From Noisy Fixed-Point Iterations to

A Unified Theory of Private Optimisation for Centralised and Federated Learning

Debabrota Basu | debabrota-basu.github.io

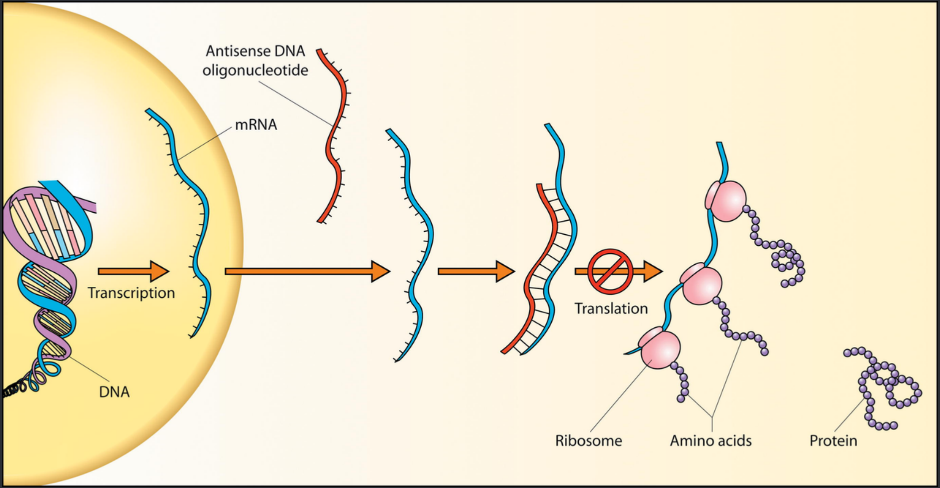
Équipe Scool, Inria, Univ. Lille, CNRS- CRIStAL, Centrale Lille, France

Indian Statistical Institute, Kolkata, July 2023

The Trajectory

1. Motivation: Collaborative Drug Design
2. Warm-up: Differentially Private Optimisation
3. Unification: Fixed-point Operators with and without Noise
4. Application: Differentially Private ADMM as Noisy Fixed-point Operator
5. The Curtain Call

Antisense Oligonucleotide (ASO) Drugs

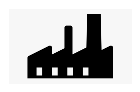


Source:https://zh.wikipedia.org/wiki/File:Antisense_DNA_oligonucleotide.png

Collaborative Drug Design

Dataset D_1

Classifier $f(.|w_1)$

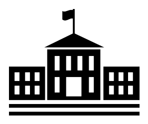


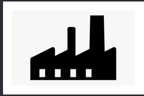
Dataset D_2

Classifier $f(.|w_2)$

Dataset D_3

Classifier $f(.|w_3)$

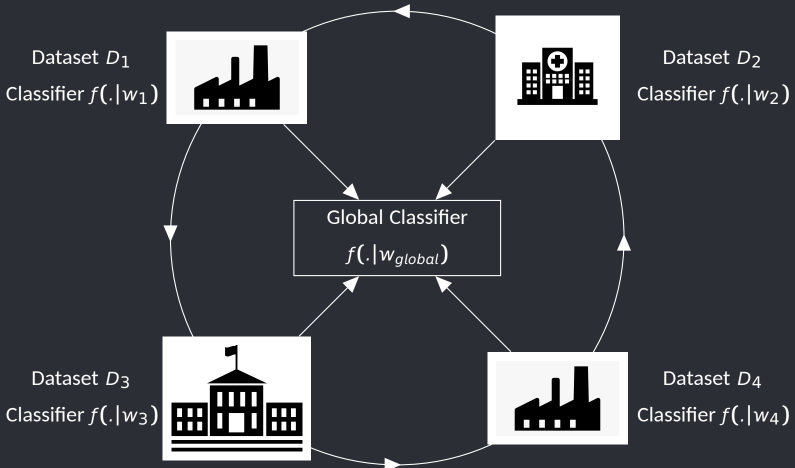




Dataset D_4

Classifier $f(.|w_4)$

Collaborative Drug Design



Collaborative Drug Design with Privacy [Tavara et al., 2021]

