Localization and Approximation Based Methods for Distributed Control and Optimization

James Anderson

Department of Electrical Engineering
Columbia University

July 15, 2022

The problem(?)

We have more data than we can compute with

"According to all the big data we've gathered, our discussions about big data are up 72% this year alone."

Systems are getting bigger, sensors are getting cheaper & smaller

The fallacy

Distributed computing/The Cloud/The Edge/GPUs will save us

Trade solution accuracy for improved computation time.

Outline

1. Approximation via sketching
   - Randomized SVD
   - Distributed iterative Hessian sketching
2. Federated learning
   - Approximation
   - Localization

Key Idea: Sketching

Ground truth

Sketch 1

Sketch 2

How to generate a sketch? Error incurred when using the sketch?

arXiv > math > arXiv:2109.02703

Mathematics > Optimization and Control

[Submitted on 6 Sep 2021]

Large-Scale System Identification Using a Randomized SVD

Han Wang, James Anderson

Singular Value Decomposition

Given $A \in \mathbb{C}^{m \times n}$, define $p = \min\{m, n\}$, the SVD of A is given by

$$A = U \Sigma V^*$$

where

- $U \in \mathbb{C}^{m \times m}$ is unitary
- $V \in \mathbb{C}^{n \times n}$ is unitary
- $\Sigma \in \mathbb{R}^{m \times n}$ is diagonal

when $m \neq n$, the matrix Σ takes the form

$$\Sigma = \begin{bmatrix} \widehat{\Sigma} \\ \mathbf{0} \end{bmatrix} \quad \text{or} \quad \Sigma = \begin{bmatrix} \widetilde{\Sigma} & \mathbf{0} \end{bmatrix}$$

where $\widehat{\Sigma} = \mathbf{diag}(\sigma_1, \ldots, \sigma_n)$ and $\widetilde{\Sigma} = \mathbf{diag}(\sigma_1, \ldots, \sigma_m)$, and

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$$

Alternative SVDs

Compact SVD

$$A = U_r \Sigma_r V_r^*$$

- Σ_r is a **square** $r \times r$ matrix
- r is given by the number of non-zero singular values
- U_r and V_r are semi-unitary

Truncated SVD

$$A \approx U_l \Sigma_l V_l^*$$

- Σ_l is a **square** $l \times l$ matrix and $l < r$
- U_l and V_l are semi-unitary

Optimal Low-Rank Approximation

Error bounds

The Eckart-Young theorem tells us that for eack k:

$$\sigma_{k+1} = \begin{array}{ll} \text{minimize} & \|A - X\| \\ \text{subject to} & \mathbf{rank}(X) \leq k \end{array}$$

Moreover, an optimal X can be constructed from the k-dominant left singular vectors of A:

$$X = QQ^*A$$

Computational cost

- Standard SVD cost $O(mn^2)$ when $n \ll m$
- l-truncated SVD takes $O(mnl)$ flops using classical methods

Randomized SVD Algorithm: RSVD

Stage 1

Find a matrix Q such that

$$A \approx QQ^*A, \quad \text{where } Q \text{ has orthonormal columns.}$$

Interpret $\approx$ as meaning that Q satisfies

$$\|(I - QQ^*)A\| \leq \epsilon, \quad \text{for some acceptable } \epsilon > 0.$$

Stage 2

Use Q and your favorite classical SVD algorithm then rearrange.

Stage 1: Random Sampling

Objective

Given a matrix $A \in \mathbb{C}^{m \times n}$ compute an approximate basis for $\mathbf{range}(A)$.

Algorithm

1. Sample the range of A by generating independent "random" vectors $\omega^{(i)}$ for $i = 1, \ldots, k$:

$$y^{(i)} = A\omega^{(i)} \qquad \Longleftrightarrow \qquad Y = A\Omega$$

2. Orthogonalize the columns of Y

Note

- The matrix Y is a called a **sketch** of A
- $Y \in \mathbb{C}^{m \times k}$ where $k \ll \min\{m, n\}$

Stage 1: Analysis

Stage 1: Algorithm

1. Draw a **random matrix** $\Omega \in \mathbb{R}^{n \times (k+l)}$ // Draw l extra samples
2. Form the **sketch** $Y = A\Omega$ // Cost $O(mn(k+l))$
3. Construct orthogonal basis: $[Q, \sim] = \texttt{QR}(Y)$ // Cost $O(n(k+l)^2)$

