higher than the average over the negative ones, then there is a strong likelihood of the i^{th} bit of the sum of the vectors in any partition to be 1.

Using the terminology that B_i^k refers to the set of observed signs of projected values (encoded bit representations), i.e.,

$$\begin{aligned} B_i^k &= \bigcup_{\mathbf{w} \in W_k} \operatorname{sgn}(\mathbf{w}.\mathbf{b}_i) = P_i^k \cup N_i^k \\ P_i^k &= \{\operatorname{sgn}(\mathbf{w}.\mathbf{b}_i) | \operatorname{sgn}(\mathbf{w}.\mathbf{b}_i) \geq 0\}, \mathbf{w} \in W_k \\ N_i^k &= \{\operatorname{sgn}(\mathbf{w}.\mathbf{b}_i) | \operatorname{sgn}(\mathbf{w}.\mathbf{b}_i) < 0\}, \mathbf{w} \in W_k, \end{aligned} \tag{13}$$

we estimate the prior probability of the positive class (probability of the i^{th} bit being set to 1) in the k^{th} partition as

$$Pr(\mathbf{h}_i^k = 1 | B_i^k) = \frac{|P_i^k|}{|B_i^k|}. \tag{14}$$

A problem with this maximum likelihood priors is that it does not take into account the relative magnitudes of the average values of the positive and negative projections. To this end, we need to address two events in the sampling process - the first of selecting a component (either positive or negative) by observing the respective counts in the partition, and the second, of sampling a value from that component. Stating this formally, the probability of the i^{th} centroid bit being set to 1 (the positive class) is given by

$$Pr(\mathbf{h}_i^k = 1|B_i^k) = \frac{|P_i^k|}{|B_i^k|}\mathcal{N}\Big(x|\mu_i^+, \sigma_i^+\Big), \tag{15}$$

where the variable x represents a sample drawn from Equation 12.

6 MULTI-COMPONENT GAUSSIANS

In Section 5 we described a Gaussian mixture model (GMM) with two components corresponding to the positive and negative projection values. In this section, we generalize the idea further by defining multiple components for the positive and negative projections.

Motivation. GMM with multiple components may model significant differences between the projection values of the same sign. With a binary GMM, the only parameter that can handle these differences is the variance parameter σ_i^+ (or σ_i^- for the negative projections). However, a multiple number of components, where each component generates projected values of the same sign (either positive or negative) within specific ranges, gives an estimate about the magnitude of the values, as opposed to estimating only their differences from the average (for the binary case). This estimate about the magnitude may potentially result in improving the estimate for the sign of $\mathbf{h}_i^k$, where the absolute value of a sum of a small number of projections could be higher than those of a much larger number of projections of the opposite sign.

Formal Description. To enable a more fine-grained approach to count the priors and the posteriors, we assume that the set of projected values follow a multi-component Gaussian mixture model, where values within a specific range are assumed to be generated from one particular component of the Gaussian mixture. In our approach, we divide the positive and the negative projected values into a number of (M a parameter) equal length intervals. More specifically, we store the global statistics of the projected values along each dimension i as

$$\begin{aligned}
&\mathscr{B}_i = (\cup_{j=1}^{M}\mathscr{P}_i^j) \cup (\cup_{j=1}^{M}\mathscr{N}_i^j), \text{ such that}\\
&\mathscr{P}_i^j = \{\mathbf{w}.\mathbf{b}_i | j\delta_i^+ \leq \mathbf{w}.\mathbf{b}_i < (j+1)\delta_i^+\}, \ \mathbf{w}.\mathbf{b}_i \sim \mathcal{N}(\mu_i^{j+}, \sigma_i^{j+})\\
&\mathscr{N}_i^j = \{\mathbf{w}.\mathbf{b}_i | j\delta_i^- \leq \mathbf{w}.\mathbf{b}_i < (j+1)\delta_i^-\}, \ \mathbf{w}.\mathbf{b}_i \sim \mathcal{N}(\mu_i^{j-}, \sigma_i^{j-}),
\end{aligned} \tag{16}$$

where each $\mathscr{P}_i^j$ ($\mathscr{N}_i^j$) represents a Gaussian generating positive (negative) projection values of the points within the j^{th} interval ($j = 1, \ldots, M$), μ_i^{j+} (μ_i^{j-}) and σ_i^{j+} (σ_i^{j-}), respectively, refer to the mean and the variance of the positive (negative) projected values within the j^{th} interval, and δ_i^+ (δ_i^-) represents the length of each positive (negative) intervals in i^{th} dimension, computed as

$$\delta_i^+ = \frac{(\mathbf{w}.\mathbf{b}_i)_{max} - (\mathbf{w}.\mathbf{b}_i)_{min}}{M}, \ \forall \mathbf{w}.\mathbf{b}_i \geq 0. \tag{17}$$

Similar to the binary case of Equation 12, to obtain the distribution of the sum, we sample a likely value of the projection of the sum vector from the distribution

$$x \sim \mathcal{N}\Big(\sum_{j=1}^{M}(\mu_i^{j+} + \mu_i^{j-}), \sum_{j=1}^{M}((\sigma_i^{j+})^2 + (\sigma_i^{j-})^2)\Big). \tag{18}$$