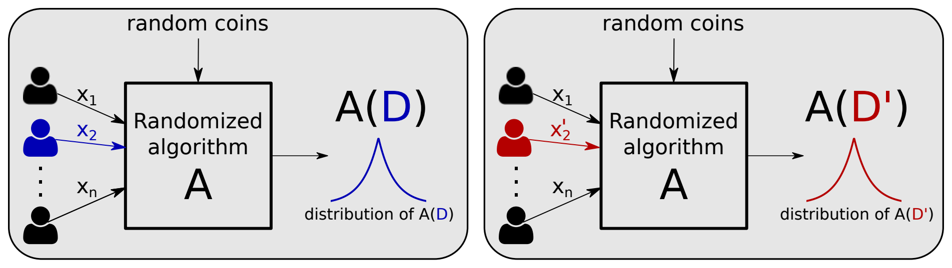
Issues in Collaboration

- Different organisations have different IP on models.
- The local datasets may contain sensitive/private information of the individuals involved.
- The data or partial models under communication can be used to leak the data and to reconstruct the models.

Solution: Distributed Learning with Differential Privacy

Add noise to the local parameters communicated between nodes such that inclusion/exclusion of an individual is indistinguishable.

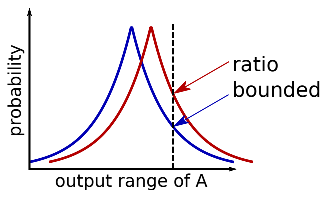
Component 1: Differential Privacy



Information in input/database becomes private
if it is indistinguishable from the output of a query/algorithm.

Component 1: Rényi Differential Privacy

- Neighbouring datasets $\mathscr{D} = \{x_1, x_2, \ldots, x_n\}$ and $\mathscr{D}' = \{x_1, x'_2, x_3, \ldots, x_n\}$
- DP implies $\mathscr{A}(\mathscr{D})$ and $\mathscr{A}(\mathscr{D}')$ should have similar distributions
- Similarity is measured in terms of different divergences leading to different DP definitions



Satisfying Rényi DP requires

$$D_\alpha(\mathscr{A}(\mathscr{D}) || \mathscr{A}(\mathscr{D}')) \leq \varepsilon$$

Compnent 2: Empirical Risk Minimisation (ERM)

Objective: Minimise expected risk of predicting erroneously $R(u|D) \triangleq \mathbb{E}_{(X,Y)\sim D}\left[\, l(u; X, Y)\right]$

Compnent 2: Empirical Risk Minimisation (ERM)

Objective: Minimise expected risk of predicting erroneously $R(u|D) \triangleq \mathbb{E}_{(X,Y)\sim D}\left[\, l(u; X, Y)\right]$

Issue: The data-generating distribution D is not known.

Solution: Use sample average of the risk obtained over the training dataset $\mathscr{D} = \{(X_i, Y_i)\}_{i=1}^{n}$ as a proxy.

$$\hat{R}(u|\mathscr{D}) \triangleq \frac{1}{n}\sum_{i=1}^{n} l(u; X_i, Y_i) = \frac{1}{n}\sum_{i=1}^{n} l(u; d_i)$$

Compnent 2: Empirical Risk Minimisation (ERM)

Objective: Minimise expected risk of predicting erroneously $R(u|D) \triangleq \mathbb{E}_{(X,Y)\sim D}[l(u;X,Y)]$

Issue: The data-generating distribution D is not known.