Theorem (Halko, Martinsson, Tropp, 2011)

Given $A \in \mathbb{R}^{m \times n}$, a target rank $k \geq 2$, and parameter $l \geq 2$ such that $k + l \leq \min\{m, n\}$. The algorithm above produces a matrix Q with orthonormal columns that satisfies

$$\mathbf{E} \, \|A - QQ^* A\| \leq \left[1 + \frac{4\sqrt{k+l}}{l-1} \sqrt{\min\{m, n\}}\right] \sigma_{k+1}.$$

Stage 2: Building an Approximate SVD

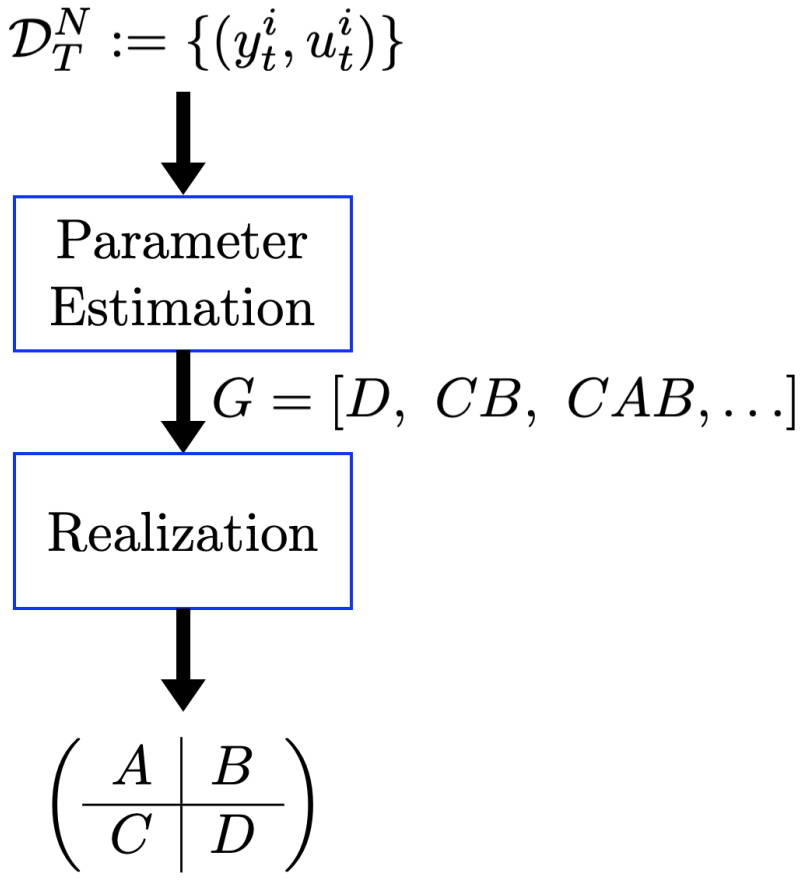
$$A \approx Q \, Q^* \, A$$

$$= Q \, Q^* A$$

$$= Q \, U \, \Sigma \, V^* = Q^* U \, \Sigma \, V^*$$

Realization Using RSVD

Linear System Identification



Realization Using RSVD

Problem formulation

Collect data from the LTI system

$$x_{t+1} = Ax_t + Bu_t + w_t$$
$$y_t = Cx_t + Du_t + v_t$$

The data

- We have $N < \infty$ observations over finite time-horizon T
- Observe: $\{y_t^i\}_{t=0}^T$, $\{u_t^i\}_{t=0}^T$ for $i = 1, \ldots, N$
- Assume: $u_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_u^2 I)$, $w_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_w^2 I)$, $v_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_v^2 I)$

Objective

Find matrices $(\hat{A}, \hat{B}, \hat{C}, \hat{D})$ that "best fit" observed data

Realization

The process of obtaining a state-space model from the Markov matrix

$$G = \begin{bmatrix} D & CB & CAB & \dots & CA^{T-2}B \end{bmatrix} \in \mathbb{R}^{m \times Tp}.$$

- When $w_t, v_t \equiv 0$ have access to G, otherwise must solve an optimization problem to obtain an estimate $\hat{G}$
- The realization problem is to determine the state-space matrices from $\hat{G}$, *i.e.*, a mapping

$$\hat{G} \mapsto \left(\begin{array}{c|c} \hat{A} & \hat{B} \\ \hline \hat{C} & \hat{D} \end{array}\right)$$

