1–33, Sep 2021

Splintering with distributions and polytopes: Unconventional schemes for private computation

Praneeth Vepakomma, Julia Balla, Susovan Pal, Ramesh Raskar
vepakom@mit.edu
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Abstract

Performing computations while maintaining privacy is an important problem in todays distributed machine learning solutions. Consider the following two set ups between a client and a server, where in setup i) the client has a public data vector $\mathbf{x}$, the server has a large private database of data vectors $\mathcal{B}$ and the client wants to find the inner products $\langle \mathbf{x}, \mathbf{y_k} \rangle, \forall \mathbf{y_k} \in \mathcal{B}$. The client does not want the server to learn $\mathbf{x}$ while the server does not want the client to learn the records in its database. This is in contrast to another setup ii) where the client would like to perform an operation solely on its data, such as computation of a matrix inverse on its data matrix $\mathbf{M}$, but would like to use the superior computing ability of the server to do so without having to leak $\mathbf{M}$ to the server. We present a stochastic as well as a polyhedral combinatorics scheme for splitting the client data into privatized shares that are transmitted to the server in such settings. The server performs the requested operations on these shares instead of on the raw client data at the server. The obtained intermediate results are sent back to the client where they are assembled by the client to obtain the final result. We present such schemes for operations such as sigmoid, softmax, matrix inverse, inner product on real-valued data and for linear computation on one-hot encoded data.

Keywords: List of keywords

1. Introduction

We describe a method for private computation in distributed machine learning via privatized stochastic shares of sensitive data called splinters. This enables client devices to receive services from server without directly sharing any sensitive data. The server performs computations over these splinters and the corresponding results are sent back to the client. The client has required private coefficients to perform specific operations referred to as unsplintering over these intermediate results in order to obtain the required final result as if the sensitive data itself was sent across to the server instead of the communicated splinters. The approach is relatively communication efficient, supports composition of operations and does not require a trusted non-colluding third party to mediate the process. Replication/download of the sensitive database is not required during this process. We show an application of this proposed method for privately performing operations such as inner product and matrix inverse. Such a private computation with such fundamental operations results in a wide array of societal applications such as private machine learning, private information retrieval, private set intersection, private stream searching, private scientific computation and private information storage.

2. Related work

1. **Formal notions of privacy** There are several formally established notions of privacy such as (ϵ, δ)-differential privacy (Dwork, 2008; Dwork and Smith, 2010; Dwork et al., 2011, 2014; Abadi et al., 2016; Machanavajjhala et al., 2008; Nissim et al., 2012; McSherry and Talwar, 2007; Nissim et al., 2007, 2012, 2017), Lipschitz privacy (Koufogiannis et al., 2015a,b; Chatzikokolakis et al., 2013; Koufogiannis, 2017; Koufogiannis and Pappas, 2016), Blowfish privacy (He et al., 2014; Nie et al., 2010; Machanavajjhala and Kifer, 2015), Pufferfish privacy (Kifer and Machanavajjhala, 2014; Song et al., 2017; Kifer and Machanavajjhala, 2012) which allows the user to specify a class of protected predicates that must be learned subject to the guarantees of differential privacy, and all other predicates can be learned without differential privacy, Minimax filter, ON-OFF privacy, Secure Shuffling, concentrated differential privacy, local differential privacy, central differential privacy and Renyi differential privacy (Mironov, 2017; Wang et al., 2018; Geumlek et al., 2017). Each of these notions of privacy has a formal mathematical definition of privacy. There is a lengthy body of work of several privacy-preserving mechanisms that can help attain one or more of these stringent notions of privacy, depending on the statistical query or statistical model being privatized in the presence of any constraints that might be present.

2. **Cryptographic computation** This can be categorized into techniques for homomorphic encryption (Gentry and Boneh, 2009; Naehrig et al., 2011; Van Dijk et al., 2010; Brakerski and Vaikuntanathan, 2011; Stehlé and Steinfeld, 2010; Brakerski and Vaikuntanathan, 2014; Brakerski et al., 2014; Gentry and Halevi, 2011; Fan and Vercauteren, 2012; Smart and Vercauteren, 2010; Damgård et al., 2012; Fontaine and Galand, 2007; Cramer et al., 2001; Sathya et al., 2018) and secure multi-party computation Yao (1982b,a); Ishai et al. (2007); Evans et al. (2017, 2018); Bogetoft et al. (2009); Lindell (2005); Goldreich (1998); Canetti et al. (1996); Cramer et al. (2015); Ben-David et al. (2008); Garg et al. (2014); Hirt et al. (2000); Atallah and Du (2001)

3. Linear Splintering

We now detail the first-order idea of splintering.

For a d dimensional input query vector $\mathbf{x}$, the client device creates d shares corresponding to $\mathbf{x}$ as $\{\mathbf{z_1}, \mathbf{z_2} \dots \mathbf{z_d}\}$ so that

$$\mathbf{x} = Splint(\mathbf{z_1}, \mathbf{z_2} \dots \mathbf{z_d}), \forall i \in 1..d$$

The most basic splint function that allows for such a representation is a linear combination using coefficients α_i as

$$\mathbf{x} = \sum_{i}^{d} \alpha_i \mathbf{z_i}, \forall i \in 1..d$$

The $\alpha_i's$ are private to the client and not shared with any other entity, be it another client or a server. The splinters $\mathbf{z_i}$ are shared with the server. The server performs a set of application dependent operations on the splinters $\mathbf{z}_i, \forall i \in 1 \dots d$ and sends results $\{\beta_i\}$ back to client on either all or a subset of the d shares. The client performs a local computation called $UnSplint$ using original shares $\mathbf{z_i}$, its corresponding α_i's that are known only to the client and received $\beta_i's$ obtained

from the server. This unsplinting operation reveals the true result l of the intended application to the client.

$$l = UnSplint(\alpha_i, \mathbf{z_i}, \beta_i), \forall i \in 1..d$$

Note that although $\mathbf{x}$ is represented via a linear combination, the computation of $\{\beta_i\}$ and $UnSplint$ is not necessarily linear.

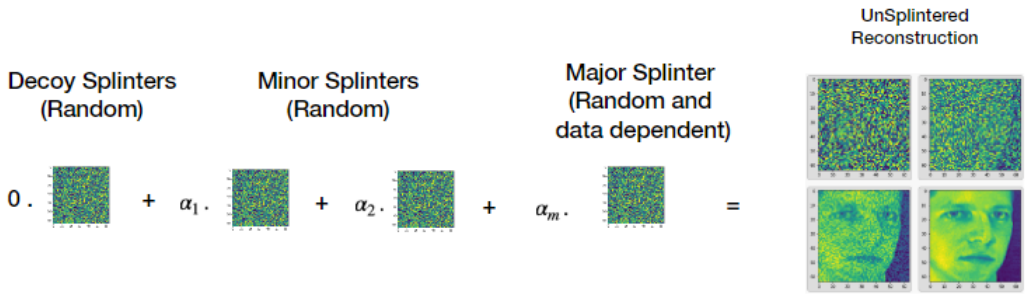


Figure 1: We show an example of splintering a face image using decoy splinters, minor splinters and major splinters. On the right side, we show the effect of the reconstruction on the precision of choosing the right unsplintering coefficients denoted by α's.

3.0.1. TERMINOLOGY: TYPES OF SPLINTERS

Decoy Splinters: Data independent splinters z_i whose corresponding coefficients α_i are zero.
Minor Splinters: Data independent splinters z_i whose corresponding coefficients α_i are non-zero.
Major Splinters: Data dependent splinters z_i whose corresponding coefficients α_i are non-zero.
Attackers would need to be able to distinguish decoy splinters from the rest of splinters, while estimating their corresponding coefficients in order to reconstruct the raw data. A good scheme for generating the splinters would need to prevent that attack.

4. Summary of results

In the rest of paper, we propose following results with regards to our splintering scheme

1. We propose splintering schemes for nonlinear operations such as sigmoid, softmax, matrix inverse, inner products and for computation on categorical/label data that is in a one-hot encoded format.

2. We propose two schemes for generating splinters:

   (a) Stochastic: We provide a modified EM algorithm for learning structured mixture distributions that minimize KL divergence between Gaussian mixtures that are used to sample different kinds of splinters. This helps us to learn distributions to sample decoy splinters from; such that they are hard to distinguish from minor and major splinters. We utilize lower bounds on sample complexity of learning Gaussian mixtures to determine, when the learnt mixture model needs to be refreshed.

(b) Combinatorial: We provide a scheme that exploits Birkhoff-vonNeumann decomposition theorem to perform the splintering and unsplintering operations. We exploit the non-unique nature of decomposing doubly stochastic matrices into permutation matrices and share lower bounds for this approach.

3. In the stochastic case, we connect KL-divergence between Gaussians mixtures to distance correlation (an RKHS distance measure). Such connections between information theoretic measures and RKHS distance measures have not been made in prior literature so far.

4. We provide upper and lower bounds on distance correlation.

5. We connect distance correlation minimization to mutual information minimization and show its equivalence to error maximizing linear regression with respect to learning Gaussian distributed covariates instead of the coefficients.

5. Linear and Nonlinear splintering

We propose examples of splintering schemes below for the operations of i) Sigmoid, iii) Softmax, iv) Matrix inverse and v)inner-product.

5.1. Splintering for sigmoid

Theorem 1 *For an input x expressed using splinters $\{z_1, z_2, \ldots, z_k\}$ as $x = \sum_{i=1}^{k} \alpha_i Z_i$, the unsplintering operation to compute $s(x)$ using $s(z_1), s(z_2), \ldots, s(z_k)$ is given by*

$$s(x) = \frac{1}{1 + \prod_{i=1}^{k} \left(\frac{1-s(Z_i)}{s(Z_i)}\right)^{-\alpha_i}}$$

Proof The sigmoid function is given by

$$s(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} \tag{1}$$

This can be rearranged as

$$\frac{1-s(x)}{s(x)} = e^{-x} \tag{2}$$

Now following our proposed approach of weighted splinter method we substitute $x = \sum_{i=1}^{k} \alpha_i Z_i$ in eqn. 1 to get

$$s(x) = \frac{1}{1+e^{-\sum_{i=1}^{k} \alpha_i Z_i}} \tag{3}$$

$$= \frac{1}{1+\prod_{i=1}^{k} e^{-\alpha_i Z_i}} \tag{4}$$

Now substituting 2 above we get

$$s(x) = \frac{1}{1 + \prod_{i=1}^{k} \left(\frac{1-s(Z_i)}{s(Z_i)}\right)^{-\alpha_i}} \tag{5}$$

Therefore the final scheme only requires computing $s(Z_i)'s$ at the server while the client can figure out $s(x)$ as α_i's are secret with the client and not shared with server.

■

5.2. Splintering for matrix inverse

We now propose a splintering scheme for the important operation of matrix inversion in this subsection. We consider a setting where the client has a large sensitive matrix $\mathbf{M}_{n\times n}$ and would like to use the service of a computationally powerful server in order to privately obtain the inverse $\mathbf{M}^{-1}$.

Theorem 2 *The unsplintering operation to compute the matrix inverse M using splinters $Z_1, Z_2, \ldots, Z_r$ with secret coefficients $\alpha_1, \alpha_2, \ldots \alpha_r$ while only having to compute the inverse of the splinters is given by*

$$(\alpha_1\mathbf{Z_1} + \mathbf{UZ_0V})^{-1} = \mathbf{1}/\alpha_1\mathbf{Z_1}^{-\mathbf{1}} - \mathbf{1}/\alpha_1\mathbf{Z_1}^{-\mathbf{1}}\mathbf{U}(\mathbf{Z_0}^{-\mathbf{1}} + \mathbf{V1}/\alpha_1\mathbf{Z_1}^{-\mathbf{1}}\mathbf{U})^{-\mathbf{1}}\mathbf{1}/\alpha_1\mathbf{VZ_1}^{-\mathbf{1}}$$

where $Z_0 = \begin{bmatrix} Z_1 & & \\ & \ddots & \\ & & Z_r \end{bmatrix}$ *and* $U = \begin{bmatrix} U_1 \ldots & U_r \end{bmatrix}, V = \begin{bmatrix} V_1 \ldots & V_r \end{bmatrix}$ *such that* $\mathbf{U_i Z_i V_i}$ *is of the form*

$$\underbrace{\begin{bmatrix} \alpha_{i3} & & \\ & \alpha_{i3} & \\ & & \alpha_{i3} \end{bmatrix}}_{\mathbf{U}} \mathbf{Z_i} \underbrace{\begin{bmatrix} \alpha_{i4} & & \\ & \alpha_{i4} & \\ & & \alpha_{i4} \end{bmatrix}}_{\mathbf{V}} \tag{6}$$

Proof In order to obtain the inverse of a private matrix $\mathbf{M}_{n\times n}$, we split it into the form using $\mathbf{A}_{n\times n}, \mathbf{U}_{n\times k}, \mathbf{V}_{k\times n}$ and $\mathbf{Z_2}$ of dimension $k \times k$ as

$$\mathbf{M^{-1}} = (\mathbf{A} + \mathbf{UZ_2V})^{-1} \tag{7}$$

where $\mathbf{A}$ is written in terms of a splinter matrix $\mathbf{Z}_1$ of dimension $n \times n$ as

$$A = \alpha_1\mathbf{Z_1} \tag{8}$$

and $\mathbf{UZ_2V}$ is of the form

$$\underbrace{\begin{bmatrix} \alpha_3 & & \\ & \alpha_3 & \\ & & \alpha_3 \end{bmatrix}}_{\mathbf{U}} \mathbf{Z_2} \underbrace{\begin{bmatrix} \alpha_4 & & \\ & \alpha_4 & \\ & & \alpha_4 \end{bmatrix}}_{\mathbf{V}} \tag{9}$$

where $\mathbf{Z}_2$ is the other splinter. Now by the pupular matrix inversion lemma (Sherman–Morrison–Woodbury formula)

$$(\mathbf{A} + \mathbf{UZ_2V})^{-1} = \mathbf{A^{-1}} - \mathbf{A^{-1}U}(\mathbf{Z_2}^{-\mathbf{1}} + \mathbf{VA^{-1}U})^{-\mathbf{1}}\mathbf{VA^{-1}}$$