During clustering, let z denote the latent variable indicating the component from which the sum of the projection along the i^{th} dimension (denoted by x in Equation 18) is most likely to be sampled from. Using uniform priors, the maximum likelihood value of this latent variable is then estimated as ζ^+ when $x \geq 0$ and ζ^- otherwise. Mathematically,

$$\begin{aligned}
\zeta^+ &= \underset{j=1}{\overset{M}{\arg\max}}\, \mathcal{N}\Big(x|\mu_i^{j^+}, \sigma_i^{j^+}\Big), \text{ if } x \geq 0,\\
\zeta^- &= \underset{j=1}{\overset{M}{\arg\max}}\, \mathcal{N}\Big(x|\mu_i^{j^-}, \sigma_i^{j^-}\Big), \text{ if } x < 0,
\end{aligned} \tag{19}$$

That is, we use $\mathcal{N}(\mu_i^{j^+}, \sigma_i^{j^+})$'s as the posteriors when $x \geq 0$ and $\mathcal{N}(\mu_i^{j^-}, \sigma_i^{j^-})$'s otherwise. Next, after estimating the values of $z = \zeta^+$ (or ζ^-), we compute the likelihood of $\mathbf{h}_i^k$ by using the local priors (similar to Equation 15) with the help of Equation 20.

$$\begin{aligned}
Pr(\mathbf{h}_i^k = 1|B_i^k, z) &= \frac{|P_i^k|}{|B^k|}\mathcal{N}\Big(x|\mu_i^{\zeta^+}, \sigma_i^{\zeta^+}\Big), \text{ if } x \geq 0\\
Pr(\mathbf{h}_i^k = 0|B_i^k, z) &= \frac{|N_i^k|}{|B^k|}\mathcal{N}\Big(x|\mu_i^{\zeta^-}, \sigma_i^{\zeta^-}\Big), \text{ if } x < 0
\end{aligned} \tag{20}$$

where, P_i^k and B_i^k are defined as per Equation 13.

The multi-component case of Equation 20 is a generalization of the binary component case (Equation 15), the generalization ensuring that the posteriors are estimated over a small (and hence more reliable) range of values. It is to be noted that multiple components only apply to the posteriors and not to the local priors of each cluster which are still binary as per the definition of Equation 13.

Detailed working steps of client-side data encoding (including computing the global projection statistics) and the server side centroid estimation (mainly involving how to use the projection values for better estimation) are presented in Algorithms 1 and 2, respectively[1] .

7 EXPERIMENTAL SETUP

We conduct experiments to show the effectiveness of our proposed approach, which we call privacy preserving K-means (PPK-means). The objective of our experiments is to investigate: a) whether PPK-means with encoded data yields results that are comparable (not significantly different clustering results) in comparison to standard K-means, which has access to the true data; b) the best settings of PPK-means, in terms of mainly how many components to use in the GMM and the effects of priors and posteriors; and c) the run-time efficiency of PPK-means with respect to standard K-means.

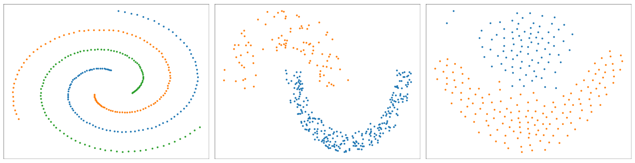
7.1 Dataset

7.1.1 Synthetic 2D Datasets. Since at each step of the PPK-means algorithm, we require to estimate the centroids from incomplete (encoded) information, it is useful to visually compare the estimated centroids at each iteration of the PPK-means algorithm with the true centroids obtained with standard K-means. For this purpose, we conduct experiments on a number of benchmark datasets in 2 dimensions [25]. Figure 1 plots the 3 datasets used in our experiments. The datasets exhibit a range of diversity in

[1] A prototype of the implementation of PPK-means is available for research purposes at https://github.com/gdebasis/superbit-kmeans.

Algorithm 1: Client: Hamming space Transformation

```
Input: $X = \{\mathbf{x}\}$: A collection of vectors, $\mathbf{x} \in \mathbb{R}^p$
Input: $m$: Hamming code length
Input: $M$: # GMM components, 0 for PPK-means with priors-only
Output: $\mu_i^{j+}$ ($\mu_i^{j-}$): Means of positive (negative) projections w.r.t. the $i^{\text{th}}$ basis
        vector ($i = 1, \ldots, m$) along the $j^{\text{th}}$ positive (negative) GMM
        component ($j = 1, \ldots, M$)
Output: $\sigma_i^{j+}$ ($\sigma_i^{j-}$): Variances of positive (negative) projections w.r.t. the
        $i^{\text{th}}$ basis vector ($i = 1, \ldots, m$) along the $j^{\text{th}}$ positive (negative)
        component ($j = 1, \ldots, M$)
Output: $X' = \{\mathbf{h} : \mathbf{h} \in \mathbb{H}^m\}$: A transformed set of Hamming vectors
begin
    Select basis vectors $\mathfrak{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$ by the Superbit Algorithm [19]
    // Send global statistics to clustering SaaS only if
       posteriors are to be used
    if $M > 0$ then
        $\delta_i^+ \leftarrow \Big(\max_{\mathbf{x} \in X : \mathbf{b}_i^T \cdot \mathbf{x} \geq 0} \mathbf{b}_i^T \cdot \mathbf{x} - \min_{\mathbf{x} \in X : \mathbf{b}_i^T \cdot \mathbf{x} \geq 0} \mathbf{b}_i^T \cdot \mathbf{x}\Big)/M$
        $\delta_i^- \leftarrow \Big(\max_{\mathbf{x} \in X : \mathbf{b}_i^T \cdot \mathbf{x} < 0} \mathbf{b}_i^T \cdot \mathbf{x} - \min_{\mathbf{x} \in X : \mathbf{b}_i^T \cdot \mathbf{x} < 0} \mathbf{b}_i^T \cdot \mathbf{x}\Big)/M$
        $(\mu_i^{j+}, \sigma_i^{j+}) \leftarrow (\mathbb{E}, \text{Var})_{\mathbf{x} \in X : j\delta_i^+ \leq \mathbf{b}_i^T \cdot \mathbf{x} < (j+1)\delta_i^+} \mathbf{b}_i^T \cdot \mathbf{x}$
        $(\mu_i^{j-}, \sigma_i^{j-}) \leftarrow (\mathbb{E}, \text{Var})_{\mathbf{x} \in X : j\delta_i^- \leq \mathbf{b}_i^T \cdot \mathbf{x} < (j+1)\delta_i^-} \mathbf{b}_i^T \cdot \mathbf{x}$
    for each $\mathbf{x} \in X$ do
        for $i = 1, \ldots, m$ do
            $\mathbf{h}_\text{i} \leftarrow sgn(\mathbf{b}_i^T \cdot \mathbf{x})$
        $X' \leftarrow X' \cup \mathbf{h}$
```