Ho-Kalman Algorithm

Assumptions

- (A, B, C) is minimal
- $n = \mathbf{rank}(\mathcal{H}) \leq \min\{T_1, T_2\}$

Algorithm

1. From G construct the Hankel matrix

$$\mathcal{H} = \left[\begin{array}{ccc|c} CB & CAB & \dots & CA^{T_2}B \\ CAB & CA^2B & \dots & CA^{T_2+1}B \\ CA^2B & CA^3B & \dots & CA^{T_2+2}B \\ \vdots & \vdots & \vdots & \vdots \\ CA^{T_1-1}B & CA^{T_1}B & \dots & CA^{T_1+T_2-1}B \end{array}\right] \in \mathbb{R}^{pT_1 \times m(T_1+1)}$$

where $T = T_1 + T_2 + 1$.

By assumption $\mathcal{H}$ and $\mathcal{H}^-$ are full rank

Ho-Kalman Algorithm

Assumptions

- (A, B, C) is minimal
- $n = \mathbf{rank}(\mathcal{H}) \leq \min\{T_1, T_2\}$

Step 2: Factorization

$\mathcal{H}^-$ is full rank and so it can be factored as

$$\mathcal{H}^- = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{T-1} \end{bmatrix} \begin{bmatrix} B & AB & \dots & A^{T_2-1}B \end{bmatrix}$$
$$= \mathcal{O}\mathcal{Q}$$

Ho-Kalman Algorithm

Algorithm

1. From G construct the Hankel matrix

$$\mathcal{H} = \left[\begin{array}{ccc|c} CB & CAB & \dots & CA^{T_2}B \\ CAB & CA^2B & \dots & CA^{T_2+1}B \\ CA^2B & CA^3B & \dots & CA^{T_2+2}B \\ \vdots & \vdots & \vdots & \vdots \\ CA^{T_1-1}B & CA^{T_1}B & \dots & CA^{T_1+T_2-1}B \end{array}\right] \in \mathbb{R}^{pT_1 \times m(T_1+1)}$$

   where $T = T_1 + T_2 + 1$.

2. Compute a n-**truncated SVD** of $\mathcal{H}^-$: $\mathcal{H}^- \approx U_n \Sigma_n V_n^*$
3. $\mathcal{O} = U_n \Sigma_n^{\frac{1}{2}}$, $\mathcal{Q} = \Sigma_n^{\frac{1}{2}} V_n^*$
4. $\hat{A} = \mathcal{O}^\dagger \mathcal{H}^+ \mathcal{Q}^\dagger$

Comments

- Realization is a non-convex problem
- H-K algorithm provides realizations unique up to similarity transform

$$(A, B, C, D) \mapsto (TAT^{-1}, TB, CT^{-1}, D)$$

- Computation cost: at least $O(pmnT_1T_2)$ flops from the SVD step.
- Robustness guarantee [Oymak & Ozay, 2019]:

$$\begin{aligned}\max \left\{ \|\hat{A} - T^{-1}AT\|, \|\hat{B} - T^{-1}B\|, \|\hat{C} - CT\| \right\} \\ \leq c\sqrt{\|G - \hat{G}\|} = O\left(\frac{1}{N^4}\right).\end{aligned}$$

Stochastic Ho-Kalman Algorithm

Idea

Replace the truncated SVD with a randomized SVD!

- Measurements contain noise, so full accuracy isn't necessary anyway
- The deterministic algorithm struggles with modest systems sizes

Main Result (Informal)

Theorem

The stochastic Ho-Kalman Algorithm reduces the computational complexity of the realization problem from $O(pmn^3)$ to $O(pmn^2 \log n)$ when $T_1 = T_2 = n$. The achievable robustness is the same as deterministic algorithm.

Numerical Experiments

Scalability and Approximation Error

Eg	(n, m, p, T)	$\dim(\hat{H}^-)$	**Running Time [s]**		**Realization Error**	
			Det.	**Stoch.**	**Det.**	**Stoch.**
1	(40,30,20,100)	2000×2970	5.7456		6.67e-04	
2	(60,50,40,360)	7200×8950	227.0116		8.27e-04	
3	(100,80,50,500)	12500×19920	922.8428		6.53e-04	
4	(120,110,90,600)	27000×32890	Inf		N/A	
5	(200,150,100,600)	30000×44850	Inf		N/A	