Therefore the proposed scheme now is to send $\mathbf{Z_1}, \mathbf{Z_2}$ to the server which sends back $\mathbf{Z_1^{-1}}, \mathbf{Z_2^{-1}}$ to the client that holds $\mathbf{M}$ along with the secret coefficients $\alpha_1, \alpha_2, \alpha_3$. The client then obtains the final solution $\mathbf{M}^{-1}$ by computing

$$(\alpha_1\mathbf{Z_1} + \mathbf{UZ_2V})^{-1} = \mathbf{1}/\alpha_1\mathbf{Z_1}^{-\mathbf{1}} - \mathbf{1}/\alpha_1\mathbf{Z_1}^{-\mathbf{1}}\mathbf{U}(\mathbf{Z_2}^{-\mathbf{1}} + \mathbf{V1}/\alpha_1\mathbf{Z_1}^{-\mathbf{1}}\mathbf{U})^{-\mathbf{1}}\mathbf{1}/\alpha_1\mathbf{VZ_1}^{-\mathbf{1}}$$

Now this can be generalized to 3 splinters as follows where

$$\mathbf{U} = \begin{bmatrix}\mathbf{U_1} & \mathbf{U_2}\end{bmatrix}, \mathbf{V} = \begin{bmatrix}\mathbf{V_1} & \mathbf{V_2}\end{bmatrix}, \mathbf{Z} = \begin{bmatrix}\mathbf{Z_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Z_2}\end{bmatrix}$$

so that

$$\mathbf{A} + \mathbf{U_1 Z_1 V_1^T} + \mathbf{U_2 Z_2 V_2^T} = \mathbf{A} + \mathbf{U Z V^T}$$

Then, just apply the Woodbury matrix identity as above to complete the proof. ■

5.2.1. COMPUTATIONAL SAVINGS

In cases where $n >> k$, the matrix $(\mathbf{Z_2}^{-1} + \mathbf{V A^{-1} U})^{-1}$ which is of dimension $k \times k$ is much easier to invert than the original private data matrix $\mathbf{M}_{n \times n}$ thereby offloading the heavier computation onto the server while preserving privacy and requiring a much smaller computation on the client.

5.3. Splintering for softmax

Theorem 3 *For an input x expressed using splinters $\{z_1, z_2, \ldots, z_k\}$ as $x = \sum_{i=1}^{k} \alpha_i Z_i$, the unsplintering operation to compute $s(x)$ where $e^{Z_1} = a_1, e^{Z_1} = a_2$ and $e^{Z_3} = a_3$ and $W_1 = \frac{a_1}{a_1+a_2+a_3}$, $W_2 = \frac{a_2}{a_1+a_2+a_3}$ and $W_3 = \frac{a_3}{a_1+a_2+a_3}$ as follows*

$$\textit{softmax}(\alpha_1 Z_1 + \alpha_2 Z_2)_i = \frac{(e^{softmax^{-1}(\boldsymbol{W_1}; Z_{11})_i})^{\alpha_1} (e^{softmax^{-1}(\boldsymbol{W_2}; Z_{11})_i})^{\alpha_2}}{\sum_{j=1}^{d} (e^{softmax^{-1}(\boldsymbol{W_1}; Z_{11})_j})^{\alpha_1} (e^{softmax^{-1}(\boldsymbol{W_2}; Z_{11})_j})^{\alpha_2}} \tag{10}$$

Proof

$$\text{softmax}(\alpha_1 Z_1 + \alpha_2 Z_2)_i = \frac{e^{\alpha_1 Z_{1i} + \alpha_2 Z_{2i}}}{\sum_{j=1}^{d} e^{\alpha_1 Z_{1j} + \alpha_2 Z_{2j}}} \tag{11}$$

$$= \frac{(e^{\alpha_1 Z_{1i}})(e^{\alpha_2 Z_{2i}})}{\sum_{j=1}^{d} (e^{\alpha_1 Z_{1j}})(e^{\alpha_2 Z_{2j}})} \tag{12}$$

Assume one component each of $\boldsymbol{Z_1}$,$\boldsymbol{Z_2}$ are known and let $\boldsymbol{W_i} = \text{softmax}(\boldsymbol{Z_1})$. Then $Z_i = \text{softmax}^{-1}(W_i; Z_{11})$. Therefore we plugin this inverse in 11 to rewrite softmax as

$$\text{softmax}(\alpha_1 Z_1 + \alpha_2 Z_2)_i = \frac{(e^{softmax^{-1}(\boldsymbol{W_1}; Z_{11})_i})^{\alpha_1} (e^{softmax^{-1}(\boldsymbol{W_2}; Z_{11})_i})^{\alpha_2}}{\sum_{j=1}^{d} (e^{softmax^{-1}(\boldsymbol{W_1}; Z_{11})_j})^{\alpha_1} (e^{softmax^{-1}(\boldsymbol{W_2}; Z_{11})_j})^{\alpha_2}} \tag{13}$$

Analytical inverse of softmax when one component is known: Let $e^{Z_1} = a_1, e^{Z_1} = a_2$ and $e^{Z_3} = a_3$. Then $W_1 = \frac{a_1}{a_1+a_2+a_3}$, $W_2 = \frac{a_2}{a_1+a_2+a_3}$ and $W_3 = \frac{a_3}{a_1+a_2+a_3}$ and therefore

$$\frac{W_1}{e^{Z_1}} = \frac{W_2}{e^{Z_2}} = \frac{W_3}{e^{Z_3}} = \lambda \tag{14}$$

This implies that

$$Z_2 = log\left(\frac{W2}{W1}\right) e^{Z_1} \tag{15}$$

■

5.4. SecureHull: Dual splintering for categorical/label data

We now provide a modified version of splintering that is suitable for data that is one-hot encoded (i.e, in the form of permutation matrices). This is usually the defacto format used in machine learning applications for such data. Prior to that we first share some preliminaries required for this problem.

5.4.1. PRELIMINARIES FOR SECUREHULL: POLYHEDRAL COMBINATORICS

Definition 4 (Doubly stochastic matric) *A square matrix $A = (a_{ij})$ of nonnegative real numbers, each of whose rows and columns sums satisfy the condition $\sum_i a_{ij} = \sum_j a_{ij} = 1$ is doubly stochastic.*

Definition 5 (Sub-permutation matrix) *A doubly-stochastic matrix is a sub-permutation matrix if every row and every column has at most one non-zero entry with value 1.*

Theorem 6 (Birkhoff–von Neumann theorem) *The Birkhoff–von Neumann theorem states that the polytope of $n \times n$ doubly B_n is the convex hull of the set of $n \times n$ permutation matrices, and furthermore that the vertices of B_n are precisely the permutation matrices. In other words, if A is doubly stochastic matrix, then there exist $\theta_1, \ldots, \theta_k \geq 0, \sum_{i=1}^{k} \theta_i = 1$ and permutation matrices $P_1, \ldots, P_k$ such that $A = \theta_1 P_1 + \cdots + \theta_k P_k$. This representation is also known as the Birkhoff–von Neumann decomposition.*

5.5. Approach

In this approach, we share a specific combination of splinters instead of the splinters themselves as seen so far in the sections on sigmoid, softmax and matrix inverse.

Desirable primitives of Birkhoff-von Neumann decomposition: Typically decompositions such as eigen-decompositions are unique (when all eigenvalues are unique). But the relationship between doubly stochastic matrices to permutation matrices is non-unique, i.e there are multiple ways in which convex combination of permutation matrices can be used to represent a doubly stochastic matrix. We exploit this in our proposed scheme as follows.

The raw one-hot encoded matrix $\mathbf{M}$ first rescaled with a coefficient $\alpha_* \in (0, 1)$ to get $\mathbf{M}_*$. This scaled matrix is combined (convex combination) with several secret permutation matrices $\mathbf{P_1}, \mathbf{P_2}, \ldots, \mathbf{P_k}$ and coefficients to obtain a doubly stochastic matrix $\mathbf{D}$. This $\mathbf{D}$ matrix is instead shared with the server to perform any linear computations and the result is shared with the client. The client then computes $\mathbf{P} = \sum_{i=1}^{k} \alpha_i \mathbf{P_i}$ and shares $\mathbf{P}_*$ with the server upon the rescaling $\mathbf{P}_* = \alpha_* P$. The server cannot reconstruct $\mathbf{M}$ due to the lack of secret coefficients and the fact that the decomposition from doubly stochastic $\mathbf{D}$ to the permutations is non-unique. The server now performs the linear operation on $\mathbf{P}$ and sends back the result. The client has everything now to obtain the corresponding final result on $\mathbf{M}$ as if it had sent it instead while still protecting its reconstruction by the server. We refer to this proposed algorithm as *SecureHull* and share it in the algorithmic block below.

5.6. Lower and upper bounds on # of possible Birkhoff von Neumann decompsitions of a doubly stochastic matrix

The # of possible Birkhoff von Neumann decompsitions of a doubly stochastic matrix is lower bounded by the square of # of positive elements in the doubly stochastic matrix $\mathbf{D}_n \times n$ and upper

Algorithm 1 SecureHull

Input: One-hot encoded data $\mathbf{M}$ is scaled down as $\mathbf{M}_* = \frac{1}{\alpha_*}\mathbf{M}$ with $\alpha_* \in (0, 1)$
Generate: Client generates;
Secret coefficients $\{\alpha_1, \ldots, \alpha_k\} \in \mathbb{R}^+$ and Permutation matrices $\mathbf{P_1}, \mathbf{P_3} \ldots \mathbf{P_k}$.
Create: Client creates doubly stochastic matrix $\mathbf{D} = \sum_{i=1}^{k} \alpha_i \mathbf{P_i} + \mathbf{M}_*$
Compute: Server computes $\mathbf{D}.\mathbf{y} = \mathbf{c}$ and sends the result $\mathbf{c}$ back to client
Compute: Client computes $\mathbf{P} = \sum_{i=1}^{k} \alpha_i \mathbf{P_i}$
Rescale: Client rescales $\mathbf{P}_* = \frac{1}{\beta}\mathbf{P}$ and sends $\mathbf{P}_*$ to server.
Compute: Server computes $\mathbf{P}_*.y = d$ and sends the result to client.
Compute: Client computes $\mathbf{M}.\mathbf{y} = \alpha_* \times (\mathbf{c} - \beta\mathbf{d})$ to obtain the final result.

bounded by $n^2 - 2n + 2$ as in Brualdi (1982); Dufossé and Uçar (2016); Dufossé et al. (2018). The upper bound originally is based on the Marcus-Ree theorem in Marcus and Ree (1959). The lower bound in the context of *SecureHull* can be inflated by adding fake rows into the original permutation matrix; thereby increasing the security. This comes with a trade-off of increase in communication. It is also known that finding a Birkhoff-vin Neumann decomposition with the least possible number of permutations is NP-Hard.

5.7. Inner-product

5.7.1. NON-MALICIOUS CLIENT

We first consider the setting where the client is non-malicious and genuinely would like to obtain the list of inner products with respect to the servers data while protecting the privacy of its data. In this setting, the client for represents its data record using d splinters as

$$\mathbf{x} = \sum_{i}^{d} \alpha_i \mathbf{z_i}, \forall i \in 1..d$$

The α_i's are private to the client and not shared with any other entity, be it another client or a server. The splinters $\mathbf{z_i}$ are shared with the server. The scheme for generation of splinters is described in sections 3,4 and 5 along with hardness of recovery guarantees for the clients privacy.

The server returns the inner products of $\langle \mathbf{z_i}, \mathbf{y_k} \rangle \forall \mathbf{z_i} \in \{\mathbf{z_1}, \mathbf{z_2} \ldots \mathbf{z_d}\}$ and $\forall \mathbf{y_k} \in \mathcal{B}$. The client obtains the inner product for any $\mathbf{x}, \mathbf{y_k}$ as

$$\sum_{i}^{d} \alpha_i \langle \mathbf{z_i}, \mathbf{y_k} \rangle$$

The client uses a different set of splinters for every single query with the server.

5.7.2. MALICIOUS CLIENT

In the case where the client is malicious and intends to recover all the vectors in the private database, the setting is equivalent to the well known problem of *learning mixtures of linear regressions* (Mazumdar and Pal, 2020; Yin et al., 2018; Chen et al., 2020; Li and Liang, 2018) which is a generalization of the compressed sensing problem when the cardinality, $|\mathcal{B}| > 1$ as in any typical database.

5.7.3. STATEMENT OF LEARNING MIXTURE OF SPARSE LINEAR REGRESSIONS PROBLEM

Given $L = |\mathcal{B}|$ unknown distinct vectors $\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_L} \in \mathbb{R}^p$ and each is t-sparse meaning that the number of non-zero entries in each $\mathbf{y_i}$ is at most t where it is a known parameter. We define an oracle on the server which, when queried with a splinter $\hat{\mathbf{z}}_i$, returns the noisy output $\langle \hat{\mathbf{z}}_\mathbf{i}, \mathbf{y_k} \rangle + \eta$ where η is a random variable with an expectation of 0 and y_k is uniformly chosen from the $\mathcal{B}$. The goal is to recover all vectors in $\mathbf{B}$ (i.e, data of the server) by making a set of queries $\hat{\mathbf{z}}_\mathbf{1}, \hat{\mathbf{z}}_\mathbf{2}, ..., \hat{\mathbf{z}}_\mathbf{m}$ to the server.

Therefore the malicious client can recover the servers dataset from the resulting inner products by sending rows of a Vandermonde matrix to the server as its splinters as described in Krishnamurthy et al. (2019). The recovery guarantees provided there are as follows. The malicious client can estimate the servers data with $O(t \log^3 n \exp(\sigma/\epsilon)^2/3)$ number of vandemonde samples where ϵ is the precision(in terms of number of digits of servers data vector) and σ is the variance of η. If there is no noise, then the malicious client can recover all of servers data with $2tL\log(tL)$ and the recovery is guaranteed with at least $1-(3/t)$ probability. Under a case of no sparsity, we can change k to p in that bound. Where p is the size of entries in each data record on the server.

