Privacy-Preserving Simplicial Depth Query over Outsourced Cloud Platform

Hassan Mahdikhani, Rasoul Shahsavarifar, Rongxing Lu, David Bremner

Faculty of Computer Science, UNB

Abstract

The *simplicial depth* (*SD*) of a query point $q \in \mathbb{R}^d$ with respect to a dataset $S \subset \mathbb{R}^d$ is defined based on counting all $(d+1)$-dimensional *simplices* obtained from S that contain the query point q. Essentially, the simplicial depth is a ranking function which has been frequently used for sorting a multivariate dataset. However, when the dimension d is high, currently no better algorithm is known than the brute force method which takes $\Theta(n^{d+1})$ time. In order to deal with the challenge, offloading the computation to a powerful cloud server is an option. However, since the cloud server may be not fully trustable, directly delegating the process to untrusted cloud would be a source of serious security breaches and privacy concerns. In this paper, we will target the privacy preserving cloud-enhanced computation of simplicial depth, where the resource abundant cloud server will be employed to perform such time consuming computation while maintaining the client's privacy.

Simplicial Depth

The simplicial depth of a query point $q \in \mathbb{R}^d$ with respect to $S = \{x_1, ..., x_n\} \subset \mathbb{R}^d$ is defined as the total number of the closed simplices formed by data points that contain q.

$$SD(q; S) = \mu \sum\nolimits_{(x_1,...,x_{d+1})} I(q \in Conv[x_1, ..., x_{d+1}]), \tag{1}$$

where $\mu = 1/\binom{n}{d+1}$ is the normalization factor, and the convex hull $Conv[x_1, ..., x_{d+1}]$ is a closed simplex formed by $d+1$ points of S. For planar points $S = \{a, b, c, d, e\}$, Figure 5 illustrates that $SD(q_1; S) = 3/10$, whereas $SD(q_2; S) = 4/10$.

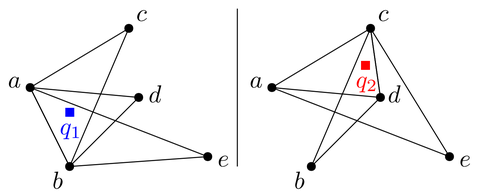

Figure 1: Planar simplicial depth

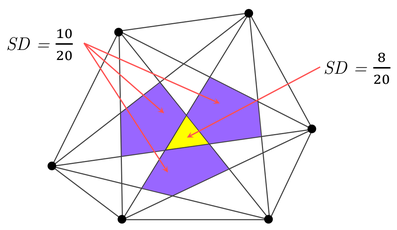

Figure 2: Simplicial depth contours

Homomorphic Encryption Techniques

The BFV Ring-LWE based homomorphic encryption is served as our underlying building block to encrypt the uploaded dataset.

- Parameter Generation: Given the security parameter λ, corresponding output $(d, \kappa, t, \chi_{key}, \chi_{err}, w)$ will be obtained, where:
  - Polynomial modulus d forms ring $R = \mathbb{Z}[x]/(\Phi_d(x)$, in which $\Phi_d(x) \in \mathbb{Z}[x]$.
  - coefficient modulus κ and plaintext modulus t satisfy $1 < t < \kappa$.
  - Two bounded probability distributions χ_{key} and χ_{err} are chosen based on R.
  - $w > 1$ is the base of logarithm in $\ell = \lfloor log_w \kappa \rfloor$.
- Key Generation:
  - Secret key SK $= s$ can be obtained by sampling on χ_{key}.
  - Public key PK $= (b, a)$, where $b = [-(as + e)]_\kappa$ in which a and e are two uniformly random samples as: $a \leftarrow R_\kappa$ and $e \leftarrow \chi_{err}$.
  - Relinearisation key RelinK $= \zeta = ([w^i \cdot s^2 - (e_i + a_i \cdot s)]_\kappa, a_i) \in R^\ell$, where $a_i \leftarrow R_\kappa$ and $e_i \leftarrow \chi_{err}$ for $i \in [0, 1, ..., \ell]$.
- Encryption: Given PK $= (b, a)$, ciphertext $\mathbf{c}$ for plaintext message m in message space R/tR can be calculated as $\mathbf{c} = (c_0, c_1) = \left([\Delta\,[m]_t + bu + e_1]_\kappa, [au + e_2]_\kappa\right) \in R^2$, where $e_1, e_2 \leftarrow \chi_{err}$, $u \leftarrow \chi_{key}$, and $\Delta = \lfloor \kappa/t \rfloor$.
- Decryption: Given the ciphertext $\mathbf{c} = (c_0, c_1)$, the corresponding message $m = \left[\lfloor \frac{t}{\kappa} \cdot [c_0 + c_1 s]_\kappa \rceil\right]_t \in R$ can be recovered by the secret key SK $= s$.
- Addition ($\oplus$): Given two ciphertexts $\mathbf{c} = (c_0, c_1)$ and $\mathbf{c'} = (c'_0, c'_1)$, $c_\oplus = ([c_0 + c'_0]_\kappa, [c_1 + c'_1]_\kappa)$.
- Multiplication ($\otimes$): Given two ciphertexts $\mathbf{c} = (c_0, c_1)$, $\mathbf{c'} = (c'_0, c'_1)$, and relinearisation key RelinK $= \zeta = (\zeta_0, \zeta_1)$, $c_\otimes =$ Relinearization$(\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \zeta)$ $= \left((\mathcal{C}_0 + \sum_{i=0}^{\ell} \zeta_{0i} \cdot \mathcal{C}_{2i}), (\mathcal{C}_1 + \sum_{i=0}^{\ell} \zeta_{1i} \cdot \mathcal{C}_{2i})\right)$, where $\mathcal{C}_0 = \left[\lfloor \frac{t}{\kappa} \cdot c_0 \cdot c'_0 \rceil\right]_\kappa$, $\mathcal{C}_1 = \left[\lfloor \frac{t}{\kappa} \cdot (c_0 \cdot c'_1 + c_1 \cdot c'_0) \rceil\right]_\kappa$, and $\mathcal{C}_2 = \left[\lfloor \frac{t}{\kappa} \cdot c_1 \cdot c'_1 \rceil\right]_\kappa$.