- $(n, m, p, T) \rightarrow$ (state, input, output, horizon)
- No parallelization used with the randomized SVD
- No power iterations
- Oversampling parameter: $p = 10$
- Relative error:

$$\frac{\|\mathcal{G} - \hat{\mathcal{G}}\|_{\mathcal{H}_\infty}}{\|\mathcal{G}\|_{\mathcal{H}_\infty}}$$

Numerical Experiments

Scalability and Approximation Error

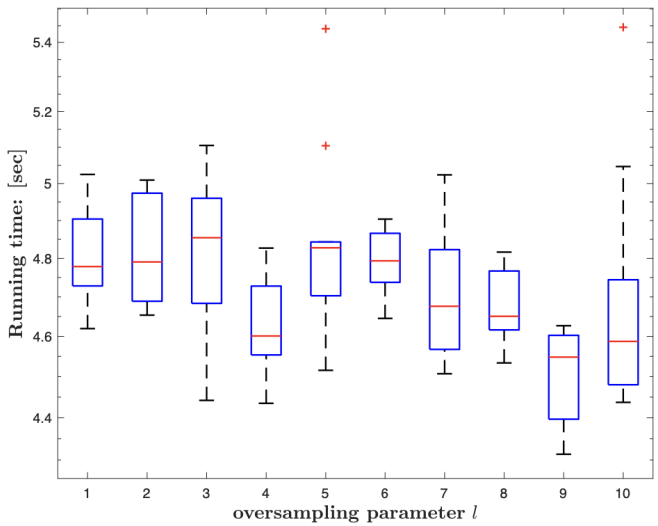
Eg	(n, m, p, T)	$\dim(\hat{H}^-)$	Running Time [s]		Realization Error	
			Det.	Stoch.	Det.	Stoch.
1	(40,30,20,100)	2000×2970	5.7456	0.0897	6.67e-04	1.19e-03
2	(60,50,40,360)	7200×8950	227.0116	0.9323	8.27e-04	1.75e-03
3	(100,80,50,500)	12500×19920	922.8428	4.4581	6.53e-04	1.66e-03
4	(120,110,90,600)	27000×32890	Inf	17.6603	N/A	1.96e-03
5	(200,150,100,600)	30000×44850	Inf	52.1762	N/A	1.45e-03

- $(n, m, p, T) \rightarrow$ (state, input, output, horizon)
- No parallelization used with the randomized SVD
- No power iterations
- Oversampling parameter: $p = 10$
- Relative error:

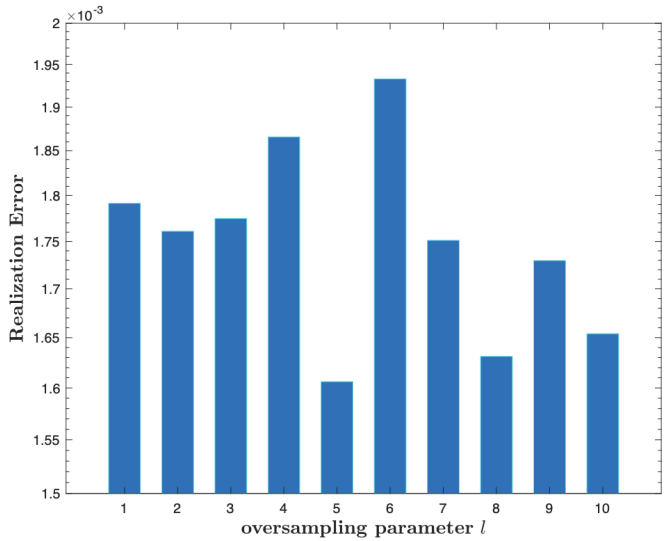
$$\frac{\|\mathcal{G} - \hat{\mathcal{G}}\|_{\mathcal{H}_\infty}}{\|\mathcal{G}\|_{\mathcal{H}_\infty}}$$

Additional Numerical Experiments

Oversampling parameter



(a) Running time of stochastic Ho-Kalman Algorithm using `RSVD` with oversampling parameter l.



(b) Realization error of stochastic Ho-Kalman Algorithm using `RSVD` with oversampling parameter l.