5.8. Preliminaries for stochastic splinters: Information theoretic and RKHS distances

Definition 7 *Differential entropy of multivariate Gaussian:*

$$\begin{aligned} h(f) &= -\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} f(\mathbf{x})\ln f(\mathbf{x})\,d\mathbf{x}, \\ &= \frac{1}{2}\ln\left(|(2\pi e)\,\mathbf{\Sigma}|\right) = \frac{1}{2}\ln\left((2\pi e)^k|\mathbf{\Sigma}|\right) = \frac{k}{2}\ln(2\pi e) + \frac{1}{2}\ln(|\mathbf{\Sigma}|) = \frac{k}{2} + \frac{k}{2}\ln(2\pi) + \frac{1}{2}\ln(|\mathbf{\Sigma}|) \end{aligned} \tag{16}$$

where the bars denote the matrix determinant and k is the dimensionality of the vector space.

Definition 8 *Information Density: Given a pair of realizations (s, x) of $(S, X) \sim P_{S,X}$, the information density between s and x is defined as*

$$i(s;x) = log\frac{P_{S,X}(s,x)}{P_S(s)P_X(x)} = \log\frac{P_{X|S}(x|s)}{P_X(x)}$$

Definition 9 *Distance Covariance Székely et al. (2007), α-dCov: Distance covariance between random variables $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{y} \in \mathbb{R}^m$ with finite first moments is a nonnegative number given by*

$$\nu^2(\mathbf{x},\mathbf{y}) = \int_{\mathbb{R}^{d+m}} |f_{\mathbf{x},\mathbf{y}}(t,s) - f_\mathbf{x}(t)f_\mathbf{y}(s)|^2 w(t,s)dtds$$

where $f_\mathbf{x}, f_\mathbf{y}$ are characteristic functions of $\mathbf{x}, \mathbf{y}$, $f_{\mathbf{x},\mathbf{y}}$ is the joint characteristic function, and $w(t,s)$ is a weight function defined as

$$w(t,s) = (C(p,\alpha)C(q,\alpha)|t|_p^{\alpha+p}|s|_q^{\alpha+q})^{-1}$$

with $C(d,\alpha) = \frac{2\pi^{d/2}\Gamma(1-\alpha/2)}{\alpha 2^\alpha \Gamma((\alpha+d)/2)}$.

The distance covariance is zero if and only if random variables $\mathbf{x}$ and $\mathbf{y}$ are independent. From above definition of distance covariance, we have the following expression for Distance Correlation:

Definition 10 *Distance Correlation Székely et al. (2007) (α-dCorr): The squared Distance Correlation between random variables $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{y} \in \mathbb{R}^m$ with finite first moments is a nonnegative number defined as*

$$\rho^2(\mathbf{x},\mathbf{y}) = \begin{cases} \frac{\nu^2(\mathbf{x},\mathbf{y})}{\sqrt{\nu^2(\mathbf{x},\mathbf{x})\nu^2(\mathbf{y},\mathbf{y})}}, & \nu^2(\mathbf{x},\mathbf{x})\nu^2(\mathbf{y},\mathbf{y}) > 0. \\ 0, & \nu^2(\mathbf{x},\mathbf{x})\nu^2(\mathbf{y},\mathbf{y}) = 0. \end{cases}$$

The Distance Correlation defined above has the following interesting properties; 1) $\rho^2(\mathbf{x},\mathbf{x})$ is defined for arbitrary dimensions of $\mathbf{x}$ and $\mathbf{y}$, 2) $\rho^2(\mathbf{x},\mathbf{y}) = 0$ if and only if $\mathbf{x}$ and $\mathbf{y}$ are independent, and 3) $\rho^2(\mathbf{x},\mathbf{y})$ satisfies the relation $0 \leq \rho^2(\mathbf{x},\mathbf{y}) \leq 1$. In our work, we use α-Distance Covariance with $\alpha = 2$ and in the following paper for simplicity just refer to it as Distance Correlation.

We define sample version of distance covariance given i.i.d. samples $\{(\mathbf{x}_k,\mathbf{y}_k)|k = 1,2,\ldots,n\}$ sampled from joint distribution of random vectors $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{y} \in \mathbb{R}^m$. To do so, we define two squared Euclidean distance matrices $\mathbf{E_X}$ and $\mathbf{E_Y}$, where each entry $[\mathbf{E_X}]_{k,l} = \mathbf{x}_k - \mathbf{x}_l$ and $[\mathbf{E_Y}]_{k,l} = \mathbf{y}_k - \mathbf{y}_l$ with $k,l \in \{1,2,\ldots,n\}$. We then make their row and column sums zero to obtain $\widehat{\mathbf{E}}_\mathbf{X}$ and $\widehat{\mathbf{E}}_\mathbf{Y}$ respectively by multiplying with a centering matrix **?** $\mathbf{J}$ given by $\mathbf{J} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T$ on both sides as $\widehat{\mathbf{E}}_\mathbf{X} = \mathbf{J}\mathbf{E}_\mathbf{X}\mathbf{J}$ and $\widehat{\mathbf{E}}_\mathbf{Y} = \mathbf{J}\mathbf{E}_\mathbf{Y}\mathbf{J}$ Now the sample distance correlation (for $\alpha = 2$) is hence defined as follows:

Definition 11 *Sample Distance Correlation Székely et al. (2007): Given i.i.d samples $\mathcal{X} \times \mathcal{Y} = \{(\mathbf{x}_k,\mathbf{y}_k)|k = 1,2,3,\ldots,n\}$ and corresponding double centered Euclidean distance matrices $\widehat{\mathbf{E}}_\mathbf{X}$ and $\widehat{\mathbf{E}}_\mathbf{Y}$, the squared sample distance correlation is defined as,*

$$\hat{\nu}^2(\mathbf{X},\mathbf{Y}) = \frac{1}{n^2}\sum_{k,l=1}^{n}[\widehat{\mathbf{E}}_\mathbf{X}]_{k,l}[\widehat{\mathbf{E}}_\mathbf{Y}]_{k,l},$$

and equivalently sample distance correlation is given by

$$\hat{\rho}^2(\mathbf{X},\mathbf{Y}) = \begin{cases} \frac{\hat{\nu}^2(\mathbf{X},\mathbf{Y})}{\sqrt{\hat{\nu}^2(\mathbf{X},\mathbf{X})\hat{\nu}^2(\mathbf{Y},\mathbf{Y})}}, & \hat{\nu}^2(\mathbf{X},\mathbf{X})\hat{\nu}^2(\mathbf{Y},\mathbf{Y}) > 0. \\ 0, & \hat{\nu}^2(\mathbf{X},\mathbf{X})\hat{\nu}^2(\mathbf{Y},\mathbf{Y}) = 0. \end{cases}$$

# of Components	Dimension	Spherical Gaussian	Lower Bound	Reference
1	d	no	$\hat{\Omega}(d^2/\epsilon^2)$	Ashtiani et al. (2017)
1	d	yes	$\hat{\Omega}(d/\epsilon^2)$	Suresh et al. (2014)
k	1	n/a	$\hat{\Omega}(k/\epsilon^2)$	Suresh et al. (2014)
k	d	no	$\hat{\Omega}(kd^2/\epsilon^2)$	Ashtiani et al. (2017)
k	d	yes	$\hat{\Omega}(kd/\epsilon^2)$	Suresh et al. (2014)

Table 1: Lower bound on sample complexities for estimating Gaussian mixtures

6. Generation of splinters with mixture distributions

We study the case where each splinter can be generated from a different Gaussian mixture with m components as, $\mathbf{z}_i \sim \sum_{a=1}^{m}\omega_a^i g^i = \sum_{a=1}^{m}\omega_a^i N\left(x;\mu_\mathbf{a}^\mathbf{i},\mathbf{\Sigma_a^i}\right)$ but with additional structural constraints that we propose in this paper. This turns the problem of generating splinters into a structured distribution learning problem.

Estimation of Gaussian mixtures has been extensively studied in (Kalai et al., 2012; Kannan et al., 2005; Moitra and Valiant, 2010; Dasgupta, 1999; Ge et al., 2015; Sanjeev and Kannan, 2001; Daskalakis and Kamath, 2014). We summarize lower bound on sample complexities of learning Gaussian mixtures in Table 1. These let the client know, as to when it needs to refresh its splinter generating distributions. These bounds only depend on the dimensionality and the number of components in the mixture. Earlier bounds depended on additional terms such as the condition number. One such result is that, for an ϵ-approximation of a p-dimensional Gaussian mixture with m components is $n > \left(\frac{\kappa(z_i).p}{\epsilon\delta}\right)^{c_m}$ samples with probability greater than $1-\delta$ where $\kappa(z_i) = \frac{1}{min(\{w_1,w_2,...,w_k\}\cup\{D_{tv}(F_i,F_j)|i\neq j\})}$ is the condition number of the estimation of the Gaussian mixture. Therefore, when each z_i is sampled with a different Gaussian mixture with many components m such that w_i is a very small probability or if the total variation distance between individual multivariate Gaussian components, F_i, F_j is small, it would lead to a high sample complexity required for any reasonably accurate estimation of the splinter distributions.

7. Decoy splinter distributions

We now propose a novel approach of *DecoySplinter* for further increasing the security of the *MixtureSplinter* scheme. The idea of a decoy splinter is to generate random splinters that are not supposed to be used in order to obtain a proper reconstruction of the dataset. A malicious server would need to set the coeeficients of these splinters to 0, in addition to figuring out the right secret coefficients of the non-decoy splinters.

7.1. Approach

For a decoy splinter to be effective, it needs to be such that it is very hard to statistically distinguish between a decoy splinter and a non-decoy splinter. Given the known probability distribution of $z_i \sim \mathcal{G}$ for a Gaussian mixture, if a decoy Gaussian mixture destribution to sample decoys from as $d_i \sim \mathcal{H}$ is generated, such that the total variation distance $D_{tv}(\mathcal{G},\mathcal{H}) \leq \tau$, then it would require an order of atleast $1/\tau$ samples to have a constant probability of distinguishing between the case that all samples arise from $\mathcal{G}$ or all samples arise from $\mathcal{H}$. The decoy splinter distributions corresponding to every non-decoy splinter distribution g^i can be therefore generated such that $D_{tv}(\mathcal{G},\mathcal{H}) \leq \tau$. Note that the covariance and mean of the corresponding Gaussian mixture of each splinter distribution is, $\Sigma^i_{\mathcal{G}} = \sum_a \Sigma^i_a w^i_a + \sum_a (\mu^i_a)^T \mu^i_a w^i_a - (\mu^i_a)^T \mu^i_a$ and $\mu^i_{\mathcal{G}} = \sum_a \mu^i_a$. By the Pinsker's inequality we have $D_{TV}(\mathcal{G},\mathcal{H}) \leq \sqrt{2 D_{KL}(\mathcal{G},\mathcal{H})}$. Lower bounds on hypothesis testing to distinguish between distributions based on samples are either based on decreasing the KL-divergence or in terms of reducing the total variation distance. Some of these bounds are also referred to as converse bounds. We therefore now propose an approach to learn a Gaussian mixture from which a decoy shadow can be sampled. We do this by ensuring that the learnt Gaussian mixture of the decoy has a KL-divergence less than τ.

7.2. Bruteforce/enumeration attack success probability

The bruteforce/enumeration attack probability of success for guessing the right combination of secret coefficients when there are d non-decoy splinters, $q-d$ decoy splinters, and each coefficient is

a b-bit number is based on a power tower function in the denominator as

$$\frac{1}{2^{b^d} \times \binom{q}{q-d}}$$

and therefore is infintesimally small.

8. Structured Decoy Distribution Learning

We now present our proposed results that help in structured distribution learning of Gaussian mixtures such that the KL-divergence between two mixtures is minimized. This helps in learning distributions to sample decoy splinters from.

Distance correlation-KL divergence separability theorem: We eventually exploit our separability theorem we present below given the fact that KL-divergence between Gaussian mixtures is separable into terms that only depend on the KL-divergence between the multivariate Gaussian components that form the mixture. We can substitute $-D_{KL}(f_a||f_\alpha)$ with distance correlation $\text{DCor}(\Sigma_a^f, \Sigma_\alpha^f)$ and $-D_{KL}(f_a||g_b)$ with $\text{DCor}(\Sigma_a^f, \Sigma_b^g)$ instead based on our Theorem 12 below which shows that optimizing KL-divergence between multivariate Gaussians is equivalent to optimizing distance correlation for the same. This helps make the optimization of learning structured distributions of splinters more tractable as estimation of KL-divergence in high-dimensions is a challenging problem.