Solution: Use sample average of the risk obtained over the training dataset $\mathscr{D} = \{(X_i, Y_i)\}_{i=1}^n$ as a proxy.

$$\hat{R}(u|\mathscr{D}) \triangleq \frac{1}{n}\sum_{i=1}^{n} l(u;X_i,Y_i) = \frac{1}{n}\sum_{i=1}^{n} l(u;d_i)$$

Methodology:

1. **Modelling**: Compute **parameters** (or hypothesis) yielding **minimum empirical risk** over **training dataset**.

$$u^* \triangleq \operatorname*{argmin}_{u\in U} \hat{R}(u|\mathscr{D})$$

2. **Optimisation**: Use an optimisation algorithm, such as SGD in centralised setting, FedSGD and ADMM in federated setting, ADMM in distributed setting to solve the minimisation problem

A Walk through Differentially Private Optimisation

Differentially Private SGD and ADMM

Warm-up: Stochastic Gradient Descent

Algorithm 1 Stochastic (Projected) Gradient Descent (SGD)

Initialise $u_0 \in C \subset \mathbb{R}^p$ (independent of $\mathscr{D}$)
for $t = 0, \ldots, T-1$ **do**
    Pick $i_t \in \{1, \ldots, n\}$ uniformly at random
    $u_{t+1} \leftarrow u_t - \gamma^{(t)}\big(\nabla f(u_t; d_{i_t})\big)$

Utility: Given a convex and Lipschitz loss function, if we set $\gamma_t = \frac{\|C\|_2}{L\sqrt{t}}$, we get

$$\mathbb{E}[\, lu_T - lu^* \,] = \mathcal{O}\left(\frac{\|C\|_2 nL \log T}{\sqrt{T}}\right).$$

DP-SGD: Noise Injection during Optimisation [Bassily et al., 2014, Abadi et al., 2016]

Algorithm 2 Differentially Private SGD (DP-SGD)

Initialise $u_0 \in C \subset \mathbb{R}^p$ (independent of $\mathcal{D}$)
for $t = 0, \ldots, T-1$ **do**
    Pick $i_t \in \{1, \ldots, n\}$ uniformly at random
    $u_{t+1} \leftarrow u_t - \gamma^{(t)}\left(\nabla f(u_t; d_{i_t}) + \eta_{t+1}\right)$ where $\eta_{t+1} \sim \mathcal{N}(0, \sigma^2 \Delta^2 \mathbb{I}_p)$
Return u_T

DP-SGD: Noise Injection during Optimisation [Bassily et al., 2014, Abadi et al., 2016]

Algorithm 3 Differentially Private SGD (DP-SGD)

Initialise $u_0 \in C \subset \mathbb{R}^p$ (independent of $\mathscr{D}$)
for $t = 0, \ldots, T-1$ **do**
    Pick $i_t \in \{1, \ldots, n\}$ uniformly at random
    $u_{t+1} \leftarrow u_t - \gamma^{(t)} \left(\nabla f(u_t; d_{i_t}) + \eta_{t+1} \right)$ where $\eta_{t+1} \sim \mathcal{N}(0, \sigma^2 \Delta^2 \mathbb{I}_p)$
Return u_T

- Utility analysis: same as non-private SGD (with additional noise due to privacy)
- Privacy analysis: DP-SGD satisfies $(\alpha, \frac{\alpha T}{2n^2\sigma^2})$ Rényi DP following the subsampled Gaussian mechanism and composition property of RDP over T iterations.

Warm-up: ADMM

- Alternating Direction Method of Multipliers (ADMM) aims to solve:

$$\begin{aligned} \underset{x,z}{\text{minimize}} \quad & f(x;\mathscr{D}) + g(z) \\ \text{subject to} \quad & Ax + Bz = c \end{aligned}$$

Algorithm 4 ADMM algorithm

Input: initial point u_0, step size $\lambda \in (0,1]$, Lagrange parameter $\gamma > 0$

for $k = 0$ to $K-1$ **do**

$z_{k+1} = \operatorname{argmin}_z \left\{ g(z) + \frac{1}{2\gamma} \|Bz + u_k\|^2 \right\}$

$x_{k+1} = \operatorname{argmin}_x \left\{ f(x;\mathscr{D}) + \frac{1}{2\gamma} \|Ax + 2Bz_{k+1} + u_k - c\|^2 \right\}$

$u_{k+1} = u_k + 2\lambda\,(Ax_{k+1} + Bz_{k+1} - c)$

Return z^K

Differential Privacy-preserving ADMM

How can we make ADMM private and analyse its utility?

