2011 11th IEEE International Conference on Data Mining Workshops

Privacy Preserving Outlier Detection using Locality Sensitive Hashing

Nisarg Raval, Madhuchand Rushi Pillutla, Piysuh Bansal, Kannan Srinathan, C. V. Jawahar
International Institute of Information Technology Hyderabad, India
{*nisarg.raval@, rushi.pillutla@, piyush_bansal@*}*research.iiit.ac.in*
{*srinathan@, jawahar@*}*iiit.ac.in*

***Abstract*—In this paper, we give approximate algorithms for privacy preserving distance based outlier detection for both horizontal and vertical distributions, which scale well to large datasets of high dimensionality in comparison with the existing techniques. In order to achieve efficient private algorithms, we introduce an approximate outlier detection scheme for the centralized setting which is based on the idea of Locality Sensitive Hashing. We also give theoretical and empirical bounds on the level of approximation of the proposed algorithms.**

***Keywords*-privacy; outlier detection; LSH**

I. Introduction

Data Mining, which is the process of extracting patterns from large datasets, has become an important research area due to the exponential growth of digital data and the storage capabilities. However, in case of large datasets collected from various input sources, oftentimes the data is distributed across the network, rising concerns for privacy and security while performing distributed data mining. To overcome this problem, Privacy Preserving Data Mining (PPDM) methods have been proposed, which are based on Data randomization techniques and Cryptographic techniques. PPDM methods based on the latter were introduced by Lindell and Pinkas in [7]. In that paper, an algorithm for Privacy preserving ID3 Classification was described. Subsequently, privacy preserving algorithms have been proposed for various data mining tasks such as association rule mining, classification, clustering and outlier detection.

Privacy preserving outlier detection (PPOD) was introduced by Vaidya *et al.* in [11]. They use the definition for distance based outliers provided in [6], and give PPOD algorithms for both horizontal and vertical partitioning of data. Subsequently, a PPOD algorithm using the k-nearest neighbor based definition [9] was given in [12], considering only vertical partitioning. However, all of the above mentioned algorithms have quadratic communication and computation complexities in the database size, making them infeasible while dealing with large datasets. To the best of our knowledge, no other work in the field of PPDM based on cryptographic techniques has addressed distance based outlier detection. Privacy preserving density based outlier detection algorithms have been proposed in [2], [10].

In this paper, we propose approximate PPOD algorithms for both horizontal and vertical partitioning of data. As opposed to the current PPOD algorithms which provide privacy for already existing outlier detection algorithms, we develop a new outlier detection scheme for the centralized setting in order to achieve efficient algorithms in private settings. The centralized scheme is based on our previous work on approximate outlier detection [8] which uses Locality Sensitive Hashing (LSH) technique [5]. We also give theoretical bounds on the level of approximation and provide the corresponding empirical evidence.

The computational complexity of our centralized algorithm is $O(ndL)$ for d-dimensional dataset with n objects. The parameter L is defined as $n^{1/1+\epsilon}$, where $\epsilon > 0$ is an approximation factor. The computational complexity of our PPOD algorithms in both horizontally and vertically distributed settings is same as that of the centralized algorithm, which is a considerable improvement over the previous known result of $O(n^2d)$. The communication complexity in vertically distributed setting is $O(nL)$ and in horizontally distributed setting it is $O(N_bL\log n)$; where $N_b << n$, is the average number of bins created during each of the L iterations of LSH. Thus in both cases, we show a significant improvement over the existing communication complexity, which is quadratic in dataset size. Further, the communication cost of our privacy preserving algorithm in horizontal distribution is independent of data dimensionality and hence works very efficiently even for datasets of very large dimensionality as opposed to the existing algorithms. However, we achieve the above mentioned improvements at the cost of an approximate solution to outlier detection.

II. Overview and Background

Our outlier detection scheme uses the definition for a distance based outlier proposed by Knorr *et al.* [6].

Definition 1. *$DB(p_t, d_t)$ outlier: An object o in a dataset D is a $DB(p_t, d_t)$ outlier if at least fraction p_t of the objects in D lie at a distance greater than d_t from o.*

In our approach, we use the converse of this definition and consider an object to be a non-outlier if it has enough neighbors (p'_t) within distance d_t, where $p'_t = (1-p_t)\times|D|$. Since the fraction p_t is very high (usually set to 0.9988), the modified point threshold p'_t will be very less compared to the number of objects in D. This allows us to easily detect

978-0-7695-4409-0/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDMW.2011.141

IEEE computer society

most of the non-outliers by finding only p'_t objects within distance d_t.