Theorem 12 (Separability theorem)

Minimization of distance correlation $\underset{\mathbf{Z}}{\text{argmin}}(\mathbf{X}, \mathbf{Z})$ *with respect to* $\mathbf{Z}$ *maximizes the Kullback-Leibler divergence,* $KL(\mathbf{X}||\mathbf{Z})$ *for* $\mathbf{X} \sim \mathcal{N}(0, \mathbf{\Sigma_X})$ *and* $\mathbf{Z} \sim \mathcal{N}(0, \mathbf{\Sigma_Z})$

Proof Distance correlation can be represented as $\frac{\text{Tr}\,(\mathbf{X^T X Z^T Z})}{\sqrt{\text{Tr}\,(\mathbf{X^T X})^2\,\text{Tr}\,(\mathbf{Z^T Z})^2}}$ Vepakomma et al. (2018). For covariance matrices $\mathbf{\Sigma_X} = \mathbf{X^T X}$ and $\mathbf{\Sigma_Z} = \mathbf{Z^T Z}$ we have

$$\begin{aligned} \det[(\mathbf{X^T X})^2]\det([\mathbf{Z^T Z}])^2 &\leq \text{Tr}\,(\mathbf{X^T X Z^T Z}) \\ &\leq \sqrt{\text{Tr}\,(\mathbf{X^T X})^2\,\text{Tr}\,(\mathbf{Z^T Z})^2} \end{aligned} \tag{17}$$

by arithmetic-geometric mean inequality for the lower bound and Cauchy-Schwartz inequality for the upper bound on distance covariance $\text{Tr}\,(\mathbf{X^T X Z^T Z})$. $\log\det(\mathbf{Z^T Z})$ is the differential entropy $h(\mathbf{Z})$ upto a constant for multivariate Gaussians. Similarly, the joint entropy $h(\mathbf{X}, \mathbf{Z})$ is given by $\log\det(\mathbf{\Sigma})$ where $\mathbf{\Sigma} = \left[\begin{array}{c|c} \mathbf{X^T X} & \mathbf{X^T Z} \\ \hline \mathbf{Z^T X} & \mathbf{Z^T Z} \end{array}\right]$ Kullback-Leibler divergence is defined using joint entropy and entropy as $h(\mathbf{X}||\mathbf{Z}) = h(\mathbf{X}, \mathbf{Z}) - h(\mathbf{X})$. By Fischer's inequality, we have

$$\det(\mathbf{\Sigma}) \leq \det(\mathbf{X^T X})\det(\mathbf{Z^T Z})$$

As $\det(\mathbf{X^T X})$ is fixed and $\det(\mathbf{Z^T Z})$ decreases with decrease in distance covariance, an increase of $h(\mathbf{X}||\mathbf{Z})$ is only possible when $h(\mathbf{X}, \mathbf{Z}) = \log\det(\mathbf{\Sigma})$ increases which is inturn only possible when $\text{Tr}(\mathbf{X^T Z})$ decreases. Thereby minimizing sum of distance covariance and $\text{Tr}(\mathbf{X^T Z})$ maximizes the Kullback-Leibler divergence in the direction stated above while it also minimizes differential entropy $\det(\mathbf{Z^T Z})$. ■

8.1. Bounds on KL-Divergence between two Gaussian Mixtures

For the distribution learning problem motivated in the previous section, the key is to be able to learn a τ-close Gaussian mixture to a given target Gaussian mixture. We therefore share some results on KL-divergences between Gaussian mixtures Durrieu et al. (2012). Let f and g be two PDFs in $\mathbb{R}^d$, where d is the dimension of the observed vectors x. The KL-divergence between f and g is defined as:

$$D_{KL}(f||g) = \int_{\mathbb{R}^d} f(x) \log \frac{f(x)}{g(x)} dx \tag{18}$$

When f and g are PDFS of multivariate normals:

$$D_{KL}(f||g) = \frac{1}{2} \log \frac{|\Sigma^g|}{|\Sigma^f|} + \frac{1}{2} \text{Tr}((\Sigma^g)^{-1}\Sigma^f) + \frac{1}{2}(\mu^f - \mu^g)^T(\Sigma^g)^{-1}(\mu^f - \mu^g) - \frac{d}{2} \tag{19}$$

When f and g are PDFs for GMMs, the expression for f becomes (with an analogous expression for g):

$$f(x) = \sum_{a=1}^{A} \omega_a^f f_a(x) = \sum_{a=1}^{A} \omega_a^f N\left(x; \mu_a^f, \Sigma_a^f\right) \tag{20}$$

Durrieu and Thiran define the bounds for KL-Divergence between GMMs to be:

$$D_{\text{lower}}(f||g) = \sum_a \omega_a^f \log \frac{\sum_\alpha \omega_\alpha^f e^{-D_{KL}(f_a||f_\alpha)}}{\sum_b \omega_b^g t_{ab}} - \sum_a \omega_a^f H(f_a) \tag{21}$$

$$D_{\text{upper}}(f||g) = \sum_a \omega_a^f \log \frac{\sum_\alpha \omega_\alpha^f z_{a\alpha}}{\sum_b \omega_b^g e^{-D_{KL}(f_a||g_b)}} + \sum_a \omega_a^f H(f_a) \tag{22}$$

where $H(f_a)$ is the entropy of f_a, and the normalization constants of the product of the individual Gaussians are given by:

$$\log t_{ab} = -\frac{d}{2} \log 2\pi - \frac{1}{2} \log |\Sigma_a^f + \Sigma_b^g| - \frac{1}{2}(\mu_b^g - \mu_a^f)^T(\Sigma_a^f + \Sigma_b^g)^{-1}(\mu_b^g - \mu_a^f) \tag{23}$$

$$\log z_{a\alpha} = -\frac{d}{2} \log 2\pi - \frac{1}{2} \log |\Sigma_a^f + \Sigma_\alpha^f| - \frac{1}{2}(\mu_\alpha^f - \mu_a^f)^T(\Sigma_a^f + \Sigma_\alpha^f)^{-1}(\mu_\alpha^f - \mu_a^f) \tag{24}$$

We will focus on optimizing the following average of the lower and upper bounds of the KL-Divergence between GMMs as it was shown to be a good estimate of the KL-Divergence between GMMs in Durrieu et al. (2012).

$$\begin{aligned}
D_{\text{avg}}(f||g) &= \frac{1}{2}\left(D_{\text{lower}}(f||g) + D_{\text{upper}}(f||g)\right) \\
&= \frac{1}{2}\left(\sum_a \omega_a^f \log \frac{\sum_\alpha \omega_\alpha^f e^{-D_{KL}(f_a||f_\alpha)}}{\sum_b \omega_b^g t_{ab}} - \sum_a \omega_a^f H(f_a)\right) \\
&+ \frac{1}{2}\left(\sum_a \omega_a^f \log \frac{\sum_\alpha \omega_\alpha^f z_{a\alpha}}{\sum_b \omega_b^g e^{-D_{KL}(f_a||g_b)}} + \sum_a \omega_a^f H(f_a)\right) \\
&= \frac{1}{2}\sum_a \omega_a^f \log \frac{\sum_\alpha \omega_\alpha^f e^{-D_{KL}(f_a||f_\alpha)}}{\sum_b \omega_b^g t_{ab}} + \frac{1}{2}\sum_a \omega_a^f \log \frac{\sum_\alpha \omega_\alpha^f z_{a\alpha}}{\sum_b \omega_b^g e^{-D_{KL}(f_a||g_b)}}
\end{aligned}$$

$$D_{\text{avg}}(f||g) = \frac{1}{2}\sum_a \omega_a^f \left[\log \sum_\alpha \omega_\alpha^f e^{-D_{KL}(f_a||f_\alpha)} + \log \sum_\alpha \omega_\alpha^f z_{a\alpha} - \log \sum_b \omega_b^g t_{ab} - \log \sum_b \omega_b^g e^{-D_{KL}(f_a||g_b)}\right]$$

9. Modified EM algorithm for our structured distribution learning problem

Therefore we have the following objective that needs to be optimized instead.

$$\frac{1}{2}\sum_a \omega_a^f \left[\log\sum_\alpha \omega_\alpha^f e^{\text{DCov}(\Sigma_a^f,\Sigma_\alpha^f)} + \log\sum_\alpha \frac{\omega_\alpha^f}{\sqrt{|\Sigma_a^f+\Sigma_\alpha^f|}} - \log\sum_b \omega_b^g e^{\text{DCov}(\Sigma_a^f,\Sigma_b^g)} - \log\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}\right] \tag{27}$$

Upon invoking the separability theorem in 12, in order to minimize the above average bound on KL-divergence $D_{\text{avg}}(f||g)$ between Gaussian mixtures, the following has to be minimized

$$\frac{1}{2}\sum_a \omega_a^f \left[\log\sum_\alpha \omega_\alpha^f e^{\text{DCov}(\Sigma_a^f,\Sigma_\alpha^f)} + \log\sum_\alpha \frac{\omega_\alpha^f}{\sqrt{|\Sigma_a^f+\Sigma_\alpha^f|}} - \log\sum_b \omega_b^g e^{\text{DCov}(\Sigma_a^f,\Sigma_b^g)} - \log\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}\right] \tag{28}$$

This is fortunately possible because the KL divergence between Gaussian mixtures is expressed via separable terms of KL between components of Gaussian mixtures. Note that two terms are constant in here with respect to the target mixture distribution as follows

$$D_{\text{avg}}(f||g) = \frac{1}{2}\sum_a \omega_a^f \left[C_1 + C_2 - \log\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}} - \log\sum_b \omega_b^g e^{\text{DCov}(\Sigma_f^a,\Sigma_b^g)}\right] \tag{29}$$

With $\Sigma_g^f = \frac{1}{N-1}Z_b^T Z_b$, where N is the number of samples, our problem is equivalent to minimizing the following for each component a

$$\omega_a^f \log\sum_\alpha \omega_\alpha^f e^{\text{DCov}(\Sigma_a^f,\Sigma_\alpha^f)} + \omega_a^f \log\sum_\alpha \frac{\omega_\alpha^f}{\sqrt{|\Sigma_a^f+\Sigma_\alpha^f|}} \tag{30}$$

$$- \omega_a^f \log\sum_b \omega_b^g e^{\text{DCov}(\Sigma_a^f,\frac{1}{N-1}Z_b^T Z_b)} - \omega_a^f \log\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\frac{1}{N-1}Z_b^T Z_b|}} \tag{31}$$

$$= \omega_a^f(C_1+C_2) - \omega_a^f \log\sum_b \omega_b^g e^{\text{DCov}(\Sigma_a^f,\frac{1}{N-1}Z_b^T Z_b)} - \omega_a^f \log\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\frac{1}{N-1}Z_b^T Z_b|}} \tag{32}$$

$$- \omega_a^f \log\sum_b \omega_b^g e^{\text{DCov}\left(\Sigma_a^f,\frac{1}{N-1}(Z_b-\mu_b^g)^T(Z_b-\mu_b^g)\right)} - \omega_a^f \log\sum_b \frac{\omega_b^g e^{-\frac{1}{2}\left(\mu_b^g-\mu_a^f\right)^T\left(\Sigma_a^f+\frac{1}{N-1}(Z_b-\mu_b^g)^T(Z_b-\mu_b^g)\right)^{-1}\left(\mu_b^g-\mu_a^f\right)}}{\sqrt{|\Sigma_a^f+\frac{1}{N-1}(Z_b-\mu_b^g)^T(Z_b-\mu_b^g)|}} \tag{33}$$

$$+ \omega_a^f(C_1+C_2) + \lambda.\text{EMLoss}$$

where the EMLoss in the last term is the standard EM loss. Here, the objective function is regularized with the standard loss used in EM-algorithms for estimating Gaussian mixtures. Therefore we now have a modified EM algorithm that learns Gaussian mixtures with respect to a target distribution while satisfying the closeness constraints with respect to KL-divergence.

10. Modified EM algorithm for structured learning of Gaussian mixtures

E-step updates: For each component b at step t, compute

$$\gamma_{ib}^{(t+1)} = \frac{\omega_b^{g(t)} p\left(y_i|\mu_b^{g(t)}, \Sigma_b^{g(t)}\right)}{\sum_{b'=1}^{B} w_{b'}^{g\,(t)} p\left(y_i|\mu_{b'}^{g\,(t)}, \Sigma_{b'}^{g\,(t)}\right)}, \quad i = 1, \ldots, N$$

and finally

$$n_b^{(t+1)} = \sum_{i=1}^{N} \gamma_{ib}^{(t+1)}$$

M-step updates: For each component b, compute the following update

$$\omega_b^{g(t+1)} = \frac{n_b^{(t+1)}}{N}$$

The rest of updates for the mean vector and covariances are in Appendix F.

Theorem 13
The function $\log \sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f + \Sigma_b^g|}}$ *is convex if*

$$\omega_b^g \sum_b \left(\frac{\omega_b^g}{\sqrt{|\Sigma_a^f + \Sigma_b^g|}} - \omega_b^g \right) \geq 0$$

as this results in a positive semi-definite Hessian.

Proof The proof is in Appendix A. ■

Theorem 14 *The function* $LogSumExp(p) = log(\sum_i (e_i^p))$ *is convex.*

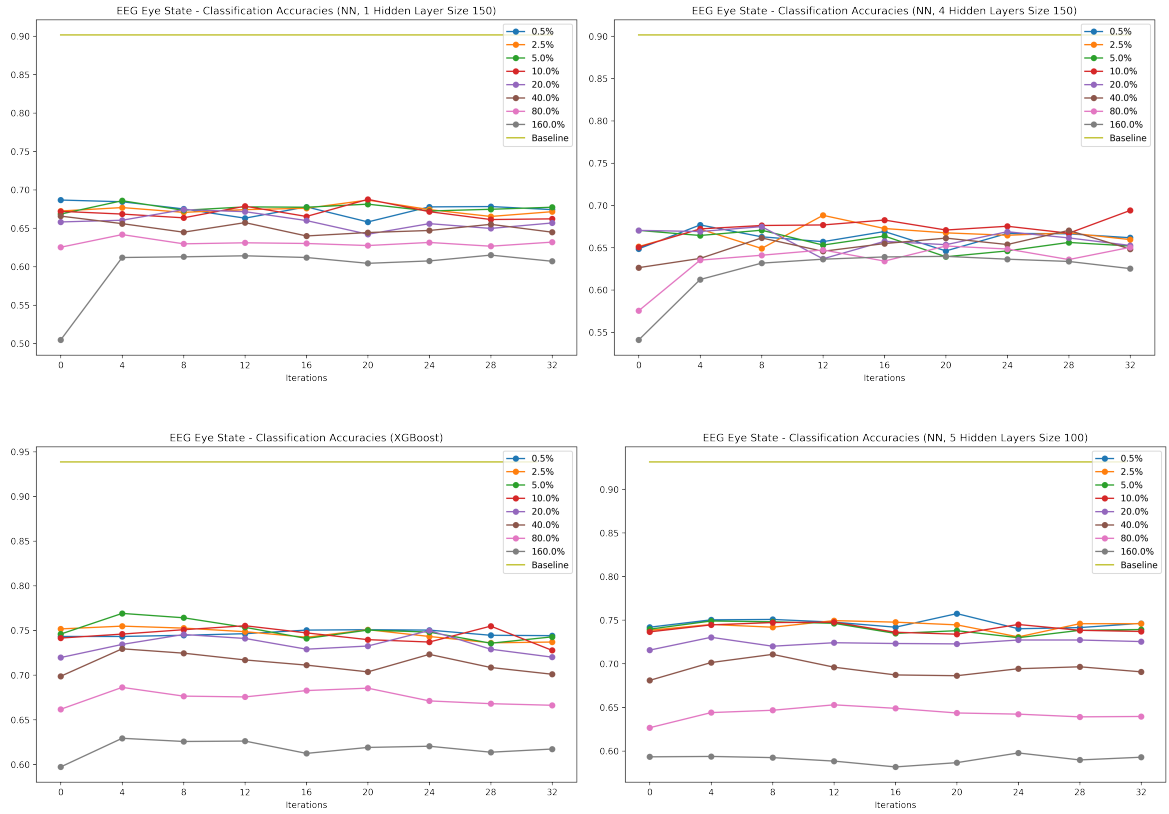
Proof The proof is in Appendix B. ■

11. Upper and lower bounds on distance correlation

Theorem 15 (Upper bound) *Given a map* $h : \mathbf{X} \mapsto \mathbf{Z}$*, the population distance covariance can be upper bounded as follows based on the Lipschitz constant of the map* f *denoted by* L*, and total variance of* $\mathbf{X}$ *as*

$$DCOV(p_{xy}, \mathcal{F}, \mathcal{G}) \lessapprox \sqrt{\frac{log(6/\delta)}{0.24n}} + \frac{C}{n} + L.[Tr(\Sigma_{\mathbf{X}})]^2 \tag{34}$$

at least with probability $1 - \delta$.
Proof The proof is in Appendix C



Theorem 16 (Lower bound) *A lower bound on sample distance covariance can be given based the following determinants as*

$$\det(\mathbf{Z^T X}) - \det(\mathbf{Z^T Z}) - \det(\mathbf{X^T Z}) + \det(\mathbf{X^T X}) - \|\mathbf{Z}\| - \|\mathbf{X} - \mathbf{Z}\| \leq DCOV(\mathbf{X}, \mathbf{Z}) \quad (35)$$