Algorithm 5 DP-ADMM algorithms

Input: initial point u_0, step size $\lambda \in (0, 1]$, Lagrange parameter $\gamma > 0$

for $k = 0$ to $K - 1$ **do**

$z_{k+1} = \mathrm{argmin}_z \left\{ g(z) + \frac{1}{2\gamma} \|Bz + u_k\|^2 \right\}$ (add a Gaussian noise and optimise)

$x_{k+1} = \mathrm{argmin}_x \left\{ f(x; \mathscr{D}) + \frac{1}{2\gamma} \|Ax + 2Bz_{k+1} + u_k - c\|^2 \right\}$ (add a Gaussian noise and optimise)

$u_{k+1} = u_k + 2\lambda \left(Ax_{k+1} + Bz_{k+1} - c \right)$ (add a Gaussian noise)

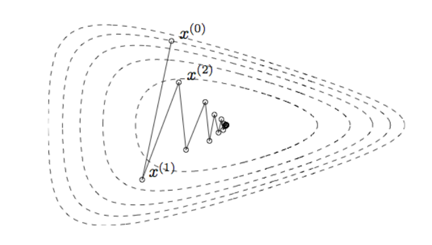
Return z^K

(Noisy) Fixed-point Operators

Fixed-point Operators with and without Noise

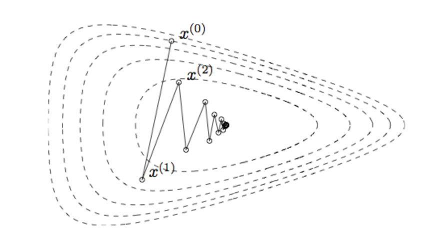
Optimisation Algorithms as Fixed-point Operators

- T an operator such that $u^{k+1} \triangleq T(u^k)$
- Non-expansive operator: $\|T(x) - T(y)\| \leq \|x - y\|, \forall x, y$
- α-averaged operator: $T = \alpha R + (1 - \alpha)I$, where R is non-expansive



Optimisation Algorithms as Fixed-point Operators

- T an operator such that $u^{k+1} \triangleq T(u^k)$
- Non-expansive operator: $||T(x) - T(y)|| \leq ||x - y||, \forall x, y$
- α-averaged operator: $T = \alpha R + (1 - \alpha)I$, where R is non-expansive



The Update Rule

$$\forall 1 \leq i \leq m, u_i^{k+1} = u_i^k + \rho_{i,k}(T_i(u^k) - u_i^k)$$

where $\rho_{i,k}$ is a Boolean (random) variable that parametrises if block i is updated at time k.

Noisy Fixed-point Operators

Algorithm 6 Noisy fixed-point iteration

Input: non-expansive operator $R = (R_1, \dots, R_B)$ over $1 \leq B \leq p$ blocks,step sizes $(\lambda_k)_{k\in\mathbb{N}} \in (0, 1]$, active blocks $(\rho_k)_{k\in\mathbb{N}} \in \{0, 1\}^B$, errors $(e_k)_{k\in\mathbb{N}}$, noise variance $\sigma^2 \geq 0$

for $k = 0, 1, \dots$ **do**

  for $b = 1, \dots, B$ **do**

    $u_{k+1,b} = u_{k,b} + \rho_{k,b}\lambda_k(R_b(u_k) + e_{k,b} + \eta_{k+1,b} - u_{k,b})$ with $\eta_{k+1,b} \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_p)$

This algorithm applies a λ_k-averaged operator with Gaussian noise, with possibly randomised, inexact and block-wise updates.