(a) Spiral [22] **(b) ΛV [23]** **(c) Flame [24]**

Figure 1: Visualization of the ground-truth clusters of the two dimensional datasets used in our experiments.

the number of visually perceived clusters, the convexity of these clusters and the connectivity between them, e.g., the dataset 'Spiral' (Figure 1a) represents 3 disconnected blocks of thin spirals, whereas the dataset in Figure 1b comprises two thick 'V' like shapes (one of them inverted), (the reason why we call it 'ΛV'). The plot in Figure 1c represents two clusters, one of them being convex (top).

7.1.2 Real datasets. In addition to using synthetic two dimensional datasets of a varying number of ground-truth clusters, we also test our clustering approach on two real-world datasets. The first dataset, named 'MNIST-8M', is comprised of 8.1M hand-written digits, each being a 784 dimensional feature vector (a gray-scale image with 28 × 28 pixels). The MNIST-8M is an extension of the original MNIST dataset of 70K images with addition of pseudo-random noise to the original MNIST images [26]. The number of components (or ground-truth clusters) of this dataset is 10, each corresponding to one of the digits, i.e. $\{0, \ldots, 9\}$.

Additionally, to evaluate our approaches on text data, we use the ODPtweets dataset[2] consisting of 8.3M tweets. Each tweet is labeled with the 'Open Directory Project' (ODP) category of the URL of the page which the tweet points to, total number of categories being 34185. On careful observation of the dataset, we found that there is a large number of ODP categories (specifically, 33770) with small number of candidates (specifically, $<$ 10), and that a number of classes (specifically, 12) have an excessively large number of tweets (specifically, 100K). For our experiments, we removed these head and tail categories, which resulted in a total of over 2.1M million tweets distributed among 403 ODP categories.

Algorithm 2: PPK-means on Clustering SaaS

```
Input: $X$: A transformed set of Hamming vectors ($\mathbb{H}^m$) received from a client
       as the output of Algorithm 1
Input: $K$: #desired clusters
Input: $M$: #GMM components, 0 for PPK-means with priors-only
Input: $\mu_i^{j+}$ ($\mu_i^{j-}$) and $\sigma_i^{j+}$ ($\sigma_i^{j-}$): Means and variances of positive (negative)
       projections w.r.t $j^{\text{th}}$ GMM component ($j = 1, \ldots, M$) along $i^{\text{th}}$ basis
       ($i = 1, \ldots, m$)
Input: $T$: maximum number of iterations
Output: A $K$-partition of $X$ such that $\bigcup_{k=1}^{K} X^k = X$
begin
    Randomly initialize $K$ cluster centres $\mathbf{h}^1, \ldots, \mathbf{h}^K \in X$
    for $t = 1, \ldots, T$ do
        // Assign every x to its nearest centroid
        foreach $\mathbf{x} \in X - \bigcup_{k=1}^{K} \{\mathbf{h}^k\}$ do
            $k' \leftarrow \arg\min_k (d_H(\mathbf{x}, \mathbf{h}^k))$ $X^{k'} \leftarrow X^{k'} \cup \mathbf{x}$
        for $k = 1, \ldots, K$ do
            // Recompute cluster center
            for $i = 1, \ldots, m$ do
                $PosCount \leftarrow 0$
                foreach $\mathbf{x} \in X^k$ do
                    if $\mathbf{x}_i = 1$ then
                        $PosCount \leftarrow PosCount + 1$
                $NegCount \leftarrow |X^k| - PosCount$
                if $M = 0$ then
                    if $rand(0, 1) \leq \frac{PosCount}{|X^k|}$ then $\mathbf{h}_i^k = 1$
                    else $\mathbf{h}_i^k = 0$
                else
                    $\alpha \leftarrow \mathcal{N}\Big(\sum_{j=1}^{M} (\mu_i^{j+} + \mu_i^{j-}), \sum_{j=1}^{M} ((\sigma_i^{j+})^2 + (\sigma_i^{j-})^2)\Big)$
                    if $\alpha > 0$ then
                        $\zeta^+ \leftarrow \arg\max_{j=1}^{M} \mathcal{N}(x|\mu_i^{j^+}, \sigma_i^{j^+})$
                        $S^+ \leftarrow \frac{PosCount}{|X^k|} \mathcal{N}(x|\mu_i^{\zeta^+}, \sigma_i^{\zeta^+})$
                        if $rand(0, 1) \leq S^+$ then $\mathbf{h}_i^k = 1$
                        else $\mathbf{h}_i^k = 0$
                    else
                        $\zeta^- \leftarrow \arg\max_{j=1}^{M} \mathcal{N}(x|\mu_i^{j^-}, \sigma_i^{j^-})$
                        $S^- \leftarrow \frac{NegCount}{|X^k|} \mathcal{N}(x|\mu_i^{\zeta^-}, \sigma_i^{\zeta^-})$
                        if $rand(0, 1) \leq S^-$ then $\mathbf{h}_i^k = 0$
                        else $\mathbf{h}_i^k = 1$
```