Proof The proof is in Appendix D ■

Theorem 17 *Under Gaussianity assumptions on* $\mathbf{Z}$*, minimizing* $DCOV(\mathbf{X_j}, \mathbf{Z})$ *is equivalent to minimizing mutual information* $MI(\mathbf{X_j}, \mathbf{Z})$*. This is also equivalent to maximizing the error in linear regression with respect to the attributes* $\mathbf{Z}$ *instead of coefficients as*

$$\underset{\mathbf{Z}}{\operatorname{argmax}} \ \|\mathbf{X_j} - \mathbf{Z}\beta\| \ \ s.t, \ \beta = (\mathbf{Z^T Z})^{-1}\mathbf{Z^T X_j}$$

Proof The proof is in Appendix E. ■

12. Numerical Experiments

We performed numerical experiments on 3 UCI-ML repository datasets of EEG eye state, occupancy and Avila. We show in a series of captioned figures below that the well-tuned classification

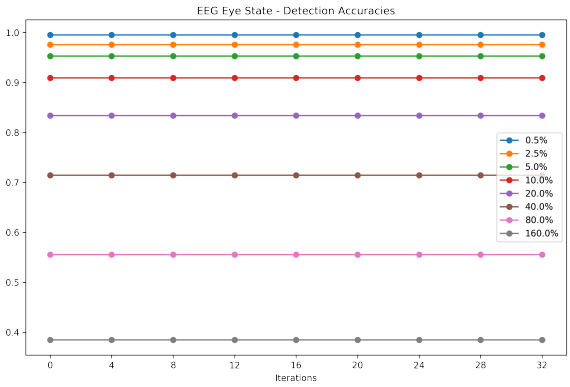


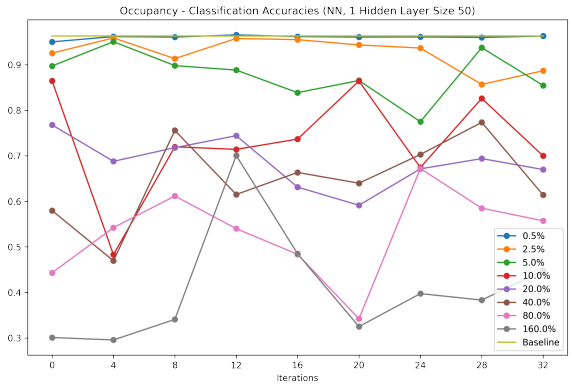
Figure 2: EEG: Classification of decoy Vs. non-decoy splinters using NN's, XGBoost and Random Forest shows that the models are unable to distinguish them when the sample size of decoy splinters is twice that of the non-decoy splinters. Our pipeline is a standard one used in data-poisoning schemes with two models; one to detect and one to classify. We obtain similar results upon using anomaly detectors such as isolation forests as well. The pipeline consists of a model to detect a decoy Vs. non-decoy and in addition we also perform a label reconstruction attack to reconstruct the raw labels of the client. The splinters are generated only using raw features. We see a spin-off empirical benefit that upon adding decoy splinters, not only do we prevent their detection; but we also make it extremely hard for the attacker to be able to reconstruct the raw labels corresponding to the raw data; via a second model.

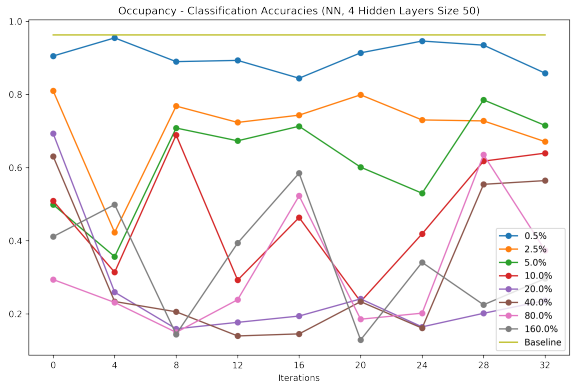
models such as neural networks with increasing hidden layers of 1, 4 , 8 and 12 as well as models such as XGBoost and Random Forests cannot distinguish between the real and decoy splinters generated by our scheme, thereby making it really hard for an attacker that is dependent on machine learning to estimate the pair of mixture distributions used to sample the real and decoy splinters. Our pipeline consists of a model to detect a decoy Vs. non-decoy and in addition we also perform a label reconstruction attack to reconstruct the raw labels of the client. The splinters are generated only using raw features. We see a spin-off empirical benefit that upon adding decoy splinters, not only do we prevent their detection; but we also make it extremely hard for the attacker to be able to reconstruct the raw labels corresponding to the raw data; via a second model.

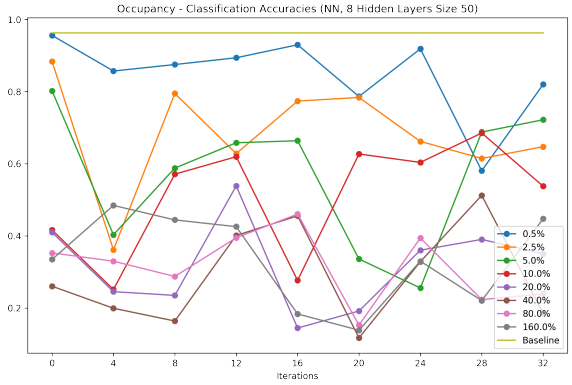
We also visualize example splinters including the decoy and real non-decoy splinters along with their reconstructions in Figure 1 upon choosing the right secret coefficients albeit at varying number of floating point precisions. We see that in this particular example the real image cannot be reconstructed back unless the coefficients were correctly chosen upto 4 decimal point places. The 2 by 2 images of face reconstructions are for exactly chosen secret coefficients upto 2,3, 4 and 5 decimal places. We use default SciPy parameters for powell minimization to optimize mu and parameters of `ftol = 0.001, xtol = 0.001, maxfev = 4000` for optimizing $\mathbf{Z_b}$.

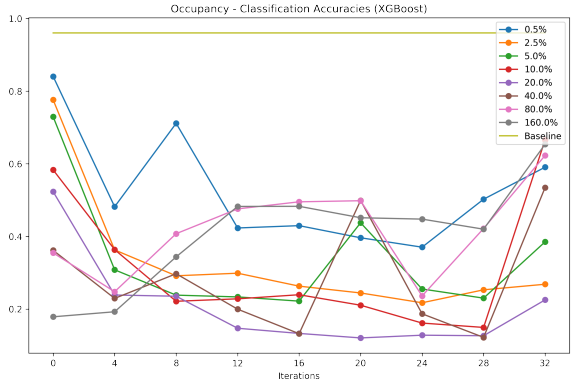
13. Conclusion

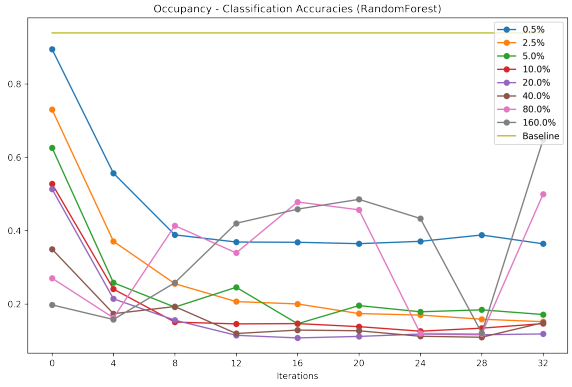
We provide a new scheme for secure computation called splintering that is well suited for distributed machine learning given its efficiency with respect to the resources of compute and bandwidth. We share various theoretical guarantees and practical insights of our approach. We would also like to

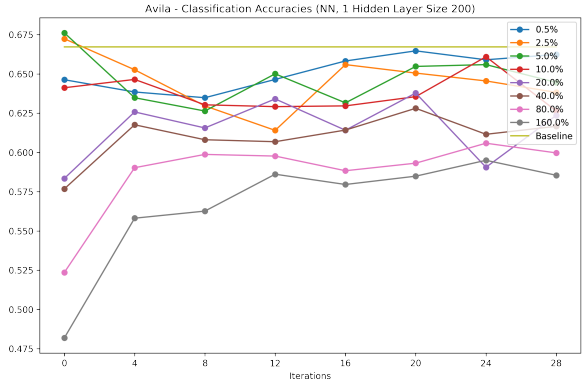

Occupancy - Classification Accuracies (NN, 1 Hidden Layer Size 50)
Iterations
0.5%
2.5%
5.0%
10.0%
20.0%
40.0%
80.0%
160.0%
Baseline

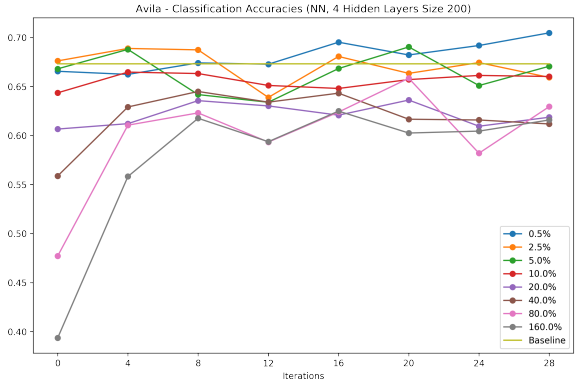

Occupancy - Classification Accuracies (NN, 4 Hidden Layers Size 50)
Iterations
0.5%
2.5%
5.0%
10.0%
20.0%
40.0%
80.0%
160.0%
Baseline

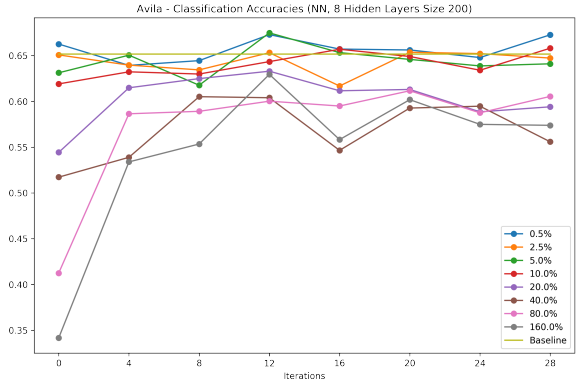

Occupancy - Classification Accuracies (NN, 8 Hidden Layers Size 50)
Iterations
0.5%
2.5%
5.0%
10.0%
20.0%
40.0%
80.0%
160.0%
Baseline

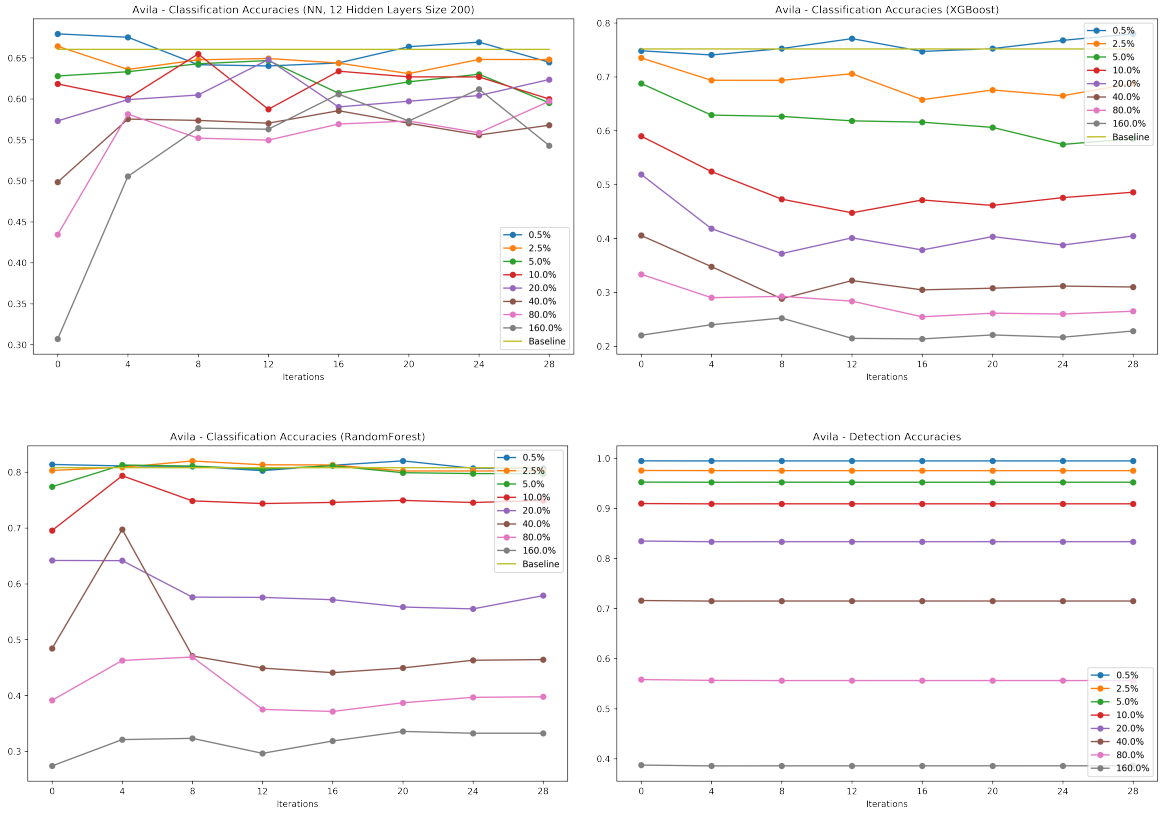

Occupancy - Classification Accuracies (XGBoost)
Iterations
0.5%
2.5%
5.0%
10.0%
20.0%
40.0%
80.0%
160.0%
Baseline