- Example 4: $(n, m, p, T) = (100, 80, 50, 500)$
- No power iterations

Stochastic Realization Algorithm: Conclusions

Methodology

- Performance degradation due to randomization almost negligible
- Non-asymptotic sample complexity bounds remain intact
- Order of magnitude gain in computation time (for large instances)
- Not yet exploited parallel computing

Algorithm tuning

- Algorithm performance can be boosted by including power iterations
  - This does impact running time
- Algorithm performance not sensitive to oversampling rate
  - Doesn't appear to impact running time

Learning Linear Models Using Distributed Iterative Hessian Sketching

Han Wang, James Anderson *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, PMLR 168:427-440, 2022.

Least Squares Sketching

Sketched Least squares

Given the problem:

$$x^\star \in \arg\min_{x \in \mathcal{C}} \underbrace{\frac{1}{2n}\|Ax - y\|^2}_{f(x)}, \quad A \in \mathbb{R}^{m \times n}$$

Instead solve:

$$x^\sharp \in \arg\min_{x \in \mathcal{C}} \underbrace{\frac{1}{2n}\|S(Ax - y)\|^2}_{g(x)},$$

with $S \in \mathbb{R}^{m \times n}, m \ll n$

- $f(x^\star) \leq f(x^\sharp) \leq (1 + \delta) f(x^\star)$

Iterative Hessian Sketch

Pilanci and Wainwright **[PW]** showed that this method is provably bad!

Rewrite the LS problem as

$$\underset{x \in \mathcal{C}}{\text{minimize}} \ \|Ax\|^2 - \langle x, A^T y \rangle.$$

Newton's method produces updates

$$x_{t+1} = x_t - \alpha (A^T A)^{-1} A^T (Ax_t - b).$$

If we **sketch** A in the norm only (and keep track of residuals), we get

$$x_{t+1} = x_t - \alpha (A^T S_t^T S_t A)^{-1} A^T (Ax_t - b).$$

Distribute this over q nodes:

$$x_{t+1} = x_t - \alpha \frac{1}{q} \sum_{k=1}^{q} (A^T S_{t,k}^T S_{t,k} A)^{-1} A^T (Ax_t - b).$$

Distributed Iterative Hessian Sketch

Proposed by Bartan and Pilanci **[BP]**

Generalized sketching and refined the analysis [Wang & Anderson 2022]

Algorithm 1 Distributed Iterative Hessian Sketch

Input: Number of iterations T, step size μ.
for $t = 1$ **to** T **do**
  for workers $k = 1$ **to** q in parallel **do**
    Sample $S_{t,k} \in \mathbb{R}^{m \times n}$.
    Sketch the data $S_{t,k}A$.
    Compute gradient $g_t = A^T(Ax_t - b)$.
    Solve $\hat{\Delta}_{t,k} = \arg\min_\Delta \frac{1}{2}\|S_{t,k}A\Delta\|_2^2 + g_t^T\Delta$ and send to master.
  end for
  Master: Update $x_{t+1} = x_t + \mu\frac{1}{q}\sum_{k=1}^{q}\hat{\Delta}_{t,k}$ and send x_{t+1} to workers.
end for
return x_T

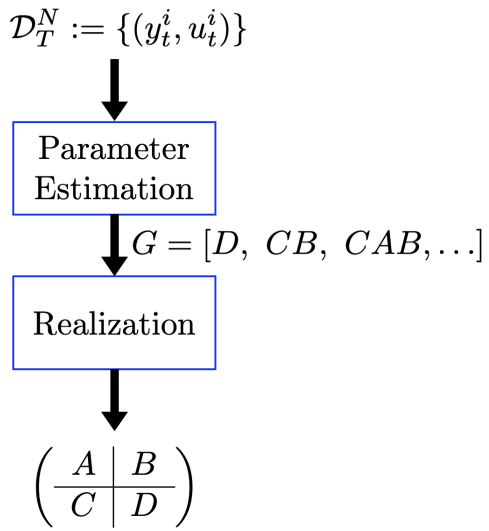
System Identification

Parameter estimation

Given observed data $\mathcal{D}$ believed to have been generated by

$$x_{t+1} = Ax_t + Bu_t + w_t$$
$$y_t = Cx_t + Du_t + v_t,$$

estimate the Markov parameters.