To efficiently find the near neighbors, we use the Locality Sensitive hashing technique. Given a set of objects, the LSH scheme hashes all the objects in such a way that all those objects within a specified distance are hashed to the same value with a very high probability. This way, all those non-outliers in the data which have many near neighbors can be identified easily, without calculating the distances to every other object in the dataset. Moreover, using LSH properties, whenever we identify a non-outlier we will be able to say most of its neighbors as non-outliers without even considering them separately. Thus, we obtain a very efficient pruning technique where, most of the non-outliers in the dataset can be easily pruned. The remaining points after pruning are the set of probable outliers which will contain very few non-outliers. To further remove these non-outliers, we use the probabilistic nature of LSH. The idea is to take the intersection of the sets of probable outliers over multiple runs, to output the final set of approximate outliers. The approach works because each set of probable outliers will contain the actual outliers with extremely high probability.

In our privacy preserving protocol for horizontally distributed data, each player locally computes its own set of probable outliers using the centralized algorithm with global distance and point threshold parameters. In the next phase, all the players communicate to obtain their subset of the actual outliers in the total database. In case of vertically distributed data, all players first communicate to obtain the LSH binning of all the objects considering all the attributes. Next, the players locally compute the probable outliers using the global LSH binning information. We consider Honest-But-Curious (HBC) adversary model in our privacy preserving algorithms [4].

A. Locality Sensitive Hashing

The idea of Locality Sensitive Hashing was first introduced in [5]. The basic concept of LSH is to hash all the objects such that similar objects are hashed to the same bin with high probability. Mathematically, this idea is formalized as follows:

Definition 2. *A family $H \equiv h : S \rightarrow U$ is called (r_1, r_2, p_1, p_2)-sensitive if for any two objects p, q in S:*

$$if\ d(p,q) \leq r_1 : Pr[h(p) = h(q)] \geq p_1 \quad (1)$$

$$if\ d(p,q) \geq r_2 : Pr[h(p) = h(q)] \leq p_2 \quad (2)$$

where $d(p,q)$ is the distance between objects p and q.

For the hashing scheme to be locality sensitive, two conditions to be satisfied are $r_2 > r_1$ and $p_2 < p_1$; $(1+\epsilon) = r_2/r_1$, is the approximation factor. In order to amplify the gap between the probabilities p_1 and p_2, standard practice is to concatenate several hash functions to obtain a hash family $G = \{g : S \rightarrow U^k\}$ such that $g(p) = (h_1(p), h_2(p), ..., h_k(p))$; where k is the width of each hash function and $h_i \in H$. For a hash function family G, the probabilities in Equation 1 and 2 are modified as:

$$if\ d(p,q) \leq r_1 : Pr[g(p) = g(q)] \geq p_1^k \quad (3)$$

$$if\ d(p,q) \geq r_2 : Pr[g(p) = g(q)] \leq p_2^k \quad (4)$$

During LSH, each object $o \in D$ is stored in the bins $g_j(o)$ for $j = 1, 2...L$; where each g is drawn independently and uniformly at random from G i.e., each object is hashed using L hash functions drawn from G and stored in the corresponding bins. The optimal values for the parameters k and L are computed as [5]: $k = \log_{1/p_2} n$ and $L = n^\rho$ where $\rho = \frac{\ln(1/p_1)}{\ln(1/p_2)}$. In our algorithms, we use the LSH scheme based on p-stable distributions introduced in [3].

III. Algorithms

A. Centralized Setting

The algorithm for centralized setting is executed in two phases. In the first phase, the LSH scheme is used to hash all the objects to generate the binning structure. In the second phase, this LSH binning is used to compute the set of approximate outliers.

The algorithm *CentralizedOD* takes as input the dataset D, distance threshold d_t and point threshold p_t and outputs the set of approximate outliers M. In the First phase, initially the modified point threshold p'_t and the LSH distance parameter R are computed as: $p'_t = (1 - p_t) \times |D|$ and $R = r_1 = d_t/c$, where $c = (1+\epsilon)$. The LSH scheme is then run on the dataset D with the parameter R. In the LSH scheme each object is hashed using L hash functions each of width k, as explained in Section II-A. The output is a binning structure T with each bin having the objects hashed to that bin.

Algorithm 1 CentralizedOD

Input: Dataset D, Distance Threshold d_t, Point Threshold p_t
Output: Outliers M

1: $p'_t = (1 - p_t) \times |D|$
2: $R = d_t/c$
3: $T = LSH(D, R)$
4: Compute b_t // Compute the optimal bin threshold using Equation 8
5: $M = Pruning(D, T, p'_t, b_t)$
6: Return M

In the second phase *Pruning*, most of the objects which cannot be outliers are pruned. Initially all the objects in the dataset are marked as *not pruned*. For pruning, each object o is considered and is processed only if it has not been marked as *pruned*, in which case we find the number of neighbors o has within distance d_t. To find the neighbors, the L bins

to which o is hashed during LSH are considered and the set of all objects stored in those L bins is formed (without removing duplicates). We denote this set by $Neighbors$. The objects in this set are the probable neighbors of o. More precisely, from Equation 4, we know that each object in the set $Neighbors$ is within the distance $r_2 = (1+\epsilon) \times r_1 = d_t$ from o, with a probability at least $(1 - p_2^k)$. To boost this probability, we consider only those objects which are repeated more than b_t ($b_t \leq L$) times in the L bins and store only those objects in a new set $RepeatedNeighbors$. In other words, we are reducing the error probability of considering a non-neighbor as a neighbor of the object o. Here, b_t is a *bin threshold* which can be computed based on the desired false negative probability. We propose a method to compute the optimal value of b_t later in this section.