Occupancy - Classification Accuracies (RandomForest)
Iterations
0.5%
2.5%
5.0%
10.0%
20.0%
40.0%
80.0%
160.0%
Baseline


Avila - Classification Accuracies (NN, 1 Hidden Layer Size 200)
Iterations
0.5%
2.5%
5.0%
10.0%
20.0%
40.0%
80.0%
160.0%
Baseline


Avila - Classification Accuracies (NN, 4 Hidden Layers Size 200)
Iterations
0.5%
2.5%
5.0%
10.0%
20.0%
40.0%
80.0%
160.0%
Baseline


Avila - Classification Accuracies (NN, 8 Hidden Layers Size 200)
Iterations
0.5%
2.5%
5.0%
10.0%
20.0%
40.0%
80.0%
160.0%
Baseline



Occupation and Avila datasets: Classification of decoy Vs. non-decoy splinters using NN's, XGBoost and Random Forest shows that the models are unable to distinguish them when the sample size of decoy splinters is twice that of the non-decoy splinters. Our pipeline is a standard one used in data-poisoning schemes with two models; one to detect and one to classify. Our pipeline consists of a model to detect a decoy Vs. non-decoy and in addition we also perform a label reconstruction attack to reconstruct the raw labels of the client. The splinters are generated only using raw features. We see a spin-off empirical benefit that upon adding decoy splinters, not only do we prevent their detection; but we also make it extremely hard for the attacker to be able to reconstruct the raw labels corresponding to the raw data; via a second model.

extend this scheme to other useful private operations and thereby build a toolbox for splintering based computational pipelines. We find our approach to be particularly suitable for privatizing training and inference on split learning a popular distributed machine learning technique.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, 2016.

Hassan Ashtiani, Shai Ben-David, Nick Harvey, Christopher Liaw, Abbas Mehrabian, and Yaniv Plan. Near-optimal sample complexity bounds for robust learning of gaussians mixtures via compression schemes. *arXiv preprint arXiv:1710.05209*, 2017.

Mikhail J Atallah and Wenliang Du. Secure multi-party computational geometry. In *Workshop on Algorithms and Data Structures*, pages 165–179. Springer, 2001.

Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: a system for secure multi-party computation. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 257–266, 2008.

Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, et al. Secure multiparty computation goes live. In *International Conference on Financial Cryptography and Data Security*, pages 325–343. Springer, 2009.

Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Annual cryptology conference*, pages 505–524. Springer, 2011.

Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on Computing*, 43(2):831–871, 2014.

Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

Richard A Brualdi. Notes on the birkhoff algorithm for doubly stochastic matrices. *Canadian Mathematical Bulletin*, 25(2):191–199, 1982.

Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 639–648, 1996.

Konstantinos Chatzikokolakis, Miguel E Andrés, Nicolás Emilio Bordenabe, and Catuscia Palamidessi. Broadening the scope of differential privacy using metrics. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 82–102. Springer, 2013.

Sitan Chen, Jerry Li, and Zhao Song. Learning mixtures of linear regressions in subexponential time via fourier moments. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 587–600, 2020.

Ronald Cramer, Ivan Damgård, and Jesper B Nielsen. Multiparty computation from threshold homomorphic encryption. In *International conference on the theory and applications of cryptographic techniques*, pages 280–300. Springer, 2001.

Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. *Secure multiparty computation*. Cambridge University Press, 2015.

Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*, pages 643–662. Springer, 2012.

Sanjoy Dasgupta. Learning mixtures of gaussians. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 634–644. IEEE, 1999.

Constantinos Daskalakis and Gautam Kamath. Faster and sample near-optimal algorithms for proper learning mixtures of gaussians. In *Conference on Learning Theory*, pages 1183–1213, 2014.

Fanny Dufossé and Bora Uçar. Notes on birkhoff–von neumann decomposition of doubly stochastic matrices. *Linear Algebra and its Applications*, 497:108–115, 2016.

Fanny Dufossé, Kamer Kaya, Ioannis Panagiotas, and Bora Uçar. Further notes on birkhoff–von neumann decomposition of doubly stochastic matrices. *Linear Algebra and its Applications*, 554: 68–78, 2018.

J-L Durrieu, J-Ph Thiran, and Finnian Kelly. Lower and upper bounds for approximation of the kullback-leibler divergence between gaussian mixture models. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4833–4836. Ieee, 2012.

Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

Cynthia Dwork and Adam Smith. Differential privacy for statistics: What we know and what we want to learn. *Journal of Privacy and Confidentiality*, 1(2), 2010.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Differential privacy—a primer for the perplexed,". In *Conf. of European Statisticians, Joint UNECE/Eurostat work session on statistical data confidentiality*, 2011.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Dominic Edelmann, Donald Richards, and Daniel Vogel. The distance standard deviation. *arXiv preprint arXiv:1705.05777*, 2017.

David Evans, Vladimir Kolesnikov, and Mike Rosulek. A pragmatic introduction to secure multiparty computation. *Foundations and Trends® in Privacy and Security*, 2(2-3), 2017.

David Evans, Vladimir Kolesnikov, Mike Rosulek, et al. A pragmatic introduction to secure multiparty computation. *Foundations and Trends® in Privacy and Security*, 2(2-3):70–246, 2018.

Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2012:144, 2012.

Caroline Fontaine and Fabien Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP Journal on Information Security*, 2007:1–10, 2007.

Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure mpc from indistinguishability obfuscation. In *Theory of Cryptography Conference*, pages 74–94. Springer, 2014.

Rong Ge, Qingqing Huang, and Sham M Kakade. Learning mixtures of gaussians in high dimensions. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 761–770, 2015.

Craig Gentry and Dan Boneh. *A fully homomorphic encryption scheme*, volume 20. Stanford university Stanford, 2009.

Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 129–148. Springer, 2011.

Joseph Geumlek, Shuang Song, and Kamalika Chaudhuri. Renyi differential privacy mechanisms for posterior sampling. In *Advances in Neural Information Processing Systems*, pages 5289–5298, 2017.

Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78, 1998.

Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer, 2005.

Xi He, Ashwin Machanavajjhala, and Bolin Ding. Blowfish privacy: Tuning privacy-utility tradeoffs using policies. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1447–1458, 2014.

Martin Hirt, Ueli Maurer, and Bartosz Przydatek. Efficient secure multi-party computation. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 143–161. Springer, 2000.

Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 21–30, 2007.

Adam Tauman Kalai, Ankur Moitra, and Gregory Valiant. Disentangling gaussians. *Communications of the ACM*, 55(2):113–120, 2012.

Ravindran Kannan, Hadi Salmasian, and Santosh Vempala. The spectral method for general mixture models. In *International Conference on Computational Learning Theory*, pages 444–457. Springer, 2005.

Daniel Kifer and Ashwin Machanavajjhala. A rigorous and customizable framework for privacy. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pages 77–88, 2012.

Daniel Kifer and Ashwin Machanavajjhala. Pufferfish: A framework for mathematical privacy definitions. *ACM Transactions on Database Systems (TODS)*, 39(1):1–36, 2014.

Fragkiskos Koufogiannis. Privacy in multi-agent and dynamical systems. *UPenn PhD Dissertation*, 2017.

Fragkiskos Koufogiannis and George J Pappas. Location-dependent privacy. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 7586–7591. IEEE, 2016.

Fragkiskos Koufogiannis, Shuo Han, and George J Pappas. Gradual release of sensitive data under differential privacy. *arXiv preprint arXiv:1504.00429*, 2015a.

Fragkiskos Koufogiannis, Shuo Han, and George J Pappas. Optimality of the laplace mechanism in differential privacy. *arXiv preprint arXiv:1504.00065*, 2015b.

Akshay Krishnamurthy, Arya Mazumdar, Andrew McGregor, and Soumyabrata Pal. Sample complexity of learning mixture of sparse linear regressions. In *Advances in Neural Information Processing Systems*, pages 10532–10541, 2019.

Yuanzhi Li and Yingyu Liang. Learning mixtures of linear regressions with nearly optimal complexity. *arXiv preprint arXiv:1802.07895*, 2018.

Yehida Lindell. Secure multiparty computation for privacy preserving data mining. In *Encyclopedia of Data Warehousing and Mining*, pages 1005–1009. IGI Global, 2005.

Ashwin Machanavajjhala and Daniel Kifer. Designing statistical privacy for your data. *Communications of the ACM*, 58(3):58–67, 2015.

Ashwin Machanavajjhala, Daniel Kifer, John Abowd, Johannes Gehrke, and Lars Vilhuber. Privacy: Theory meets practice on the map. In *2008 IEEE 24th international conference on data engineering*, pages 277–286. IEEE, 2008.

Marvin Marcus and Rimhak Ree. Diagonals of doubly stochastic matrices. *The Quarterly Journal of Mathematics*, 10(1):296–302, 1959.

Arya Mazumdar and Soumyabrata Pal. Recovery of sparse signals from a mixture of linear samples. *arXiv preprint arXiv:2006.16406*, 2020.

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103. IEEE, 2007.

Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.

Ankur Moitra and Gregory Valiant. Settling the polynomial learnability of mixtures of gaussians. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 93–102. IEEE, 2010.

Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124, 2011.

Tingyuan Nie, Chuanwang Song, and Xulong Zhi. Performance evaluation of des and blowfish algorithms. In *2010 International Conference on Biomedical Engineering and Computer Science*, pages 1–4. IEEE, 2010.

Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84, 2007.

Kobbi Nissim, Rann Smorodinsky, and Moshe Tennenholtz. Approximately optimal mechanism design via differential privacy. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 203–213, 2012.

Kobbi Nissim, Thomas Steinke, Alexandra Wood, Micah Altman, Aaron Bembenek, Mark Bun, Marco Gaboardi, David R O'Brien, and Salil Vadhan. Differential privacy: A primer for a non-technical audience. In *Privacy Law Scholars Conf*, 2017.

Arora Sanjeev and Ravi Kannan. Learning mixtures of arbitrary gaussians. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 247–257, 2001.

Sai Sri Sathya, Praneeth Vepakomma, Ramesh Raskar, Ranjan Ramachandra, and Santanu Bhattacharya. A review of homomorphic encryption libraries for secure computation. *arXiv preprint arXiv:1812.02428*, 2018.

Nigel P Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *International Workshop on Public Key Cryptography*, pages 420–443. Springer, 2010.

Shuang Song, Yizhen Wang, and Kamalika Chaudhuri. Pufferfish privacy mechanisms for correlated data. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1291–1306, 2017.

Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 377–394. Springer, 2010.

Ananda Theertha Suresh, Alon Orlitsky, Jayadev Acharya, and Ashkan Jafarpour. Near-optimal-sample estimators for spherical gaussian mixtures. In *Advances in Neural Information Processing Systems*, pages 1395–1403, 2014.

Gábor J Székely, Maria L Rizzo, Nail K Bakirov, et al. Measuring and testing dependence by correlation of distances. *The annals of statistics*, 35(6):2769–2794, 2007.

Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 24–43. Springer, 2010.

Praneeth Vepakomma, Chetan Tonde, Ahmed Elgammal, et al. Supervised dimensionality reduction via distance correlation maximization. *Electronic Journal of Statistics*, 12(1):960–984, 2018.

Yu-Xiang Wang, Borja Balle, and Shiva Kasiviswanathan. Subsampled r\'enyi differential privacy and analytical moments accountant. *arXiv preprint arXiv:1808.00087*, 2018.

Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, Washington, DC, USA, 1982a. IEEE Computer Society. doi: 10.1109/SFCS.1982.88. URL https://doi.org/10.1109/SFCS.1982.88.

Andrew C Yao. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*, pages 160–164. IEEE, 1982b.

Dong Yin, Ramtin Pedarsani, Yudong Chen, and Kannan Ramchandran. Learning mixtures of sparse linear regressions using sparse graph codes. *IEEE Transactions on Information Theory*, 65(3):1430–1451, 2018.

Appendix A. Proof of Theorem 13

Proof

This condition simplifies to requiring

$$\sqrt{|\Sigma_a^f + \Sigma_b^g|} \leq \omega_b^g, \forall b$$

By the arithmetic-geometric-mean (A.G.M) inequality we have,

$$\prod_{k=1}^{n} \lambda_k \leq \frac{1}{n^n} \left(\sum_{k=1}^{n} \lambda_k \right)^n$$

Therefore $\sum_b |\Sigma_a^f + \Sigma_b^g| \leq \frac{\sum_b [\mathrm{Tr}(\Sigma_a^f + \Sigma_b^g)]^n}{n^n}$ This implies that if,

$$\sum_b \mathrm{Tr}(\Sigma_a^f + \Sigma_b^g) \leq n \sqrt[n]{\omega_b^g}, \forall b$$

then the condition for convexity $\sum_b \sqrt{|\Sigma_a^f + \Sigma_b^g|} \leq n \sqrt[n]{\omega_b^g}, \forall b$ will be satisfied. ■

Appendix B. Proof of Theorem 14

Proof We now show that the LogSumExp function $\log \sum_b \omega_b^g e^{\mathrm{DCov}(\Sigma_f^a, \Sigma_b^g)}$ is convex as well. In fact, $LogSumExp(f(z))$ happens to be convex for any convex function $f(z)$ as shown below.

$$\frac{\partial^2}{\partial z^2} log \sum e^{f_i(z)} = \frac{\partial}{\partial z} \left[\frac{\sum (e_i^f(z) \frac{\partial}{\partial z} f_i(z))}{\sum e^{f_i(z)}} \right] \tag{36}$$

which is equal to

$$\frac{\sum e_i^f \frac{\partial^2}{\partial z^2} f_i(z)}{\sum e^{f_i(z)}} + \frac{\sum e^{f^i(z)} [\frac{\partial}{\partial z} f_i(z)]^2}{\sum e^{f_i(z)}} - \frac{(\sum e^{f_i(z)} \frac{\partial}{\partial z} f_i(z))^2}{(\sum e^{f_i(z)})^2} \tag{37}$$

The first term is positive. The difference of the next two terms is positive due to Jensen's inequality as

$$\sum \left[a_i \left(\frac{\partial}{\partial z} f_i(z) \right)^2 \right] \geq \left[\sum a_i \frac{\partial}{\partial z} f_i(z) \right]^2 \tag{38}$$

This proves convexity of $\log \sum_b \omega_b^g e^{\text{DCov}(\Sigma_f^a, \Sigma_b^g)}$.