System Model

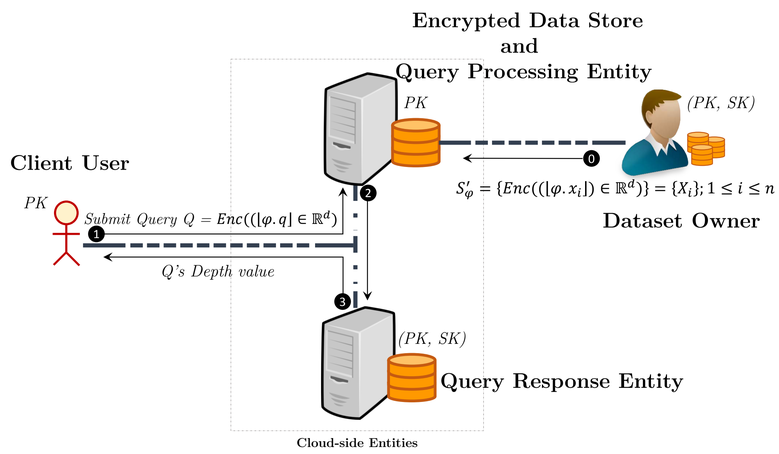

Figure 3: System model of the privacy-preserving simplicial depth query

$$\begin{aligned} &DetCompute(A, B, C) \\ =&det(B \ominus A, C \ominus A) = det\left(\begin{bmatrix} B_1 \ominus A_1 & C_1 \ominus A_1 \\ B_2 \ominus A_2 & C_2 \ominus A_2 \end{bmatrix}\right) \\ =&((B_1 \ominus A_1) \otimes (C_2 \ominus A_2)) \ominus \ ((B_2 \ominus A_2) \otimes (C_1 \ominus A_1)). \end{aligned}$$

and

$$Dets = \{DetCompute(X_i, X_j, X_k)\} \tag{2}$$
$$DetsSq = Dets \circledast Dets = \{D \otimes D;\ D \in Dets\}, \tag{3}$$

.

Algorithm 1 Intermediate Batch Results

Input: Q, S'_φ, Precomputed $Dets$ and $DetsSq$
Output: Encrypted batch $\$ = \left\{\{\alpha, \beta, \gamma\}_t; 1 \le t \le \binom{n}{3}\right\}$

$\$ \leftarrow \emptyset$
for each $X_i, X_j, X_k \in S'_\varphi$ **do**
  $D_{ijk} \leftarrow [Dets]_{ijk}$
  $\alpha' \leftarrow DetCompute(X_i, Q, X_k) \otimes D_{ijk}$
  $\beta' \leftarrow -DetCompute(X_i, Q, X_j) \otimes D_{ijk}$
  $\gamma' \leftarrow [DetsSq]_{ijk} \ominus \alpha \ominus \beta$
  $(R_1, R_2, R_3) \leftarrow (Enc(r_1), Enc(r_2), Enc(r_3))$;
  r_1, r_2, r_3 are positive integer random values
  $\{\alpha, \beta, \gamma\}_t \leftarrow (R_1 \otimes \alpha', R_2 \otimes \beta', R_3 \otimes \gamma')$
  $\$ \leftarrow \$ \cup \{\alpha, \beta, \gamma\}_t$
return $\$$

Algorithm 2 Response Depth Value

Input: Encrypted batch $\$ = \left\{\{\alpha, \beta, \gamma\}_t; 1 \le t \le \binom{n}{3}\right\}$
Output: Normalized $DepthValue$

$DepthValue \leftarrow 0$
for each $\{\alpha, \beta, \gamma\} \in \$$ **do**
  $(A, B, \Gamma) \leftarrow (Dec(\alpha), Dec(\beta), Dec(\gamma))$
  if ($A \ge 0$ *and* $B \ge 0$ *and* $\Gamma \ge 0$)
    $DepthValue \leftarrow DepthValue + 1$
return $DepthValue/|\$|$