Algorithm 2 Pruning

Input: Dataset D, Hash Table T, Point Threshold p'_t, Bin Threshold b_t,
Output: M

```
M = {}
∀o ∈ D, pruned[o] = false
for each object o in D do
    if pruned[o] = false then
        Neighbors = ⋃_{i=1}^{L} T[g_i(o)] // g is the hash function
        RepeatedNeighbors = {o' | o' ∈ Neighbors and occurrence(o') ≥ b_t}
        if |RepeatedNeighbors| > p'_t then
            ∀o' ∈ RepeatedNeighbors, pruned[o'] = true
        else
            M = M ∪ {o}
        end if
    end if
end for
```

If the cardinality of the set $RepeatedNeighbors$ is greater than the modified point threshold p'_t, with a very high probability o cannot be an outlier since it has sufficient neighbors within distance d_t. Moreover, this holds true for all the objects in $RepeatedNeighbors$ because from Equation 4, any two objects in this set are within distance d_t, i.e., every object other than o also has more than p'_t neighbors with in the distance d_t, so it can not be an outlier (with very high probability). Hence, all objects in $RepeatedNeighbors$ are marked as *pruned* (non-outlier). If, on the other hand the cardinality of $RepeatedNeighbors$ is less than or equal to p'_t, we consider the object o as a probable outlier and add it to the set of probable outliers M. This procedure is repeated till all the objects are either marked as *pruned* or as outliers. Finally, the set M of the probable outliers is returned. We denote the number of objects actually processed during *Pruning* as N_{pr} and later in sub-section III-A3, we give a bound for N_{pr}.

The set M contains the actual outliers as well as a few false positives and extremely low false negatives. In the following sub-section we give a theoretical bound on the number of false positives and false negatives and in sub-section III-A2 we give a method to reduce the false positives.

1) False Positives and False Negatives: In the context of outlier detection, a false positive (fp) is to label a non-outlier as an outlier and false negative (fn) is to label an outlier to be a non-outlier. From [8], we know that the false positive and false negative probabilities are:

$$Pr[fp] < b_t \times \binom{L}{L-b_t}(1 - p_1^{k(L-b_t)})p_1^{kb_t} \quad (5)$$

$$Pr[fn] < (L - b_t) \times \binom{L}{b_t} p_2^{kb_t}(1 - p_2^{kb_t})^{L-b_t} \quad (6)$$

2) Reducing False Positives: The technique to reduce the false positives is based on the probabilistic nature of the LSH scheme. The LSH scheme based on p-stable distribution projects all objects onto random lines. Due to this randomization, each execution of the LSH scheme projects the objects onto different lines which in turn ensures that the output of the centralized algorithm is probabilistic. However, choosing an optimal bin threshold ensures that the actual outliers in the dataset are returned in the output of the centralized algorithm even over multiple runs. Hence, to further reduce the false positives, we run the centralized algorithm over a fixed number of iterations $iter$, and take the intersection of the resulting sets and return this intersection set S as the final output. i.e., $S = \bigcap_{i=1}^{iter} M_i$. Since for a false positive to occur in the set S it should occur in each of the sets M_i, we have the modified false positive and false negative probabilities as:

$$Pr'[fp] = (Pr[fp])^{iter} \quad (7)$$

$$Pr'[fn] = 1 - (1 - Pr[fn])^{iter} \quad (8)$$

As can be seen from the above formulas, the modified probability of a false positive decreases exponentially with the number of iterations and thus we need to run only few iterations to achieve very few false positives. We discuss more about the effect of $iter$ on the false positives and false negatives in the experiments section. Furthermore, the false positives which remain in the final output can be termed as *weak* non-outliers, in the sense that most of these objects have marginally greater number of neighbors than the required threshold of objects to make an object a non-outlier. We support this claim by giving empirical results in the experiments section.

An optimal value for b_t would be one which would remove the false negatives at the cost of introducing minimal false positives. In our scheme, the user has the flexibility to fix b_t, based on the false negative probability desired, using Equation 8.