■

Appendix C. Proof of Theorem 15

Proof Based on the definition of Lipschitz continuity we have the following bound where L is the Lipschitz constant of the map that learns $\mathbf{Z}$ from $\mathbf{X}$,

$$\|f(\mathbf{X_i}) - f(\mathbf{X_j})\|^2 = \|\mathbf{Z_i} - \mathbf{Z_j}\|^2 \leq L \|\mathbf{X_i} - \mathbf{X_j}\|^2 \tag{39}$$

Multiplying by $\langle \mathbf{X_i}, \mathbf{X_j} \rangle$ on both sides and summing over all points we have

$$\sum_{ij} \|\mathbf{Z_i} - \mathbf{Z_j}\|^2 \langle \mathbf{X_i}, \mathbf{X_j} \rangle \leq L \sum_{ij} \|\mathbf{X_i} - \mathbf{X_j}\|^2 \langle \mathbf{X_i}, \mathbf{X_j} \rangle$$

Now dividing on both sides by
$\sqrt{\sum_{ij} \|\mathbf{Z_i} - \mathbf{Z_j}\|^2 \langle \mathbf{Z_i}, \mathbf{Z_j} \rangle} \sqrt{\sum_{ij} \|\mathbf{X_i} - \mathbf{X_j}\|^2 \langle \mathbf{X_i}, \mathbf{X_j} \rangle}$ we get

$$DCOR(\mathbf{X}, \mathbf{Z}) \leq \frac{L \sqrt{\sum_{ij} \|\mathbf{X_i} - \mathbf{X_j}\|^2 \langle \mathbf{X_i}, \mathbf{X_j} \rangle}}{\sqrt{\sum_{ij} \|\mathbf{Z_i} - \mathbf{Z_j}\|^2 \langle \mathbf{Z_i} \mathbf{Z_j} \rangle}} \tag{40}$$

But $\frac{\sqrt{\sum_{ij} \|\mathbf{X_i} - \mathbf{X_j}\|^2 \langle \mathbf{X_i}, \mathbf{X_j} \rangle}}{\sqrt{\sum_{ij} \|\mathbf{Z_i} - \mathbf{Z_j}\|^2 \langle \mathbf{Z_i} \mathbf{Z_j} \rangle}}$ is the ratio of distance standard deviations which is the square root of distance variance which is inturn distance covariance between a variable and itself. It has been shown in Edelmann et al. (2017) that the distance standard deviation can be upper bounded by the trace of the covariance matrix. Therefore we have

$$DCOR(\mathbf{X}, \mathbf{Z}) \leq \frac{L.Tr(\Sigma_{\mathbf{X}})}{\sqrt{\sum_{ij} \|\mathbf{Z_i} - \mathbf{Z_j}\|^2 \langle \mathbf{Z_i} \mathbf{Z_j} \rangle}} \tag{41}$$

and similarly

$$DCOV(\mathbf{X}, \mathbf{Z}) \leq L.[Tr(\Sigma_{\mathbf{X}})]^2 \tag{42}$$

Therefore combining our sample SIV inequality with a concentration Hoeffding bound on the quality of estimating population distance covariance from sample distance covariance in Gretton et al. (2005) we get with high-probability $1-\delta$ an updated bound of

$$DCOV(p_{xy},\mathcal{F},\mathcal{G}) \pm \epsilon \leq \sqrt{\frac{log(6/\delta)}{0.24n}} + \frac{C}{n} + L.[Tr(\Sigma_{\mathbf{X}})]^2 \tag{43}$$

■

Appendix D. Proof of Theorem 16

Proof

$$\det(\mathbf{Z^T X}) - \det(\mathbf{Z^T Z}) - \det(\mathbf{X^T Z}) + \det(\mathbf{X^T X})$$

can be bounded using Hadamard's inequality as

$$\begin{aligned}
&\det(\mathbf{Z^T X}) - \det(\mathbf{Z^T Z}) + \det(\mathbf{X^T X}) - \det(\mathbf{X^T Z}) \\
&\leq \left\|\mathbf{Z^T X} - \mathbf{Z^T Z}\right\|_2 \frac{\left\|\mathbf{Z^T X}\right\|_2^n - \left\|\mathbf{Z^T Z}\right\|_2^n}{\left\|\mathbf{Z^T X}\right\|_2 - \left\|\mathbf{Z^T Z}\right\|_2} \\
&+ \left\|\mathbf{X^T Z} - \mathbf{X^T X}\right\|_2 \frac{\left\|\mathbf{X^T Z}\right\|_2^n - \left\|\mathbf{X^T X}\right\|_2^n}{\left\|\mathbf{X^T Z}\right\|_2 - \left\|\mathbf{X^T X}\right\|_2}
\end{aligned}$$

The fractional terms $\frac{\left\|\mathbf{Z^T X}\right\|_2^n - \left\|\mathbf{Z^T X}\right\|_2^n}{\left\|\mathbf{Z^T X}\right\|_2 - \left\|\mathbf{Z^T Z}\right\|_2}$, $\frac{\left\|\mathbf{X^T Z}\right\|_2^n - \left\|\mathbf{X^T X}\right\|_2^n}{\left\|\mathbf{X^T Z}\right\|_2 - \left\|\mathbf{X^T X}\right\|_2}$ can be written as a sum of geometric-series, with factors of change of $\frac{\left\|\mathbf{Z^T X}\right\|}{\left\|\mathbf{Z^T Z}\right\|}$, $\frac{\left\|\mathbf{X^T Z}\right\|}{\left\|\mathbf{X^T X}\right\|}$ respectively because

$$\begin{aligned}
\frac{\left\|\mathbf{Z^T X}\right\|_2^n - \left\|\mathbf{Z^T Z}\right\|_2^n}{\left\|\mathbf{Z^T X}\right\|_2 - \left\|\mathbf{Z^T Z}\right\|_2} &= \frac{1 - (\frac{\left\|\mathbf{Z^T X}\right\|_2}{\left\|\mathbf{Z^T Z}\right\|_2})^n}{1 - \frac{\left\|\mathbf{Z^T X}\right\|_2}{\left\|\mathbf{Z^T Z}\right\|_2}} \\
&= \sum_{p=0}^{n-1} \left\|\mathbf{Z^T X}\right\|_2^p \left\|\mathbf{Z^T Z}\right\|_2^{p-1}
\end{aligned}$$

Therefore these fractional terms can be minimized by minimizing $\left\|\mathbf{Z^T X}\right\|_2$ and $\left\|\mathbf{Z^T Z}\right\|_2$ as the sums of products of decreasing functions of norms are also decreasing. By Cauchy-Schwarz inequality $\left\|\mathbf{Z^T}(\mathbf{X}-\mathbf{Z})\right\| \leq \|\mathbf{Z}\| \, \|\mathbf{X}-\mathbf{Z}\|$.

Therefore the upper-bound on difference of KL-divergence can be minimized by minimizing $\|\mathbf{Z}\|$ and $\|\mathbf{X}-\mathbf{Z}\|$ to minimize terms $\left\|\mathbf{Z^T X} - \mathbf{Z^T Z}\right\|$,$\left\|\mathbf{X^T Z} - \mathbf{X^T X}\right\|$ in addition to minimizing $\left\|\mathbf{Z^T Z}\right\|$, $\left\|\mathbf{Z^T X}\right\|_2 = Tr(\mathbf{Z^T X X^T Z}) = DCOV(\mathbf{X},\mathbf{Z})$ to minimize terms $\frac{\left\|\mathbf{Z^T X}\right\|_2^n - \left\|\mathbf{Z^T X}\right\|_2^n}{\left\|\mathbf{Z^T X}\right\|_2 - \left\|\mathbf{Z^T Z}\right\|_2}$, $\frac{\left\|\mathbf{X^T Z}\right\|_2^n - \left\|\mathbf{X^T X}\right\|_2^n}{\left\|\mathbf{X^T Z}\right\|_2 - \left\|\mathbf{X^T X}\right\|_2}$.

■

Appendix E. Proof of Theorem 17

Proof Maximizing $KL(P_{\mathbf{Z},\mathbf{X}}||P_{\mathbf{Z}}P_{\mathbf{X}}) \equiv$ minimizing $MI(\mathbf{Z},\mathbf{X}) \equiv$ maximizing $\mathbb{E}(i(\mathbf{Z}(\mathbf{X});\mathbf{X}))$ where KL refers to KL-divergence, MI refers to mutual information and $i(.)$ refers to information density **?**.
For obfuscation of a chosen sensitive feature j, the goal is to minimize 46

$$i(\mathbf{Z}(\mathbf{X_1},\mathbf{X_2}\dots\mathbf{X_{j-1}});\mathbf{X_j})|(\mathbf{X1},\mathbf{X_2},\dots\mathbf{X_{j-1}}) \tag{44}$$

For ease of notation, denote $\mathbf{X_1},\mathbf{X_2}\dots\mathbf{X_{j-1}}$ as $\mathbf{X^{-j}}$ and the goal above can be simply restated as being

$$\underset{\mathbf{Z}}{\operatorname{argmin}}\ \mathbf{i}(\mathbf{Z}(\mathbf{X^{-j}};\mathbf{X_j})) - \mathbf{i}(\mathbf{Z}(\mathbf{X^{-j}});\mathbf{X^{-j}})$$

$$\equiv \underset{Z}{\operatorname{argmin}}\ MI(\mathbf{Z}(\mathbf{X^{-j}});\mathbf{X_j}) - \mathbf{MI}(\mathbf{Z}(\mathbf{X^{-j}});\mathbf{X^{-j}})$$

and in the Gaussian case

$$MI(\mathbf{Z},\mathbf{X_j}) = \frac{-1}{2} log[\frac{det(\mathbf{C})}{det(\mathbf{Z^T Z})\mathbf{det}(\mathbf{X_j^T X_j})}] \tag{45}$$

where $\mathbf{C} = \begin{pmatrix} \mathbf{Z^T Z} & \mathbf{Z^T X_j} \\ \mathbf{X_j^T Z} & \mathbf{X_j^T X_j} \end{pmatrix}$. Since minimizing mutual information is equivalent to maximizing

$$e^{-2MI} = \frac{\det(\mathbf{C})}{\det(\mathbf{Z^T Z X_j^T X_j})} \tag{46}$$

We show in 12, that minimizing regularized distance covariance maximizes the determinant $\det(\mathbf{C})$ while minimizing the determinant $\det(\mathbf{Z^T Z})$. As the denominator of 46 decreases while the numerator increases with the reduction of distance covariance, we also have that the mutual information between the smashed data $\mathbf{Z}$ and hidden attribute $\mathbf{X_j}$ decreases as a result. In addition, we know that equation 46 is equivalent to

$$\equiv \underset{\mathbf{Z}}{\operatorname{argmax}}\ \mathbf{X_j^T X_j} - \mathbf{X_j^T Z}(\mathbf{Z^T Z})^{-1}\mathbf{Z^T X_j} \equiv \underset{\mathbf{Z}}{\operatorname{argmin}}\ \mathbf{X_j^T \hat{X}_j}$$

This is in turn equivalent to doing the opposite of classical linear regression by maximizing the error with respect to learning covariates $\mathbf{Z}$ as

$$\underset{\mathbf{Z}}{\operatorname{argmax}}\ \|\mathbf{X_j} - \mathbf{Z}\beta\| \text{ s.t, } \beta = (\mathbf{Z^T Z})^{-1}\mathbf{Z^T X_j}$$

This provides an additional interpretation as well as a classical connection with respect to distance correlation for attribute privacy. ■

Appendix F. Modified EM updates for the structured distribution learning problem in section 10

We use Powell minimization method to obtain the update for the mean vectors and covariance matrices as follows

$$
\mu_b^{g(t+1)} =
$$

$$
\min_{\mu}
\left\{
\begin{array}{l}
-\omega_b^f \log \left[\sum_{b' \neq b} \omega_{b'}^{g\,(t)} e^{\mathrm{DCov}\left(\Sigma_{b'}^f, \frac{1}{N-1}\left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)^T \left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)\right)} + \omega_b^{g(t)} e^{\mathrm{DCov}\left(\Sigma_b^f, \frac{1}{N-1}\left(Z_b^{(t)} - \mu\right)^T \left(Z_b^{(t)} - \mu\right)\right)} \right] \\
-\omega_b^f \log \left[\sum_{b' \neq b} \frac{\omega_{b'}^{g\,(t)} e^{-\frac{1}{2}\left(\mu_{b'}^g - \mu_{b'}^f\right)^T \left(\Sigma_{b'}^f + \frac{1}{N-1}\left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)^T \left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)\right)^{-1} \left(\mu_{b'}^g - \mu_{b'}^f\right)}}{\sqrt{\left|\Sigma_{b'}^f + \frac{1}{N-1}\left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)^T \left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)\right|}} \right. \\
\left. + \frac{\omega_b^{g(t)} e^{-\frac{1}{2}\left(\mu - \mu_b^f\right)^T \left(\Sigma_b^f + \frac{1}{N-1}\left(Z_b^{(t)} - \mu\right)^T \left(Z_b^{(t)} - \mu\right)\right)^{-1} \left(\mu - \mu_b^f\right)}}{\sqrt{\left|\Sigma_b^f + \frac{1}{N-1}\left(Z_b^{(t)} - \mu\right)^T \left(Z_b^{(t)} - \mu\right)\right|}} \right] \\
+ \omega_b^f (C_1 + C_2) \\
+ \frac{1}{2} \sum_{b' \neq b} \left[n_{b'}^{(t+1)} \log \left| \left(\frac{1}{N-1} \left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)^T \left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right) \right)^{-1} \right| \right. \\
\left. + \sum_{i=1}^{N} \gamma_{ib'}^{(t+1)} \operatorname{Tr} \left(\left(\frac{\left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)^T \left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)}{N-1} \right)^{-1} \left(x_i - \mu_{b'}^{g\,(t)}\right) \left(x_i - \mu_{b'}^{g\,(t)}\right)^T \right) \right] \\
+ \frac{1}{2} \left[n_b^{(t+1)} \log \left| \left(\frac{1}{N-1} \left(Z_b^{(t)} - \mu\right)^T \left(Z_b^{(t)} - \mu\right) \right)^{-1} \right| \right. \\
\left. + \sum_{i=1}^{N} \gamma_{ib}^{(t+1)} \operatorname{Tr} \left(\left(\frac{\left(Z_b^{(t)} - \mu\right)^T \left(Z_b^{(t)} - \mu\right)}{N-1} \right)^{-1} (x_i - \mu)(x_i - \mu)^T \right) \right]
\end{array}
\right\}
$$