In order to obtain feature representations of each tweet, we trained word embedding employing 'skipgram' model of 'word2vec'[3] (with default parameters) over the tweet collection using 200 dimensions to represent each word. A dense vector representation of each tweet is then obtained by summing the word-embedded vectors of its constituent words.

7.2 Baselines

To test the effectiveness of estimating centroids with incomplete information (privacy preservation settings), we employ a number

[2] http://www.zubiaga.org/datasets/odptweets/

[3] https://github.com/tmikolov/word2vec

of baseline K-means clustering methods. Additionally, we also compare our results with the standard K-means, which works with the true data without the privacy preservation constraint. It is to be noted that since the standard K-means is not privacy preserving, instead of treating it as a baseline, it is rather treated as an *apex-line* to get an idea about the best results that could be obtained under ideal settings on a particular dataset.

7.2.1 LSH-based partitioning. Locality sensitive hashing (LSH) is a general class of data compression methods which seek to assign identical hash codes (called signatures) to vectors that are similar to each other. A commonly used LSH algorithm, called the MinHash, involves intersection of random permutations of the components in data [27]. The algorithm proposed in [8] extended MinHash based LSH to real-valued vectors in high dimensions by taking projections with respect to randomly chosen basis vectors. As our first baseline, we use the method proposed in [8] to partition the data into K classes. More specifically, for a given value of K, we compute the LSH signature of each data point ranging from 1 to K and then group together the data points by their binary encoded signature values. This ensures that similar points are clustered together (since they are expected to have similar signatures). In this algorithm, the K-means computation only needs to access the binary encoded signature values, as a result of which it is privacy preserving. We name this baseline approach 'LSH-partition'.

7.2.2 K-means over Hamming Space. To show the usefulness of centroid estimation, we employ the standard K-means approach that takes as inputs the encoded data, $\mathbf{x} = \phi(\mathbf{w})$, where $\phi : \mathbb{R}^p \mapsto \mathbb{H}^m$ is the superbit encoding function [19] (see Section 3.1). Different to our approach, we perform standard K-means clustering over the continuous space $\mathbb{R}^m$ (instead of considering only the discrete subspace $\mathbb{H}^m$), as a result of which the vector sum operation becomes a closed operation in $\mathbb{R}^m$. Since this baseline does not use a modified vector sum operation for computing the centroids (as PPK-means does with the GMM-based estimation), any errors in the encoding function are likely to propagate and potentially cause significant differences in results with respect to applying K-means on the original data. Note that we call this baseline 'HK-means' (K-means algorithm on a Hamming space).

7.2.3 K-means convergent on Hamming Space. Similar to the approach in Section 7.2.2, we execute K-means on the encoded set of binary vectors (signatures) in m dimensions. However, instead of treating the embedded space as the extended space of m dimensional real vectors, $\mathbb{R}^m$, we restrict the embedded space to the discrete space $\mathbb{H}^m$. Consequently, the standard notion of the vector sum operation involving component-wise addition is no longer a closed operation in $\mathbb{H}^m$, which requires redefining this operation to be able to execute K-means. Specifically, using the property that $\mathbb{R}^m$ is point-wise convergent, we compute the k^{th} cluster centroid as

$$\mathbf{h}_i^k = \text{sgn}\Big(\frac{1}{|X^k|}\sum_{\mathbf{x}\in X^k} x_i - \frac{1}{2}\Big), \ \mathbf{x}\in\mathbb{H}^m. \tag{21}$$

It can be easily verified that the centroid computation of Equation 21 is a closed operation, i.e. $\mathbf{h}^k \in \mathbb{H}^m$. Informally speaking, we first compute centroid vectors $\mathbf{h}^k$'s over the extended space $\mathbb{R}^m$, and then to maintain the closure property, we map a centroid $\mathbf{h}^k$ to its nearest point in the Hamming space (a similar approach was used in [28] to modify the skip-gram [29] objective function for obtaining binary embedding of graph nodes).

Since this baseline computes centroids using the Euclidean space and then 'truncates' these centroids to the nearest point in the Hamming space, we call this baseline 'E2HK-means' (Euclidean to Hamming convergent K-means).