3) Bound on Number of Processed Objects N_{pr}: We give an upper bound on the number of objects which are processed during *Pruning*. Let the actual number of outliers in the dataset be m. Consider the worst case scenario where all the actual non-outliers in the dataset have exactly p'_t number of objects as neighbors. The number of objects processed, N_{pr}, would be maximum in the above mentioned case for which the bound is given as:

$$N_{pr} = (\frac{n-m-fp}{p'_t} + m + fp)$$

where $n = |D|$ and fp is the number of false positives. Usually, in a $DB(p_t, d_t)$ outlier detection scheme the fraction p_t is set to 0.9988, in which case p'_t will be: $p'_t = (1-p_t)n = .0012n$. Since $n >> m$ and $n >> fp$ we can approximate $(n-m-fp)$ to n and hence, the bound for number of objects processed is: $N_{pr} < ((n/.0012n) + m + fp) < m + fp + 834$. This is the worst case bound and usually the actual number of objects processed will be much less than the above mentioned inequality. Later in the results section, we show that for large datasets, N_{pr} is in fact less than 1% of the size of the total dataset.

B. Horizontal Distribution

In case of Horizontal partitioning, each player has the same attributes for a subset of the total objects. The algorithm for privacy preserving outlier detection over horizontally distributed data is executed in two phases. In the First phase, each player locally computes its set of local probable outliers, by running *CentralizedOD* on its local dataset. These local outliers contain the global outliers as well as a few non-outliers for which enough neighbors do not exist in the respective local datasets but exist in the entire dataset considering all the players. To prune these non-outliers, all the players engage in communication in the Second phase in order to generate the global neighbor information and compute their subsets of the global probable outliers. We define the local and global outliers as follows.

Definition 3. local outlier*: given a distance threshold d_t and a point threshold p_t, an object o with player P_i is a local outlier if the number of objects in the local dataset D_i lying at a distance greater than d_t is at least a fraction p_t of the total dataset D.*

Definition 4. global outlier*: given a distance threshold d_t and a point threshold p_t, an object o in a dataset D is a global outlier if at least fraction p_t of the objects in D lie at a distance greater than D from o.*

We consider t players, each with dataset D_i for $i = 1$ to t and $n_i = |D_i|$ such that the size of the entire dataset D is $n = \sum_{i=1}^{t} n_i$. We assume that n is known beforehand to all the players, otherwise it could be computed using a *SecureSum* protocol. Apart from this, the algorithm takes as input distance and point thresholds. At the end of the algorithm each player has its subset of the outliers in the dataset D.

Algorithm 3 Horizontal Distribution

Input: t Players, Datasets D_i for $i = 1$ to t, Total Dataset size n, Distance Threshold d_t, Point Threshold p_t
Output: Outliers M_i; $i = 1$ to t

At P_1:
  Compute LSH parameters k, L and w
  Generate A and B // A and B are $k \times L$ matrices
  Publish k, L, w, A and B
for each player $i = 1$ to t **do**
  $p'_t = (1 - p_t) \times n$
  $R = d_t / c$
  $T_i = LSH(D_i, R, A, B)$
  Compute b_t
  $M'_i = Pruning(D_i, T_i, p'_t, b_t)$
end for
for each player $i = 1$ to t **do**
  $BinLabels_i = \{$label of each bin in $T_i\}$
end for
$BinLabels = SecureUnion(BinLabels_i)$; i = 1 to t
for each player $i = 1$ to t **do**
  for $l = 1$ to $|BinLabels|$ **do**
    if $BinLabels(l) \in BinLabels_i$ **then**
      $C_i(l) = |T_i(l)|$
    else
      $C_i(l) = 0$
    end if
  end for
end for
$C = SecureSum(C_1, C_2, ..., C_t)$
for each player $i = 1$ to t **do**
  $\widehat{C}_i = C - C_i$
  for each object o in M'_i **do**
    $Neighbors_i = \bigcup_{j=1}^{L} T_i[g_j(o)]$
    $RepeatedNeighbors_i = \{q \in Neighbors_i \mid occurrence(q) \geq b_t\}$
    $req\_nn_i = p'_t - |RepeatedNeighbors_i|$
    $ValidBins_i = \{b_j \mid b_j = g_j(o)\ and\ \widehat{C}_i[g_j(o)] > req\_nn_i,\ j = 1, 2, ...L\ \}$
    if $|ValidBins_i| < b_t$ **then**
      $M_i = M_i \cup \{o\}$
    end if
  end for
  Return M_i
end for