We use Powell minimization method to optimize for $\mathbf{Z_b}$ as

$$
Z_b^{(t+1)} = \min_Z \left\{ \begin{array}{l}
-\omega_b^f \log \left[\sum_{b' \neq b} \omega_{b'}^{g\,(t)} e^{\mathrm{DCov}\left(\Sigma_{b'}^f, \frac{1}{N-1}\left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)^T \left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)\right)} + \omega_b^{g(t)} e^{\mathrm{DCov}\left(\Sigma_b^f, \frac{1}{N-1}\left(Z - \mu_b^{g(t)}\right)^T \left(Z - \mu_b^{g(t)}\right)\right)} \right] \\
-\omega_b^f \log \left[\sum_{b' \neq b} \frac{\omega_{b'}^{g\,(t)} e^{-\frac{1}{2}\left(\mu_{b'}^g - \mu_{b'}^f\right)^T \left(\Sigma_{b'}^f + \frac{1}{N-1}\left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)^T \left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)\right)^{-1} \left(\mu_{b'}^g - \mu_{b'}^f\right)}}{\sqrt{\left|\Sigma_{b'}^f + \frac{1}{N-1}\left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)^T \left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)\right|}} \right. \\
\left. + \frac{\omega_b^{g(t)} e^{-\frac{1}{2}\left(\mu_b^{g(t)} - \mu_b^f\right)^T \left(\Sigma_b^f + \frac{1}{N-1}\left(Z - \mu_b^{g(t)}\right)^T \left(Z - \mu_b^{g(t)}\right)\right)^{-1} \left(\mu_b^{g(t)} - \mu_b^f\right)}}{\sqrt{\left|\Sigma_b^f + \frac{1}{N-1}\left(Z - \mu_b^{g(t)}\right)^T \left(Z - \mu_b^{g(t)}\right)\right|}} \right] \\
+ \omega_b^f (C_1 + C_2) \\
+ \frac{1}{2} \sum_{b' \neq b} \left[n_{b'}^{(t+1)} \log \left| \left(\frac{1}{N-1} \left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)^T \left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right) \right)^{-1} \right| \right. \\
\left. + \sum_{i=1}^{N} \gamma_{ib'}^{(t+1)} \operatorname{Tr} \left(\left(\frac{\left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)^T \left(Z_{b'}^{(t)} - \mu_{b'}^{g\,(t)}\right)}{N-1} \right)^{-1} \left(x_i - \mu_{b'}^{g\,(t)}\right) \left(x_i - \mu_{b'}^{g\,(t)}\right)^T \right) \right] \\
+ \frac{1}{2} \left[n_b^{(t+1)} \log \left| \left(\frac{1}{N-1} \left(Z - \mu_b^{g(t)}\right)^T \left(Z - \mu_b^{g(t)}\right) \right)^{-1} \right| \right. \\
\left. + \sum_{i=1}^{N} \gamma_{ib}^{(t+1)} \operatorname{Tr} \left(\left(\frac{\left(Z - \mu_b^{g(t)}\right)^T \left(Z - \mu_b^{g(t)}\right)}{N-1} \right)^{-1} \left(x_i - \mu_b^{g(t)}\right) \left(x_i - \mu_b^{g(t)}\right)^T \right) \right]
\end{array} \right\}
$$

Differentiating $\log \sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f + \Sigma_b^g|}}$ with respect to Σ_b^g,

$$
\begin{aligned}
\frac{\partial}{\partial \Sigma_b^g} \left(\log \sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f + \Sigma_b^g|}} \right) &= \frac{\omega_b^g}{\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f + \Sigma_b^g|}}} \cdot \frac{\partial}{\partial \Sigma_b^g} \left(\left| \Sigma_a^f + \Sigma_b^g \right|^{-\frac{1}{2}} \right) \\
&= \frac{\omega_b^g}{\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f + \Sigma_b^g|}}} \cdot -\frac{1}{2} \left| \Sigma_a^f + \Sigma_b^g \right|^{-\frac{3}{2}} \cdot \frac{\partial}{\partial \Sigma_b^g} \left(|\Sigma_a^f + \Sigma_b^g| \right)
\end{aligned}
$$

The second derivative is then given by:

$$\frac{\partial^2}{\partial(\Sigma_b^g)^2}\left(\log\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}\right) = \frac{\partial}{\partial\Sigma_b^g}\left(\frac{-\omega_b^g}{2\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{3}{2}}\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}}\cdot\frac{\partial}{\partial\Sigma_b^g}\left(|\Sigma_a^f+\Sigma_b^g|\right)\right)$$

$$= -\frac{\omega_b^g}{2}\frac{\partial}{\partial\Sigma_b^g}\left(\left(\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{3}{2}}\sum_b\frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}\right)^{-1}\frac{\partial}{\partial\Sigma_b^g}\left(|\Sigma_a^f+\Sigma_b^g|\right)\right)$$

$$= \begin{aligned}&\frac{\omega_b^g}{2}\left(\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{3}{2}}\sum_b\frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}\right)^{-2}\cdot\frac{\partial}{\partial\Sigma_b^g}\left(\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{3}{2}}\sum_b\frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}\right)\cdot\frac{\partial}{\partial\Sigma_b^g}\left(|\Sigma_a^f+\Sigma_b^g|\right)\\ &-\frac{\omega_b^g}{2}\left(\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{3}{2}}\sum_b\frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}\right)^{-1}\cdot\frac{\partial^2}{\partial(\Sigma_b^g)^2}\left(|\Sigma_a^f+\Sigma_b^g|\right)\end{aligned}$$

where

$$\frac{\partial}{\partial\Sigma_b^g}\left(\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{3}{2}}\sum_b\frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}\right)$$

$$= \frac{3}{2}\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{1}{2}}\cdot\frac{\partial}{\partial\Sigma_b^g}\left(\left|\Sigma_a^f+\Sigma_b^g\right|\right)\cdot\sum_b\frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}+\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{3}{2}}\cdot\frac{-\omega_b^g}{2\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{3}{2}}}\cdot\frac{\partial}{\partial\Sigma_b^g}\left(|\Sigma_a^f+\Sigma_b^g|\right)$$

$$= \frac{1}{2}\cdot\frac{\partial}{\partial\Sigma_b^g}\left(|\Sigma_a^f+\Sigma_b^g|\right)\left(3\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{1}{2}}\sum_b\frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}-\omega_b^g\right)$$

Then, for the second derivative, we have

$$\frac{\partial^2}{\partial(\Sigma_b^g)^2}\left(\log\sum_b\frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}\right)$$

$$= \frac{\omega_b^g}{2\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{3}{2}}\sum_b\frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}}\left[\frac{\left(\frac{\partial}{\partial\Sigma_b^g}\left(|\Sigma_a^f+\Sigma_b^g|\right)\right)^2\left(3\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{1}{2}}\sum_b\frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}-\omega_b^g\right)}{2\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{3}{2}}\sum_b\frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}}-\frac{\partial^2}{\partial(\Sigma_b^g)^2}\left(|\Sigma_a^f+\Sigma_b^g|\right)\right.$$

Using the property $\nabla_X \det^k(X+tY) = k\det^k(X+tY)(X+tY)^{-T}$ (from page 449 of *Convex Optimization & Euclidean Distance Geometry*) with $k=t=1$, the first and second derivatives of

$\left|\Sigma_a^f + \Sigma_b^g\right|$ are given by

$$\begin{aligned}
\frac{\partial}{\partial \Sigma_b^g}\left(|\Sigma_a^f + \Sigma_b^g|\right) &= \left|\Sigma_a^f + \Sigma_b^g\right| (\Sigma_a^f + \Sigma_b^g)^{-T} \\
&= \left|\Sigma_a^f + \Sigma_b^g\right| \left((\Sigma_a^f + \Sigma_b^g)^T\right)^{-1} \\
\frac{\partial}{\partial \Sigma_b^g}\left(|\Sigma_a^f + \Sigma_b^g|\right) &= \left|\Sigma_a^f + \Sigma_b^g\right| \left(\Sigma_a^f + \Sigma_b^g\right)^{-1} \text{ since } \Sigma_a^f \text{ and } \Sigma_b^g \text{ are symmetric} \qquad (47)
\end{aligned}$$

Note that $\left(\Sigma_a^f + \Sigma_b^g\right)$ must be invertible.

$$\begin{aligned}
\frac{\partial^2}{\partial (\Sigma_b^g)^2}\left(|\Sigma_a^f + \Sigma_b^g|\right) &= \frac{\partial}{\partial \Sigma_b^g}\left(\left|\Sigma_a^f + \Sigma_b^g\right| \left(\Sigma_a^f + \Sigma_b^g\right)^{-1}\right) \\
&= \frac{\partial}{\partial \Sigma_b^g}\left(|\Sigma_a^f + \Sigma_b^g|\right) \cdot \left(\Sigma_a^f + \Sigma_b^g\right)^{-1} + |\Sigma_a^f + \Sigma_b^g| \cdot \frac{\partial}{\partial \Sigma_b^g}\left(\left(\Sigma_a^f + \Sigma_b^g\right)^{-1}\right)
\end{aligned}$$

Applying Property 40 of the Matrix Cookbook, $\partial(X^{-1}) = -X^{-1}(\partial X)X^{-1}$, and equation (12):

$$\begin{aligned}
\frac{\partial^2}{\partial (\Sigma_b^g)^2}\left(|\Sigma_a^f + \Sigma_b^g|\right) &= \left|\Sigma_a^f + \Sigma_b^g\right| \left(\left(\Sigma_a^f + \Sigma_b^g\right)^{-1}\right)^2 + \left|\Sigma_a^f + \Sigma_b^g\right| (-1) \left(\Sigma_a^f + \Sigma_b^g\right)^{-1} \frac{\partial}{\partial \Sigma_b^g}\left(\Sigma_a^f + \Sigma_b^g\right) \left(\Sigma_a^f + \Sigma_b^g\right) \\
&= \left|\Sigma_a^f + \Sigma_b^g\right| \left[\left(\left(\Sigma_a^f + \Sigma_b^g\right)^{-1}\right)^2 - \left(\Sigma_a^f + \Sigma_b^g\right)^{-1} \frac{\partial}{\partial \Sigma_b^g}\left(\Sigma_a^f + \Sigma_b^g\right) \left(\Sigma_a^f + \Sigma_b^g\right)^{-1}\right]
\end{aligned}$$

If $\frac{\partial}{\partial \Sigma_b^g}\left(\Sigma_a^f + \Sigma_b^g\right) = I$,

$$\begin{aligned}
\frac{\partial^2}{\partial (\Sigma_b^g)^2}\left(|\Sigma_a^f + \Sigma_b^g|\right) &= \left|\Sigma_a^f + \Sigma_b^g\right| \left[\left(\left(\Sigma_a^f + \Sigma_b^g\right)^{-1}\right)^2 - \left(\Sigma_a^f + \Sigma_b^g\right)^{-1} \cdot I \cdot \left(\Sigma_a^f + \Sigma_b^g\right)^{-1}\right] \\
&= \left|\Sigma_a^f + \Sigma_b^g\right| \left[\left(\left(\Sigma_a^f + \Sigma_b^g\right)^{-1}\right)^2 - \left(\left(\Sigma_a^f + \Sigma_b^g\right)^{-1}\right)^2\right] \\
\frac{\partial^2}{\partial (\Sigma_b^g)^2}\left(|\Sigma_a^f + \Sigma_b^g|\right) &= 0 \qquad (48)
\end{aligned}$$

Therefore,

$$\frac{\partial^2}{\partial(\Sigma_b^g)^2}\left(\log\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}\right)$$

$$= \frac{\omega_b^g}{2\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{3}{2}}\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}}\left[\frac{\left(\left|\Sigma_a^f+\Sigma_b^g\right|\left(\Sigma_a^f+\Sigma_b^g\right)^{-1}\right)^2\left(3\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{1}{2}}\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}-\omega_b^g\right)}{2\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{3}{2}}\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}}-0\right]$$

$$= \frac{\omega_b^g}{2\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{3}{2}}\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}}\cdot\frac{\left|\Sigma_a^f+\Sigma_b^g\right|^2\left(\left(\Sigma_a^f+\Sigma_b^g\right)^{-1}\right)^2\left(3\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{1}{2}}\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}-\omega_b^g\right)}{2\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{3}{2}}\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}}$$

$$= \frac{\omega_b^g\left(\left(\Sigma_a^f+\Sigma_b^g\right)^{-1}\right)^2\left(3\left|\Sigma_a^f+\Sigma_b^g\right|^{\frac{1}{2}}\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}-\omega_b^g\right)}{4\left|\Sigma_a^f+\Sigma_b^g\right|\left(\sum_b \frac{\omega_b^g}{\sqrt{|\Sigma_a^f+\Sigma_b^g|}}\right)^2}$$