7.3 Parameters and Evaluation Metrics

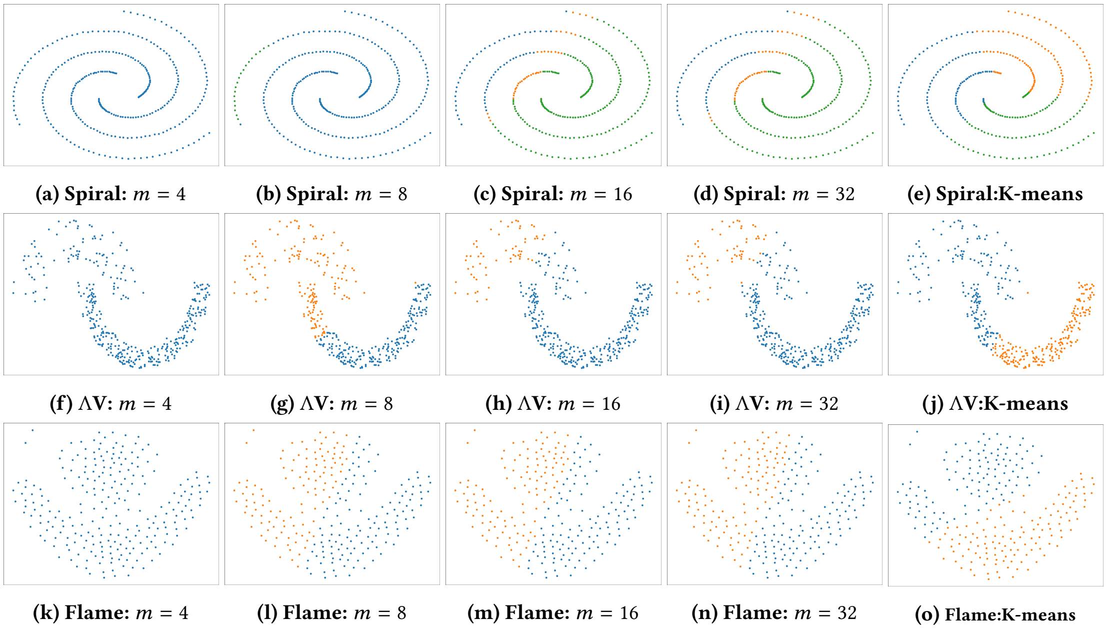
A parameter to PPK-means is the dimensionality of the Hamming space in which the p dimensional data needs to be transformed. Another parameter is the number of components, M, for the GMMs used to estimate the projected values of each sign (positive and negative) in PPK-means. A value of $M = 1$ corresponds to using a Normal distribution each for the positive and negative projections. We also investigate the use of posteriors in combination with the priors (Equation 15) vs. the use of priors only (Equation 14).

To measure the effectiveness of our proposed method, we use standard clustering metrics. Each dataset, that we experiment with, comprises the ground-truth information of class (cluster) labels. As an evaluation metric, we report Normalized Mutual Information (NMI), which measures how homogeneous the clusters are. A different type of clustering effectiveness measure is aggregation of classification results over pairs of data points, yielding higher values if a pair of data points from the same class (in the ground-truth) are predicted to belong to the same cluster. We thus also report F-score and adjusted rand index (ARI) aggregated over these pairwise grouping decisions. Additionally, we also measure the efficiency of the clustering methods in terms of computational latency. For a fair comparison of runtimes, all experiments were conducted on a 64-Bit Linux workstation with Intel Xeon 'E5-1620 3.60GHz' CPU and 48 GB RAM.

8 RESULTS

Visual comparison with K-means. To visually investigate the effectiveness of the centroid estimation process of PPK-means, we first report results of PPK-means on the three synthetic 2D datasets described in Section 7.1.1, and then compare these with the ideal scenario of standard K-means executed on the original (non-encoded) data. It is to be mentioned that we do not report results with the other baselines (outlined in Section 7.2) because our experiments with the MNIST-8M and ODPtweet dataset already revealed that these baselines resulted in worse clustering effectiveness (see Table 1). Further, we also report clustering results for PPK-means with priors-only configuration, because after visual inspection, we noticed that these results were indistinguishable from the ones that used the posteriors.

Figure 2 shows the partitions obtained during intermediate steps of executing PPK-means. An interesting observation is that for PPK-means to work well, the dimension of the Hamming space needs to be sufficiently larger than p (the dimension of the original data points). As an extreme case, it can be seen that most points are clustered into a single group with the configuration $m = 4$ on all the datasets. The results improve with $m = 8$ and higher. For the 'Spiral' dataset, $m = 8$ is not able to find out the 3 natural clusters (it finds only 2). It can also be seen that the results with $m = 16$


(a) Spiral: $m = 4$ (b) Spiral: $m = 8$ (c) Spiral: $m = 16$ (d) Spiral: $m = 32$ (e) Spiral:K-means

(f) ΛV: $m = 4$ (g) ΛV: $m = 8$ (h) ΛV: $m = 16$ (i) ΛV: $m = 32$ (j) ΛV:K-means

(k) Flame: $m = 4$ (l) Flame: $m = 8$ (m) Flame: $m = 16$ (n) Flame: $m = 32$ (o) Flame:K-means

Figure 2: Comparison of PPK-means after 5 iterations (on a variable number of encoding dimensions, m) shown in Figure 2a, 2f and 2k. The rightmost plot in each row (Figure 2e, 2j and 2o) plots standard K-means results on non-encoded data. The parameter K for both PPK-means and K-means were set to the number of true clusters (as per the ground-truth). The PPK-means version used for obtaining the plots only involved the priors only (Section 5.3).

and $m = 32$ are comparable with those of K-means. This is an important observation which shows that PPK-means, even without the complete knowledge of data, can yield comparable results with those of K-means. This shows that the PPK-means can potentially work well as a privacy preserving K-means algorithm.