DP Optimisation as a Noisy Fixed-point Operator

Algorithm 7 Noisy fixed-point iteration

Input: non-expansive operator $R = (R_1, \dots, R_B)$ over $1 \leq B \leq p$ blocks,step sizes $(\lambda_k)_{k\in\mathbb{N}} \in (0, 1]$, active blocks $(\rho_k)_{k\in\mathbb{N}} \in \{0, 1\}^B$, errors $(e_k)_{k\in\mathbb{N}}$, noise variance $\sigma^2 \geq 0$

for $k = 0, 1, \dots$ **do**

  for $b = 1, \dots, B$ **do**

    $u_{k+1,b} = u_{k,b} + \rho_{k,b}\lambda_k(R_b(u_k) + e_{k,b} + \eta_{k+1,b} - u_{k,b})$ with $\eta_{k+1,b} \sim \mathcal{N}(0, \sigma^2\mathbb{I}_p)$

- We recover DP-SGD with

$$R(u) = u - \frac{2}{\beta}\nabla f(u; \mathscr{D}),$$

$$e_k = \frac{2}{\beta}(\nabla f(u_k; \mathscr{D}) - \nabla f(u_k; d_{i_k})), \text{ and } B = 1.$$

- This setup is linient to combine with amplification by iteration and by subsampling

General utility analysis [Cyffers et al., 2023]

Theorem (Utility guarantees for noisy fixed-point iterations)

Assume that R is τ-contractive with fixed point u^*. Let $P[\rho_{k,b} = 1] = q$ for some $q \in (0, 1]$. Then there exists a learning rate $\lambda_k = \lambda \in (0, 1]$ such that the iterates satisfy:

$$\mathbb{E}\left(\|u_{k+1} - u^*\|^2\right) \leqslant \left(1 - \frac{q^2(1-\tau)}{8}\right)^k D + 8\left(\frac{\sqrt{p}\sigma + \zeta}{\sqrt{q}(1-\tau)} + \frac{p\sigma^2 + \zeta^2}{q^3(1-\tau)^3}\right) \tag{1}$$

where $D = \|u_0 - u^*\|^2$, p is the dimension of u, and $\mathbb{E}[\|e_k\|^2] \leq \zeta^2$ for some $\zeta \geq 0$.

- The only assumption on R is that it is τ-contractive
- We roughly recover DP-SGD rate for strongly convex objective
- Let's apply it to ADMM

Differentially Private ADMM as a Noisy Fixed-point Operator

An Algorithmic Framework for Centralised, Federated, and Decentralised Settings

ADMM as a Fixed-point for ERM

ADMM can be written as Lions Mercier operator

$$T = \lambda R_{\gamma p_1} R_{\gamma p_2} + (1-\lambda)I$$

with $R_{\gamma p} = 2\,\mathrm{prox}_{\gamma p} - I$.

The consensus problem fits the general form solved by ADMM algorithms:

$$\begin{array}{ll} \underset{x \in \mathbb{R}^{np},\, z \in \mathbb{R}^{p}}{\text{minimize}} & \frac{1}{n}\sum_{i=1}^{n} f(x_i; d_i) + r(z) \\ \text{subject to} & x - I_{n(p\times p)} z = 0, \end{array}$$

where each data item d_i has its own parameter $x_i \in \mathbb{R}^p$

A Recipe for Centralised, Federated, and Decentralised DP-ADMM

Algorithm 8 Private ADMM

Input: initial point z_0, step size $\lambda \in (0, 1]$, privacy noise variance $\sigma^2 \geq 0$, parameter $\gamma > 0$, number of sampled users $1 \leq m \leq n$