Learning Markov Parameters

OLS formulation

An estimate $\hat{G}$ of the Markov matrix is obtained by solving

$$\underset{X}{\text{minimize}} \ \|UX - Y\|_F^2$$

where

- $X \in \mathbb{R}^{mT \times p}$
- U and Y are Toeplitz
- Solution via QR decomposition: $O(NT(mT)^2)$

Result

DIHS applied to OLS problem

- Assume number of rollouts, N, satisfies $N > 8mT + 16\log(T/\delta)$
- Define $\kappa = mNT^2$

Theorem (Informal)

Fix $\delta \in (0,1)$ and $\rho \in (0, \frac{1}{2})$. If the sketch dimension satisfies

$$s > \frac{c_0 \log^4(\kappa)}{\rho^2} mT,$$

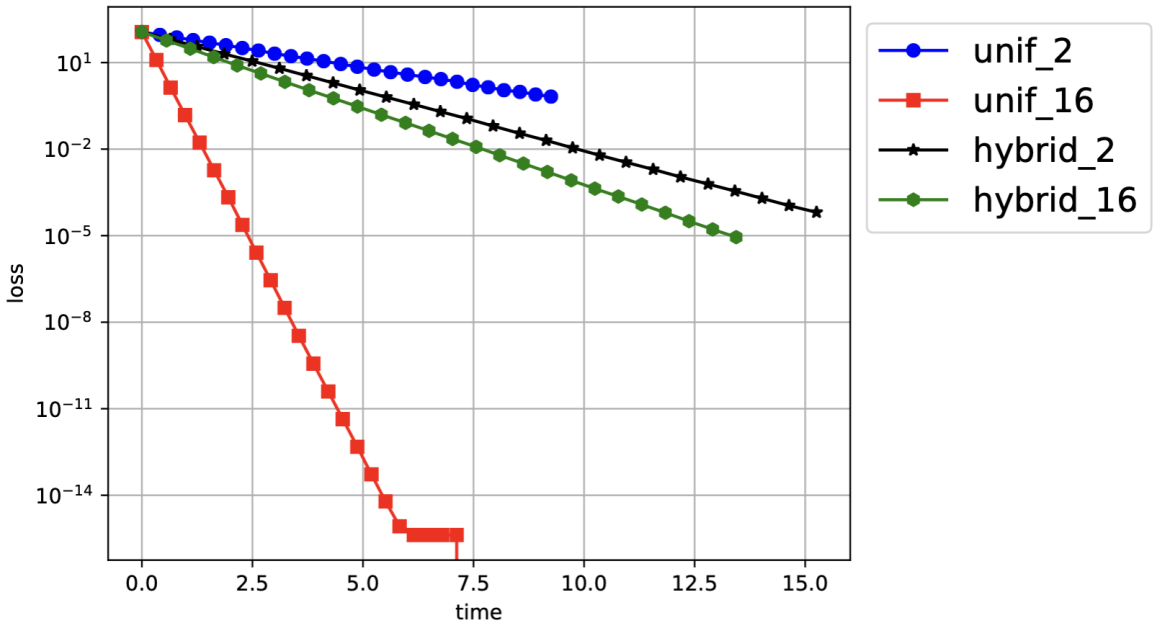
then at iteration k, DIHS satisfies

$$\|X_k - X^{\mathrm{LS}}\|_F \leq 2\left(\frac{\rho}{\sqrt{q}}\right)^k \|X^{\mathrm{LS}}\|_F$$

with high probability.