Table 1 shows the results of comparing the performance of two different settings of PPK-means (with and without posteriors) with the three baseline algorithms (as presented in Section 7.2) on the MNIST-8M dataset. Firstly, we observe that the LSH-based partition yields poor results in terms of F-score, ARI, and NMI, which shows that it tends to group dissimilar feature instances into the same group, i.e., it classifies most of the digits into a small number of clusters thus resulting in largely non-homogeneous clusters. Multi-component based PPK-means outperforms the other baselines (including the single component K-means), indicating the usefulness of the posteriors and a multiple number of Gaussians to better estimate the centroids of each cluster. In fact, the performance of the proposed multi component PPK-means is seen to be comparable with the standard K-means for the MNIST-8M dataset.

It is worth noting that the execution times of all variants of the proposed method are significantly smaller than that of the standard K-means algorithm. An important implication is that the proposed method (specifically, PPK-means with multi component) achieves comparable performance as standard K-means with significantly smaller execution time.

Similar trends are also observed for the text dataset. In particular, from Table 1, it can be seen that the clustering results of the proposed GMM based methods that make use of the posterior information are significantly better than the baselines (and also standard K-means). While the performance of the proposed methods are comparable to that of HK-means and E2HK-means for some metrics, the optimal performance is observed for the multi-component GMM based PPK-means on both F-score and ARI.

The fact that the effectiveness of PPK-means without using the posteriors and the multiple components is lower (in comparison to the case where we use this information) indicates that

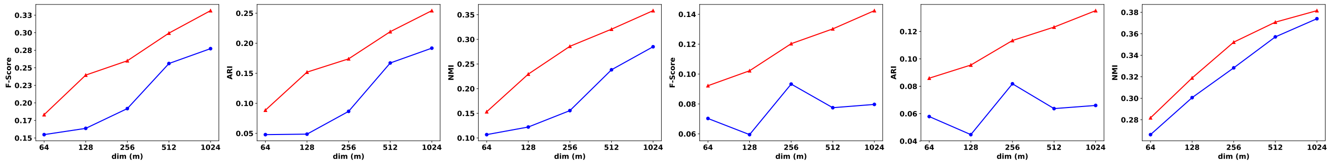
(1) Only using the priors in PPK-means may not be able to capture the situation when a small number of positive projected values can dominate the overall sum involving a larger number of negative values or vice-versa.
(2) Using a single Normal distribution to model the projected values of a particular sign may not be expressive enough to capture the variations in the projected values themselves.

The above limitations of the priors-only based and the single component GMM based PPK-means are addressed by a) using the posterior information (in the form of global statistics of the averages and the variances of the projected values), and b) by employing a multiple number of intervals to generalize the $M = 1$ case. With $M = 10$ (the best we achieved by varying M within a range of 2 to 10), the PPK-means algorithm is able to better estimate the centroid vectors by using a more fine-grained approach leveraging the additional information about the different ranges of the projected values. Consequently, the estimated centroid vectors are more similar to their true counterparts (i.e. the ones obtained with K-means on non-encoded data and then transformed to the Hamming space).

With respect to run-times, it can be observed that making use of posteriors and multiple components can lead to increase in runtimes as opposed to the single component priors-only case. This

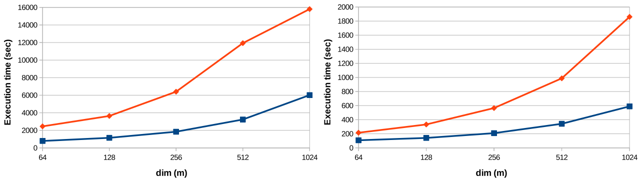
Table 1: Comparison of PPK-means against baseline clustering approaches on MNIST-8M ($K = 10$) and ODPtweets ($K = 403$) dataset. The value of K (#desired clusters) was set to same as #reference clusters. #iterations for each method was set to 10.

Method	Centroid	$\phi : \mathbb{R}^p \mapsto \mathbb{H}^m$	Privacy	MNIST-8M				ODPtweets			
Name	Estimation	($m =$)	preserve	F-score	ARI	NMI	Time (s)	F-score	ARI	NMI	Time (s)
LSH-partition [8]	None	1024	True	0.1871	0.0460	0.0817	6664	0.0236	0.0037	0.0936	512
HK-means	$\sum \mathbb{R}^m$ (centroids $\in \mathbb{R}^m$)	1024	True	0.2967	0.2143	0.3012	18782	0.1311	0.1261	0.3790	7492
E2HK-means	$\lim(\sum \mathbb{R}^m) \mapsto \mathbb{H}^m$	1024	True	0.3015	0.2196	0.3307	10669	0.1205	0.1161	**0.3833**	1580
PPK-means	GMM priors only	1024	True	0.2773	0.1918	0.2850	6013	0.0797	0.0659	0.3740	625
PPK-means ($M = 1$)	Single-component GMM	1024	True	0.2812	0.1981	0.2851	13196	0.1200	0.1125	0.3758	1820
PPK-means ($M = 10$)	Multi-component GMM	1024	True	**0.3314**	**0.2542**	**0.3582**	15807	**0.1423**	**0.1351**	0.3815	1860
K-means	$\sum \mathbb{R}^p$	N/A	False	0.3573	0.2852	0.3951	190138	0.1078	0.1015	0.3610	2057



(a) MNIST-8M:F-score **(b) MNIST-8M:ARI** **(c) MNIST-8M:NMI** **(d) ODP-Tweet:F-score** **(e) ODP-Tweet:ARI** **(f) ODP-Tweet:NMI**

Figure 3: Sensitivity of PPK-means with priors-only (Blue), PPK-means with multi-component GMM ($M = 10$) (Red) against variations in the encoding dimensionality.