for $k = 0$ to $K - 1$ **do**

$\hat{z}_{k+1} = \frac{1}{n} \sum_{i=1}^{n} u_{k,i}$

$z_{k+1} = \mathrm{prox}_{\gamma r}(\hat{z}_{k+1})$

for $i = 1$ to n **do**

$x_{k+1,i} = \mathrm{prox}_{\gamma f_i}(2z_{k+1} - u_{k,i})$

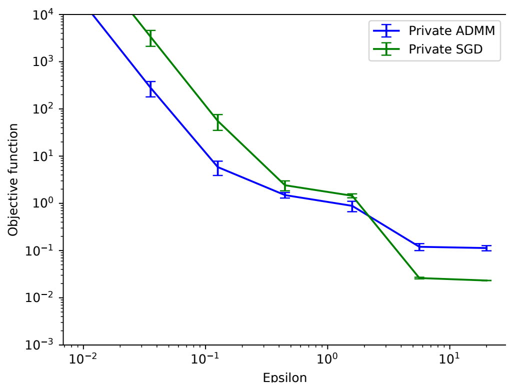
$u_{k+1,i} = u_{k,i} + 2\lambda\left(x_{k+1,i} - z_{k+1} + \frac{1}{2}\eta_{k+1,i}\right)$ with $\eta_{k+1,i} \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_p)$

Return z_K

Privacy-utility Trade-offs for Centralised, Federated, and Decentralised DP-ADMMs

	Centralised	Federated	Decentralised
Privacy loss	$\frac{8\alpha K L^2 \gamma^2}{\sigma^2 n^2}$	$\frac{16\alpha K L^2 \gamma^2}{\sigma^2 n^2}$	$\frac{8\alpha K_i L^2 \gamma^2 \ln n}{\sigma^2 n}$
$\mathbb{E}(\lVert u^K - u^* \rVert^2)$	$\frac{\sqrt{p\alpha} L\gamma}{\sqrt{\varepsilon n}(1-\tau)} + \frac{p\alpha L^2 \gamma^2}{\varepsilon n^2 (1-\tau)^3}$	$\frac{\sqrt{p\alpha} L\gamma}{\sqrt{\varepsilon r n}(1-\tau)} + \frac{p\alpha L^2 \gamma^2}{\varepsilon r^2 n^2 (1-\tau)^3}$	$\frac{\sqrt{p\alpha} L\gamma}{\sqrt{\varepsilon n}(1-\tau)} + \frac{p\alpha L^2 \gamma^2}{\varepsilon n (1-\tau)^3}$

Numerical Illustration: LASSO



- Synthetic sparse data with baseline DP-Prox SGD
- DP-ADMM shows a good robustness to high level of noise

Code: `https://github.com/totilas/padadmm`

The Curtain Call

Conclusion

- We provide a unifying view of private optimization algorithms by framing them as noisy fixed-point iterations, and prove general utility guarantees.
- Our framework can be used to derive and analyze new private algorithms by instantiating our general scheme with particular fixed-point operators.
- We illustrate this by designing private ADMM algorithms for centralised and federated learning; in contrast, prior work used ad-hoc algorithmic modifications and customised analysis with many privacy parameters.

Future Work

- Algorithm Design: Study this framework further to design novel algorithms with simpler and cleaner analysis.
- Analysis: Proving (weaker) utility guarantees for λ-averaged operators that are non-expansive but not contractive.
- Application: Deploying these algorithms for collaborative drug design.

Thanks to Edwige Cyffers and Aurélien Bellet,
who have been central to develop this research.

References

[Abadi et al., 2016] Abadi, M., Chu, A., Goodfellow, I. J., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016).
Deep learning with differential privacy.
In *CCS*.

[Bassily et al., 2014] Bassily, R., Smith, A. D., and Thakurta, A. (2014).
Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds.
In *FOCS*.

[Cyffers et al., 2023] Cyffers, E., Bellet, A., and Basu, D. (2023).
From noisy fixed-point iterations to private admm for centralized and federated learning.
arXiv preprint arXiv:2302.12559.