Numerical Experiments

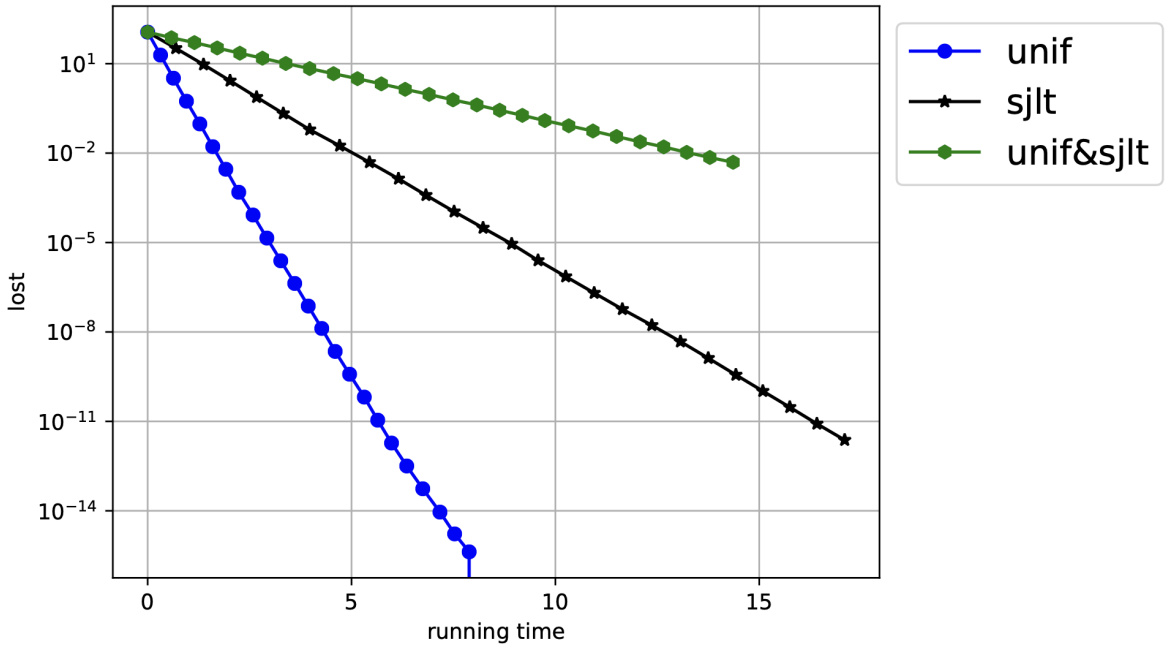
Sketch selection and number of workers



- 40 states, 30 inputs, 20 outputs
- $\sim 45M$ data points

Numerical Experiments

12 workers: same system



DIHS: Conclusions

- Randomized numerical linear algebra can be applied to Sys ID
- General least squares problems (and beyond)
- Applications to control synthesis?

arXiv > cs > arXiv:2203.15104

Computer Science > Machine Learning

[Submitted on 28 Mar 2022]

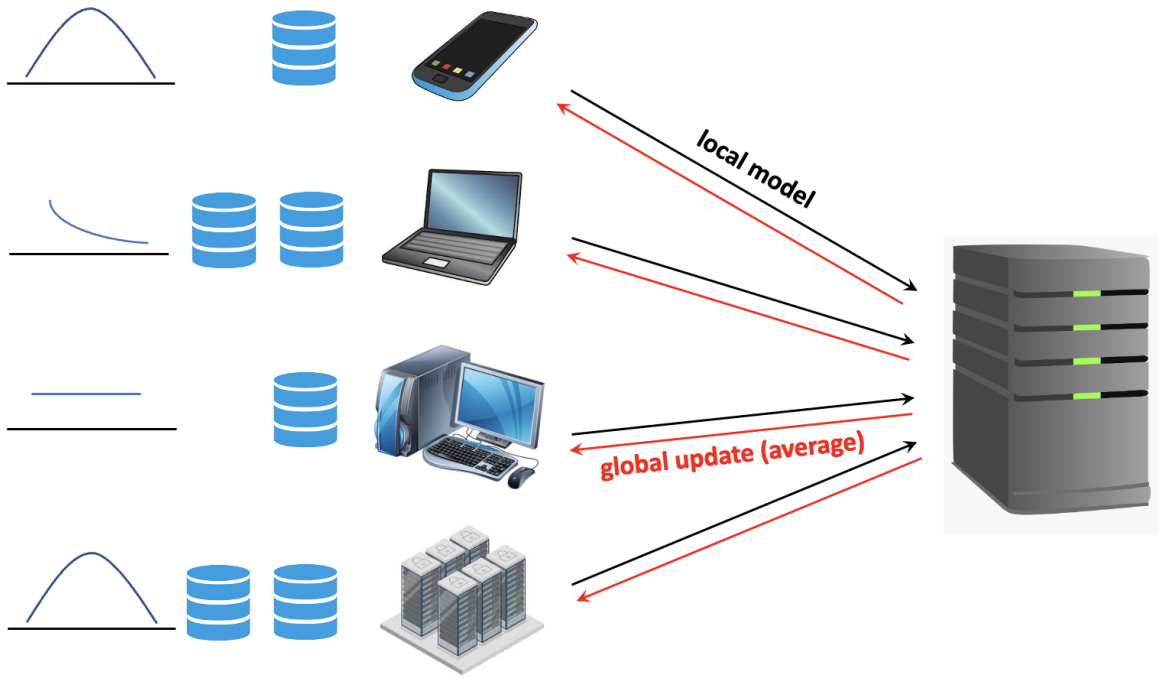
FedADMM: A Federated Primal-Dual Algorithm Allowing Partial Participation