(a) MNIST-8M **(b) ODPtweets**

Figure 4: Variations in execution time for PPK-means with priors-only (Blue) and PPK-means with multi-component GMM ($M = 10$) (Red).

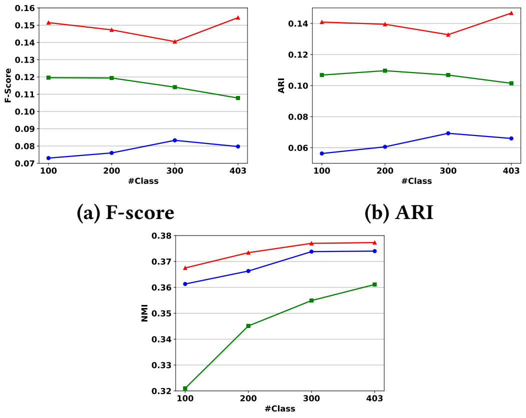
increase in time can be attributed to the computation and transportation (from a client to the cluster SaaS) of more information, namely the mean and the variances of the projected values. Increasing the number of GMM components (M) also leads to increasing the run-time since the computation then involves estimating the likelihood of the sign of each component of a centroid vector from a multiple number (M) of components.

Another interesting observation about the run-times is that all the baseline approaches and PPK-means execute much faster than the standard K-means without the privacy constraint. This is because the encoded data is stored as integers of 8 bytes (a word size) in the main memory which is much smaller than storing real-valued vectors (e.g. storing 1024 dimensional Hamming vectors requires 1024/64=16 words of memory, whereas storing a 784 dimensional real-valued vector consumes 784 words of memory). Moreover, encoding vectors as integers also leads to much faster inner product based similarity computation between them in comparison to the computationally expensive floating point operations of real-valued vectors, e.g., to compute the similarity between two 1024 dimensional Hamming vectors, one simply needs to execute the `POPCOUNT` machine instruction 16 times (16=1024/64).

Parameter Sensitivity. We now investigate the effects of varying the encoding dimension (m) and the number of components for GMM estimation (M) in PPK-means. Figure 3 shows the relative differences between PPK-means (priors only) and PPK-means with GMM posteriors ($M = 10$) for different values of m within 64 to 1024, each value of m being a multiple of 16 (CPU word size). From the figure, it can be seen that the downstream clustering effectiveness is proportional to the value of m implying that low dimensional Hamming representations tend to lose information about the original data vectors. It is interesting to see that even with noisy representation of the encoded vectors, GMM posterior-based PPK-means shows significant differences in results as compared to its prior-only counterpart (see the relatively large differences of F-score, ARI and NMI values). This suggests that the posterior based PPK-means is more robust under parsimonious settings of memory and CPU usage. Figure 4 shows that the execution time increases drastically with larger values of m (which was one of the reasons why the value of m was restricted up to 1024 in our experiments).

As the MNIST-8M dataset contains images of the 10 digits (0 to 9), the number of desired clusters (K) is 10. However in many practical scenarios, the ideal number of clusters is not known apriori. To test the robustness of proposed methods without the information about the true number of clusters, we evaluate the clustering effectiveness of the competing methods by varying the value of K. From Figure 5, we observe that PPK-means (multi-component GMM) outperforms both standard K-means and the version of PPK-means that uses priors only. In Figure 6, we investigate the effect of varying the number of GMM components in PPK-means. It can be seen that increasing M tends to increase clustering effectiveness.

Figure 7 plots an intrinsic clustering evaluation metric, namely the residual sum of squares (RSS), which is measured by first aggregating the distances of the constituent points of a cluster from its centroid and then averaging these values over all clusters in the dataset. The smaller the RSS value, the better is the clustering output. Figure 7 shows that the multi-component ($M = 10$) GMM

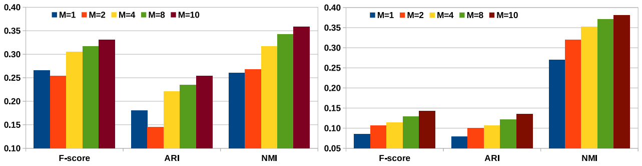


(a) F-score

(b) ARI

(c) NMI

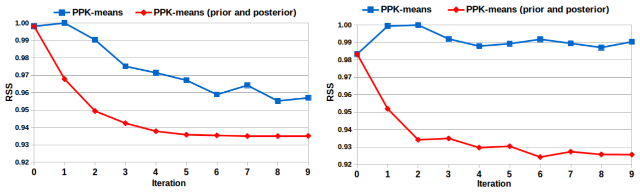
Figure 5: Sensitivity of PPK-means with priors-only (Blue), PPK-means with multi-component GMM ($M = 10$) (Red) and K-means (Green) with K (#desired clusters) on ODPtweets.



(a) MNIST-8M

(b) ODPtweets

Figure 6: Sensitivity of PPK-means with multi-component GMM for varying M (#components).



(a) MNIST-8M

(b) ODPtweets

Figure 7: Variations in RSS values for PPK-means with priors-only and PPK-means ($M = 10$).

setting of PPK-means leads to sharper drops in normalized RSS values across iterations than its priors-only counterpart. The priors only mode of PPK-means can sometimes also lead to increasing the RSS value across iterations (see the increase from iteration 6 to 7), which can happen due to the uncertainties involved in centroid estimation. However, the fact that the RSS values steadily decrease for the posterior mode of PPK-means, shows that the centroid estimations in this case are more robust.