[Tavara et al., 2021] Tavara, S., Schliep, A., and Basu, D. (2021).
Federated learning of oligonucleotide drug molecule thermodynamics with differentially private admm-based svm.
In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 459–467. Springer.

ADMM as a Fixed-point Operator

ADMM can be written as Lions Mercier operator

$$T = \lambda R_{\gamma p_1} R_{\gamma p_2} + (1 - \lambda) I$$

with $R_{\gamma p} = 2 \operatorname{prox}_{\gamma p} - I$.

Two ways to instantiate it:

1. $p_1(u) = (-A \triangleright f)(-u - c)$ and $p_2(u) = (-B \triangleright g)(u)$ with

$$(M \triangleright f)(y) = \inf\{f(x) \mid Mx = y\}$$

2. $p_1(u) = \gamma^{-1} \partial f^* (-A^*)$ and $p_2(u) = \gamma^{-1} \partial g^*$

Private Centralised (per-coordinate) ADMM

Algorithm 9 Private Centralised ADMM

Initial vector u^0, step size $\lambda \in (0, 1]$, privacy noise variance $\sigma^2 \geq 0$, $\gamma > 0$

for $k = 0$ to $K - 1$ **do**

  $\hat{z}_{k+1} = \frac{1}{n} \sum_{i=1}^{n} u_{k,i}$

  $z_{k+1} = \mathrm{prox}_{\gamma r}(\hat{z}_{k+1})$

  for $i = 1$ to n **do**

    $x_{k+1,i} = \mathrm{prox}_{\gamma f_i}(2z_{k+1} - u_{k,i})$

    $u_{k+1,i} = u_{k,i} + 2\lambda\left(x_{k+1,i} - z_{k+1} + \frac{1}{2}\eta_{k+1,i}\right)$ with $\eta_{k+1,i} \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_p)$

return z_K

Private Federated ADMM

Algorithm 10 Private federated ADMM

Initial point z_0, step size $\lambda \in (0, 1]$, privacy noise variance $\sigma^2 \geq 0$, parameter $\gamma > 0$, number of sampled users $1 \leq m \leq n$
Server loop:
for $k = 0$ to $K - 1$ **do**
  Subsample a set S of m users
  for $i \in S$ **do**
    $\Delta u_{k+1,i} =$ **LocalADMMstep**(z_k, i)
  $\hat{z}_{k+1} = z_k + \frac{1}{n} \sum_{i \in S} \Delta u_{k+1,i}$
  $z_{k+1} = \mathrm{prox}_{\gamma r}(\hat{z}_{k+1})$
return z_K

Algorithm 11 LocalADMMstep

Sample $\eta_{k+1,i} \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_p)$
$x_{k+1,i} = \mathrm{prox}_{\gamma f_i}(2z_k - u_{k,i})$
$u_{k+1,i} = u_{k,i} + 2\lambda \left(x_{k+1,i} - z_k + \frac{1}{2} \eta_{k+1,i} \right)$
return $u_{k+1,i} - u_{k,i}$

Private Decentralised ADMM

Algorithm 12 Private (fully) Decentralised ADMM

Initial points u_0 and z_0, step size $\lambda \in (0, 1]$, privacy noise variance $\sigma^2 \geq 0$, $\gamma > 0$
for $k = 0$ to $K - 1$ **do**
    Let i be the currently selected user
    Sample $\eta_{k+1,i} \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_p)$
    $x_{k+1,i} = \mathrm{prox}_{\gamma f_i}(2z_k - u_{k,i})$
    $u_{k+1,i} = u_{k,i} + 2\lambda \left(x_{k+1,i} - z_k + \frac{1}{2}\eta_{k+1,i} \right)$
    $\hat{z}_{k+1} = z_k + \frac{1}{n}(u_{k+1,i} - u_{k,i})$
    $z_{k+1} = \mathrm{prox}_{\gamma r}(\hat{z}_{k+1})$
    Send z_{k+1} to a random user