Our algorithms are based on LSH scheme using p-stable distributions [3], where each hash function for a d-dimensional object v is computed as:

$$h_{a,b}(v) = \lfloor \frac{a.v + b}{w} \rfloor \tag{9}$$

While performing LSH on the local datasets, in each iteration of LSH, all the players need to use the same randomness so that, in any one iteration, all the objects are projected onto the same line. Hence, any one player, let us say P_1 computes the LSH parameters k and L and generates the random vectors A and B. Here, each element a of the $k \times L$ matrix A is a d-dimensional vector whose each entry is independently drawn from a p-stable distribution and each element b of the $k \times L$ matrix B is a random number in $[0, w]$. P_1 then publishes these values to all the other players. Each player then computes the modified point threshold $p'_t = (1 - p_t) \times n$ and LSH distance parameter R. Note that the parameter p'_t is computed based on the size of the entire dataset instead of the size of the local dataset. The LSH scheme is then run on the local dataset using the above computed parameters which outputs the binning structure T_i. (Here, the LSH scheme is a bit different from that of the centralized setting in that the random vectors A and B are given as input to the LSH scheme whereas in the actual LSH scheme these values are generated with in the LSH protocol.) The protocol *Pruning* is then invoked, which returns the set of local probable outliers M'_i.

In the Second phase, to form the set of global outliers, each player requires the total number of objects with all the other players that are hashed to each bin. Since the binning at each player is performed with the same global LSH parameters, we can find the correspondence between LSH bin structure across the players. However, the bin labels with each player might be different (considering only the non-empty bins). Hence the players communicate to form the union of all the bin labels (*BinLabels*) using the *SecureUnion* protocol [1] (steps 12-15). While forming *BinLabels*, each bin label needs to be indexed with the iteration number during LSH (1 to L) in order to differentiate between bins of same labels created during different iterations of LSH. Each player then counts the number of objects it has in each of the bins in *BinLabels* to form the local bin count structure C_i (steps 17-23). Next, the *SecureSum* protocol [1] is used to compute the global bin count structure C from the corresponding local bin count structures C_i of all the players (step 25). Each player i then locally computes the sum of the other $t - 1$ player's counts for all the bins i.e., $\widehat{C}_i = C - C_i$ (step 27).

At every player, each object o in the set of local probable outliers M'_i is then considered to determine whether it is a global outlier. To get the number of neighbors of o in the local dataset, the cardinality of the set $RepeatedNeighbors_i$ for o is computed. This step is actually redundant if the value is stored in the First phase (during *Pruning*). This value would be less than p'_t since the object was considered a local probable outlier in the First phase. The number of neighbors required to make it a non-outlier is computed as: $req\_nn_i = p'_t - |RepeatedNeighbors_i|$. To get the count of neighbors of o which are available with other players, the corresponding bins in $\widehat{C}_i$ (i.e. those L entries in $\widehat{C}_i$ indicated by $g_j(o)$ where $j = 1$ to L) are considered. If any of these L bins have count greater than req_nn_i, we can consider the object o to be a non-outlier. However, as explained in the centralized scheme, to reduce the false negative probability, we require b_t such bins to make the object a non-outlier. To achieve this, only those bins which have object count greater than req_nn_i are said to be $ValidBins_i$. If the cardinality of $ValidBins_i$ is less than b_t, the object o is considered as a global outlier and added to the set of global probable outliers M_i.

Algorithm 4 Vertical Distribution

Input: t Players, Dataset D vertically distributed among the players, Distance Threshold d_t, Point Threshold p_t
Output: Outliers M

1: **for** each player i = 1 to t **do**
2: $p'_t = (1 - p_t) \times |D_i|$
3: $R = d_t/c$
4: Compute LSH parameters k and L
5: Generate A_i where $dim(A_i) = d_i$
6: **for** each object o in D_i **do**
7: $S_i^o = (A_i \ . \ o)$ // S and A are $k \times L$ matrices
8: **end for**
9: **end for**
10: **for** each object o in D **do**
11: $S^o = SecureSum(S_1^o, S_2^o, .., S_t^o)$
12: **end for**
13: At P_1 :
14: Generate B and w // B is $k \times L$ matrix
15: **for** each object o in D_1 **do**
16: $H^o = \lfloor \frac{S^o + B}{w} \rfloor$ // H is $k \times L$ matrix
17: **end for**
18: Compute T using H
19: $M = Pruning(D_1, T, p'_t, b_t)$
20: Publish M

As in the centralized setting, the whole algorithm is run over multiple iterations and the intersection of all the global probable outlier sets is returned as the final set of outliers at each player. Since the procedure to find the global outliers is a bit different from that of finding outliers in the centralized scheme, the false positive rate increases slightly in comparison to that of the centralized algorithm. We discuss more about the performance of Algorithm 3 in the experiments section.

C. Vertical Distribution

In the case of vertically distributed data, each player collects information about different attributes for the same set of objects such that the union of all the attribute subsets equals the global attributes and all the attribute subsets are disjoint. The algorithm for PPOD over vertically partitioned data is executed in two phases. In the First phase, all

Table I
ALGORITHM COMPLEXITY

Setting	Round	Communication	Computation
Centralized	-	-	$O(ndL)$
Vertical	2	$O(nL)$	$O(ndL)$
Horizontal	3	$O(N_b L \log n)$	$O(ndL)$

players engage in communication to get the LSH binning of all the objects in the dataset considering all the attributes (distributed across players). In the Second phase, using this global binning information, each player locally computes the probable outliers.