9 CONCLUSIONS

We investigated the problem of K-means clustering under a privacy preservation constraint. This constraint requires the input data to be sent in an encoded format to a server offering clustering as a 'software as a service' (SaaS), such that the data is protected from any information leaking threats (e.g. deanonymization and authorship attribution). We propose a modified K-means algorithm, called PPK-means, that leverages additional pieces of information, e.g. global statistics on the projected values of the original data vectors along random basis vectors used for the purpose of encoding. Experimentation on image and textual data demonstrate that the proposed approach, by leveraging information in addition to the encoded data itself, is better able to estimate the centroids during K-means iterations eventually leading to better clustering effectiveness in comparison to a range of baseline approaches for privacy preserving clustering. Further, the proposed PPK-means method (multi-component variant) is less computationally expensive than the standard K-means method. It was observed that text data, the proposed method outperforms the standard K-means.

In future, we would like to address privacy preservation constraints for other clustering methods, e.g. DBSCAN, and also formalize the notions of differential privacy under such a setup.

REFERENCES

[1] B. Michael and Z. Tom. A Face is exposed for AOL searcher no. 4417749. *New York Times*, page A1, 08 2006.
[2] W. Benjamin et al. Syntf: Synthetic and differentially private term frequency vectors for privacy-preserving text mining. In *SIGIR '18*, pages 305–314, 2018.
[3] J. Marek, J. Martin, and R. Konrad. Smart metering de-pseudonymization. In *Proc. of ACSAC '11*, pages 227–236. ACM, 2011.
[4] Ricardo Mendes and João P Vilela. Privacy-preserving data mining: methods, metrics, and applications. *IEEE Access*, 5:10562–10582, 2017.
[5] D. Irit and N. Kobbi. Revealing information while preserving privacy. In *Proc. of Symposium on Principles of Database Systems*, pages 202–210. ACM, 2003.
[6] V. S Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *Sigmod Record*, 33:50–57, 2004.
[7] J. William and L. Joram. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability*, volume 26 of *Contemporary Mathematics*, pages 189–206. 1984.
[8] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.
[9] S. Xiao-Bo, L. Weiwei, T Ivor W, S. Fumin, and S. Quan-Sen. Compressed k-means for large-scale clustering. In *AAAI*, pages 2527–2533, 2017.
[10] K. Siddhesh and A. Amit. Faster k-means cluster estimation. In *European Conference on Information Retrieval*, pages 520–526. Springer, 2017.
[11] Y. Jinfeng, W. Jun, and J. Rong. Privacy and regression model preserved learning. In *Proc. of AAAI '14*, pages 1341–1347, 2014.
[12] J. Geetha and W. Rebecca N. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proc. of KDD'05*, pages 593–599. ACM, 2005.
[13] M. Noman, C. Rui, F. Benjamin, and Philip S Y. Differentially private data release for data mining. In *Proc. of KDD'11*, pages 493–501, 2011.
[14] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. *Journal of Privacy and Confidentiality*, 7:17–51, 2017.
[15] H. Kashima, J. Hu, B. Ray, and M. Singh. K-means clustering of proportional data using l1 distance. In *Proc. of PR '08*, pages 1–4, 2008.
[16] S. David. Web-scale k-means clustering. In *WWW'10*, pages 1177–1178, 2010.
[17] J. Herve, D. Matthijs, and S. Cordelia. Product quantization for nearest neighbor search. *IEEE Trans. PAMI*, 33(1):117–128, 2011.
[18] M. Yusuke et al. PQk-means: Billion-scale clustering for product-quantized codes. In *Multimedia Conference*, pages 1725–1733, 2017.
[19] J. Ji, J. Li, S. Yan, B., and Q. Tian. Super-bit locality-sensitive hashing. In *Proc. of NIPS'12*, pages 108–116, 2012.
[20] X. Yi, C. Caramanis, and E. Price. Binary embedding: Fundamental limits and fast algorithm. In *Proc. of ICML'15*, pages 2162–2170, 2015.
[21] B. Eisenberg and R. Sullivan. Why is the sum of independent normal random variables normal. In *Math. Mag.*, volume 81, pages 362–366, 2008.
[22] H. Chang et al. Robust path-based spectral clustering. *PR*, 41:191–203, 2008.
[23] J. Anil K et al. Data clustering: A user's dilemma. In *PReMI*, pages 1–10, 2005.
[24] F. Limin and M. Enzo. Flame, a novel fuzzy clustering method for the analysis of dna microarray data. *BMC Bioinformatics*, 8(1), Jan 2007.
[25] F. Pasi and S. Sami. K-means properties on six clustering benchmark datasets. *Applied Intelligence*, 48(12):4743–4759, Dec 2018.
[26] K. Krauth, E. V. Bonilla, K. Cutajar, and M. Filippone. Autogp: Exploring the capabilities and limitations of gaussian process models. In *Proc. of UAI'17*, 2017.
[27] C. Moses S. Similarity estimation techniques from rounding algorithms. In *Proc. of STOC '02*, pages 380–388. ACM, 2002.
[28] V. Misra and S. Bhatia. Bernoulli embeddings for graphs. In *AAAI'18*, 2018.
[29] T. Mikolov et al. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS'13*, pages 3111–3119, 2013.