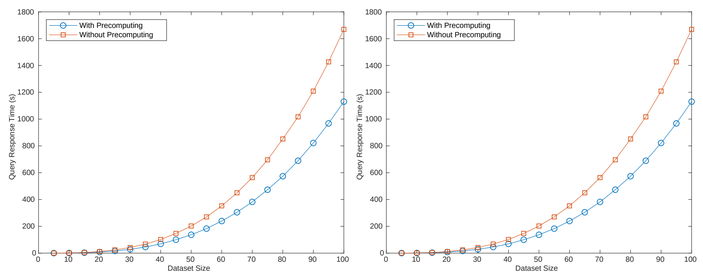

Figure 4: Computational costs and response time

UNB EST. 1785 UNIVERSITY OF NEW BRUNSWICK
Computer Science

Experimental Results

To evaluate the performance of the proposed *PSDQ* scheme, we implemented both plaintext and ciphertext algorithms for computing planar simplicial depth of a query point.

Table 1: The Parameter Settings

Parameter	Value
Security parameter λ	$\lambda = 128$
Coeffient modulus $\kappa = \kappa_1.\kappa_2$	$\lvert\kappa_1\rvert = 55, \lvert\kappa_2\rvert = 56$
Polynomial modulus $\Phi_d = x^d + 1$	$d = 4096$
Plaintext modulus t	$t = 256$
Dataset size n	$n \in \{5, 10, 15, ..., 100\}$
Scale factor φ	$\varphi \in \{10, 100, 1000, 10000\}$

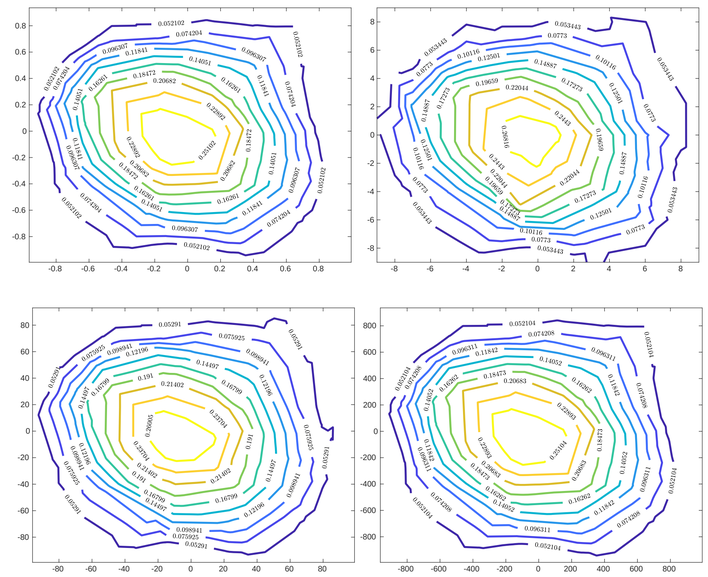

Figure 5: Original dateset, $\varphi = 10, 100, 1000$.

$$|\,SD_a, SD_b\,|_e = 1 - r^2 \tag{4}$$
$$|\,SD_a, SD_b\,|_c = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} |M_{ij}^{SD_a} - M_{ij}^{SD_b}|}{n^2 - n}, \tag{5}$$

where $M_{n\times n}^P$ is defined as follows:

$$M_{ij}^D = \begin{cases} 1 & depth_D(x_i) \le depth_D(x_j) \\ 0 & \text{otherwise.} \end{cases}$$

The accuracy of the results based on the geometric mean and the arithmetic mean of two dissimilarity values can be computed in two following forms.

$$Acc_1 = 100\left(1 - \sqrt{|\,SD_a, SD_b\,|_e \cdot |\,SD_a, SD_b\,|_c}\right)$$
$$Acc_2 = 100\left(1 - \frac{|\,SD_a, SD_b\,|_e + |\,SD_a, SD_b\,|_c}{2}\right),$$

.

Table 2: A Comparison between Dissimilarity Values

S'_φ	$\lvert(SD, SD_{PSDQ})\rvert_e$	$\lvert(SD, SD_{PSDQ})\rvert_c$	Acc_1	Acc_2
$\varphi = 10$	0.0359	0.0763	94.77	94.39
$\varphi = 100$	1.6860e-04	0.0116	99.86	99.41
$\varphi = 1000$	8.1835e-06	0.0012	99.99	99.94
$\varphi = 10000$	0	0	100	100

Conclusion and Future Work

In this work, we presented a privacy preserving scheme namely *PSDQ* to compute the simplicial depth of a query point. In other words, considering the privacy preserving, computing the simplicial depth of a query point $q \in \mathbb{R}^2$ with respect to a dataset $S \subset \mathbb{R}^2$, is outsourced. To do this, BFV RLWE-based homomorphic encryption scheme has been considered in order to offload both storing and computing of encrypted dataset. The experimental results evaluate the performance and correctness of the developed scheme. In our future work, an extended scheme will be developed as a framework to outsource computing different depth functions. Another future work direction would be generalizing the scheme in order to deal with various data depth related applications such as depth contours.