The algorithm takes as input dataset D of dimensionality d, vertically distributed across t players such that $dim(D_i) = d_i$, $|D_i| = |D| = n$, for $i = 1$ to t and $\sum_{i=1}^{t} dim(D_i) = d$. The algorithm also takes as input the parameters d_t and p_t and outputs the set of outliers M. As explained earlier, in the LSH scheme based on p-stable distributions, each hash function for a d-dimensional object v is computed as shown in Equation 9. In the vertical distribution, where each player has some d_i dimensions of the total d dimensions, each player can locally generate the respective d_i entries of the vector a (this is possible since each entry of the d-dimensional vector a is independently chosen from a p-stable distribution) and compute its local share $(a_i.v_i)$ of the dot product $(a.v)$. The players then compute the *SecureSum* of their shares to get $(a.v)$. Since the values of b and w can be public (can be generated by one player and then be published), all the players can build the LSH binning structure using the dot products previously computed. Using this binning structure, each player can locally invoke the *Pruning* protocol to compute the set of probable outliers. To reduce the overall computation, all these operations can be carried out by any one player who then publishes the final outlier set. The procedure is outlined in Algorithm 4. Now we will explain the important steps of the given algorithm.

Initially, each player i locally generates A_i, where each element a_i is drawn from a p-stable distribution as explained before. In Steps 6-8 each player will compute their respective shares of the dot products for all the objects. In step 11, the *SecureSum* protocol is used to compute the dot product from all the shares. Steps 14-20 are carried out at any single player, let us say P_1. In step 14 the values of B and w are generated, where each element b of the $k \times L$ matrix B is a random number in $[0, w]$. In step 16, the previously computed dot products are used to evaluate the hash function values for all the objects. The LSH binning is generated using these hash values and then the *Pruning* protocol is invoked to compute the set of probable outliers M. Finally, this set M is published.

We can reduce the false positives by applying the same technique of centralized setting discussed in Section III-A2. That is, we run the entire Algorithm 4, $iter$ times and take the intersection of the resulting sets of probable outliers M_i for i = 1 to $iter$. Since all these runs are independent, we can execute them in parallel to lower the number of rounds.

IV. ANALYSIS

A. Centralized Setting

Computational Complexity: In Algorithm 1, steps 1 and 2 are of constant complexity. The complexity for computing the LSH (step 3) is $O(ndkL)$. The complexity of *Pruning* sub-protocol is $O(nLN_{pr})$. As explained in Section III-A3, $N_{pr} << n$. Considering only dominating terms and since $k << n$, the total complexity of the algorithm over a constant number of iterations is $O(ndL)$.

B. Horizontal Distribution

Computational Complexity: We give the computational complexity from the perspective of player P_1. The computational complexity of step 3 is $O(kL)$. Steps 8 and 10 have complexity of $O(n_1 dkL)$ and $O(n_1 N_{pr_1} L)$ respectively, where N_{pr_1} is the number of processed objects of player 1. The complexity of step 13 and steps 17-23 depends on the average number of non-empty bins N_b created during each iteration of LSH. In the worst case where each object in the dataset is hashed to a different bin, the number of bins would be equal to the dataset size. However, the number of bins would be much less than the total data size in the average case. Thus the average case complexity for step 13 and steps 17-23 is $O(N_b L)$; where $N_b << n$. Similarly, the complexity of step 27 is $O(N_b L)$. The complexity for steps 28-36 is $O(m'_1 L)$; where $m' = |M'_1|$. Considering the dominating terms, the computational complexity for player P_1 is $O(n_1 dL)$. The overall computational complexity of Algorithm 3 is $O(ndL)$.

Communication Complexity: In Algorithm 3, communication among the players happens in steps 4, 15 and 25. Step 4 has a communication complexity of $O(kL)$. The average case communication complexity for step 15 is $O(N_b L)$; where $N_b << n$. The corresponding average case communication complexity for step 25 is $O(N_b L \log n)$. Thus the overall communication complexity for Algorithm 3 is $O(N_b L \log n)$.

Security Analysis: In Algorithm 3, the values communicated in step 4 are public values and do not reveal any private information. After executing steps 15 and 25, each player has the count of objects in the entire database that are hashed to each bin. From this, each player can infer about the distribution of the objects of all the other players but cannot infer about the distribution of any single player when more than two players are involved (since *SecureUnion* and *SecureSum* are used). However, in most cases where the objects with each player come from a same distribution, this information is usually known before hand and thus there is no extra information revealed.

Table II
DATASET DESCRIPTION

Dataset	Objects	Attributes	Source
Corel	68040	32	kdd repository
MiniBooNE	130064	50	UCI repository
Landsat	275465	60	vision lab, ucsb
Darpa	458301	23	Lincoln Laboratory, MIT
Household	1000000	3	US Census Bureau

C. Vertical Distribution

Computational Complexity: We give the computational complexity from the perspective of player P_1. In Algorithm 4, steps 2-4 have constant complexity. Step 5 has a complexity of $O(kL)$. Step 7 has a complexity of $O(nd_1kL)$. Step 14 has a complexity of $O(kL)$. Steps 16 and 18 have a complexity of $O(nkL)$ and finally step 19 has a complexity of $O(nLN_{pr})$. Thus the complexity for player P_1 would be $O(nd_1L)$; where d_1 is the number of dimensions P_1 has. The overall computational complexity of Algorithm 4 is $O(ndL)$.

Communication Complexity: In Algorithm 4, communication among the players is necessary only in steps 11 and 20. Among these, step 11 is the dominating factor in terms of the communication complexity, where for each object in the dataset, we need to perform $k \times L$ *SecureSum* operations. Thus, the overall communication complexity of the algorithm would be $O(nL)$, since $k << n$.

Security Analysis: In Algorithm 4, the set of outliers is published by one player in step 20 and this would not reveal any private information since this is the desired output to be made public. In step 11, the *SecureSum* protocol is used to evaluate the dot product of all the shares with the players and hence no information about the local shares is revealed. Since each player knows the LSH binning considering all dimensions, some information about the global distribution of the dataset could be inferred. However, as explained in the case of horizontal distribution, this information is known beforehand in most of the cases.

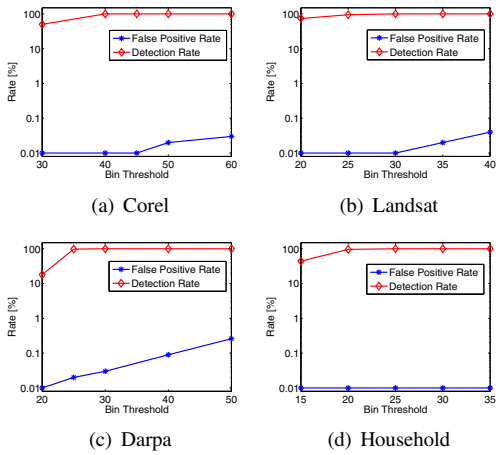
V. EXPERIMENTS

Experiments are performed on datasets listed in Table II. For all our experiments the approximation factor ϵ is set to 2 (empirically determined). All experiments are executed on Intel(R) Core $i7$ CPU $3.33GHz$ machine.

A. Centralized Setting

Execution Time: The execution time (in seconds) averaged over multiple runs of our centralized algorithm is tabulated in Column 2 of Table III. Since all the iterations of our algorithm are independent they can be run in parallel to achieve the speed up.

Bin Threshold: We computed the optimal bin threshold b_t as described in Section III-A2 for mentioned datasets and ran our algorithm using the same bin threshold. The results are summarized in Table III. Column 3 specifies the optimal value of b_t for each dataset and column 4 specifies



(a) Corel (b) Landsat

(c) Darpa (d) Household

Figure 1. Bin Threshold Vs False Positive and Detection rate

Table III
PERFORMANCE OF CENTRALIZED ALGORITHM

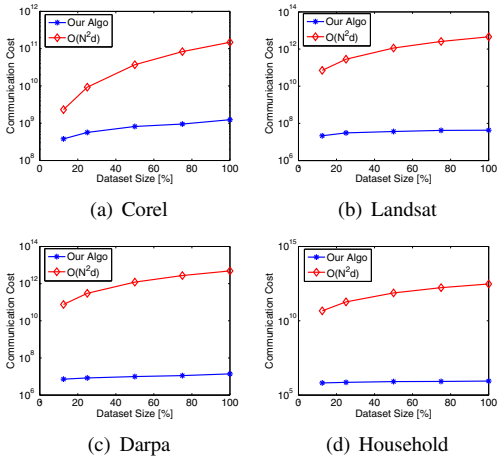
Dataset	Time (s)	b_t	FP (%)	N_{pr} (%)	p_t	NN
Corel	30.77	45	0.011	0.62	82	147
MiniBooNE	26.55	40	0.006	0.04	157	366
Landsat	44.11	30	0.014	0.36	331	929
Darpa	50.00	30	0.031	0.68	550	1103
Household	61.63	25	0.009	0.14	1200	1326

the number of false positives (in %) at the optimal b_t. As can be seen from the results, the false positive rate is quite less (the average false positive rate across the datasets is 0.0142%) proving that the approximation of our algorithm is very good. The detection rate for each dataset at the optimal b_t is 100% (i.e., no false negatives). Furthermore, for each of these false positives, we calculated the actual number of neighbors with in distance threshold d_t and listed the average number of neighbors (denoted as NN) for each false positive in column 7. For each dataset, this number NN is only marginally greater than the actual point threshold p_t (mentioned in column 6) when compared to the total dataset size. Hence, these false positives could be considered as weak non-outliers as claimed in Section III-A2 or even as outliers for many practical purposes.

Number of processed objects: For each dataset, the number of objects processed (N_{pr}) during *Pruning* protocol in Algorithm 1 are listed in column 5 (in %). The values in this columns shows the efficiency of our pruning technique and the number of objects processed is indeed less than 1% of the dataset size as is claimed in Section III-A3.

The effect of varying bin threshold on false positives and false negatives is shown in Figure 1. It is evident from the figure, as we increase the value of b_t, the number of false negatives decrease (i.e., detection rate increases) at the cost of increase in the number of false positives.

Iterations: The effect of varying the number of iterations $iter$ on the false positives at the optimal bin threshold



(a) Corel (b) Landsat

(c) Darpa (d) Household

Figure 2. Communication in Horizontal Distribution

is shown in Figure 3. As can be seen from the figure, the number of false positives decreases exponentially with increase in $iter$. Thus, the number of iterations to be run to achieve a very good approximation would be very less.

B. Distributed Settings

We discuss about the performance of the proposed privacy preserving algorithms for horizontal and vertical distributions. We have run the Algorithm 3 considering two players with uniform distribution of dataset among the players. At the optimal value of b_t, the rate of false positive increases by an average of 0.02% across the datasets, while the false negatives remain zero. We have computed the communication cost of the horizontal algorithm over 100 iterations for the 2-player case and compared it with that of Algorithm 1 given in [11], which has a cost of $O(n^2d)$ and gave the results for different datasets and by varying the dataset size in Figure 2. From the figure, it is clear that the rate of increase in the cost is very less compared to the $O(n^2d)$ method.

In case of vertical distribution, the performance of Algorithm 4 in terms of false positive and false negative rates remain same as that of the centralized algorithm. We ran Algorithm 4 considering two players with uniform distribution of dimensions and compared the communication cost with that of Algorithm 2 given in [11], which has a cost of $O(n^2)$. Up to datasets having size of order 10^6, the difference in the cost is up to order 10^3, but the difference would become more considerable as the dataset size increases.

VI. Conclusion

In this paper, we have proposed an approximate distance based outlier detection algorithm for the centralized setting, which is based on the LSH technique. Theoretical bounds on the approximation of our algorithm are given and supported by providing experimental results on various datasets. Next, we gave privacy preserving algorithms in both horizontal and vertical distributions of data with improved computation

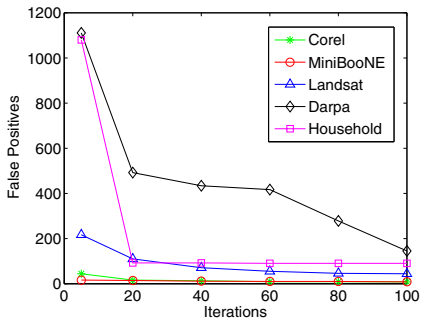


Figure 3. Effect of Iterations

as well as communication costs. Also, a comparison of the communication cost of both algorithms with the previous known result was given. One direction for future work is to use the LSH technique in other data mining tasks in order to achieve efficient privacy preserving algorithms.

Acknowledgment

This work was partly supported by Naval Research Board, India.

References

[1] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explor. Newsl.*, 2002.

[2] Z. Dai, L. Huang, Y. Zhu, and W. Yang. Privacy preserving density-based outlier detection. *Communications and Mobile Computing, International Conference on*, 1:80–85, 2010.

[3] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG '04*, pages 253–262.

[4] O. Goldreich. Foundations of cryptography volume 2-applications. 2004.

[5] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC '98*, pages 604–613.

[6] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB '98*, pages 392–403.

[7] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology CRYPTO 2000*, volume 1880, pages 36–54.

[8] M. Pillutla, N. Raval, P. Bansal, K. Srinathan, and C.V. Jawahar. LSH based outlier detection and its application in distributed setting. In *CIKM '11*.

[9] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *SIGMOD '00*, pages 427–438.

[10] M. Shaneck, Y. Kim, and V. Kumar. Privacy preserving nearest neighbor search. *Data Mining Workshops, International Conference on*, 0:541–545, 2006.

[11] J. Vaidya and C. Clifton. Privacy-preserving outlier detection. In *ICDM '04*, pages 233 – 240.

[12] Z. Zhou, L. Huang, Y. Wei, and Y. Yun. Privacy preserving outlier detection over vertically partitioned data. In *EBISS '09*, pages 1 –5.