Han Wang, Siddartha Marella, James Anderson

Federated Learning

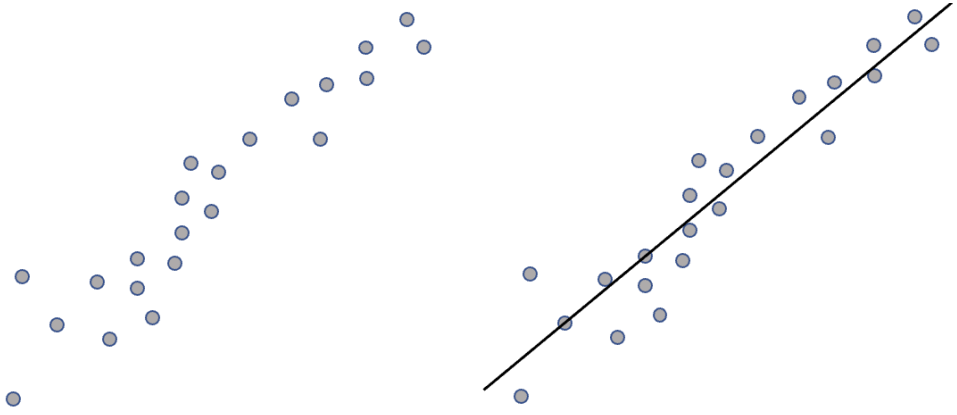
A framework for distributed optimization that accounts for:

- Device and data **heterogeneity**, and data **locality**



Federated Learning: Local Data

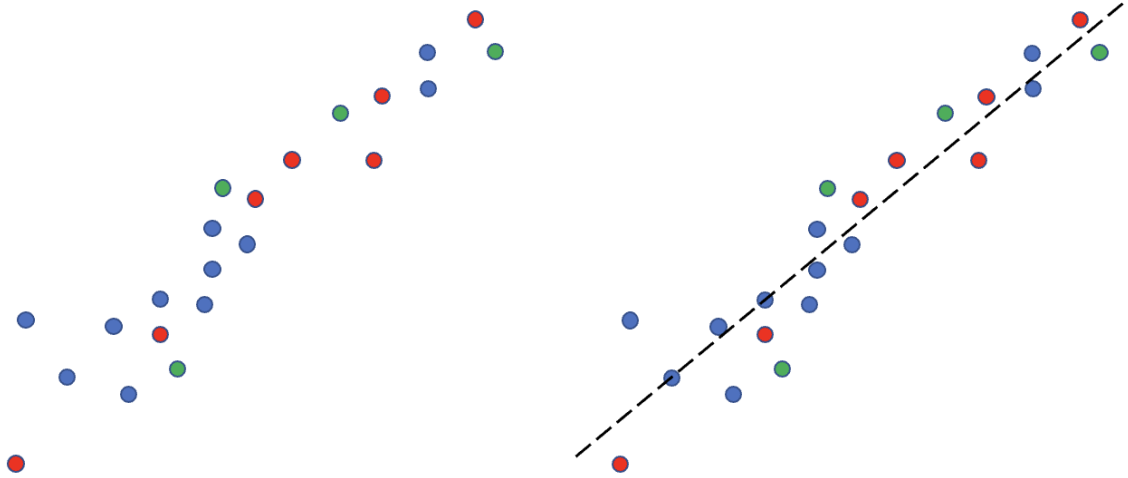
Centralized OLS



- All data in one place (or globally accessible)

Federated Learning: Local Data

Federated OLS



- Data is **not** shared between clients

Federated Learning

General problem formulation:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \frac{1}{N}\sum_{i=1}^{N} f_i(x) + g(x)$$

- f_i non-convex, L-smooth
- g non-smooth, convex
- problem data is stored locally on each device and is **never** shared
- client-server computation model

Federated Learning

We want to solve the distributed optimization problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \frac{1}{N}\sum_{i=1}^{N} f_i(x) + g(x)$$

No shortage of algorithms:

- FedAvg, FedSplit, FedProx, FedDR, SCAFFOLD, FedPD, FedDyn,...
- Our contribution: **FedADMM**
  - Converges with **partial participation** and **approximate** local solutions

FedADMM

Rewrite the problem as

$$\begin{aligned} \underset{x,\bar{x}}{\text{minimize}} \quad & \frac{1}{N}\sum_{i=1}^{N} f_i(x_i) + g(\bar{x}) \\ \text{s.t.} \quad & \mathbb{I} = \mathbb{1}\bar{x} \end{aligned}$$

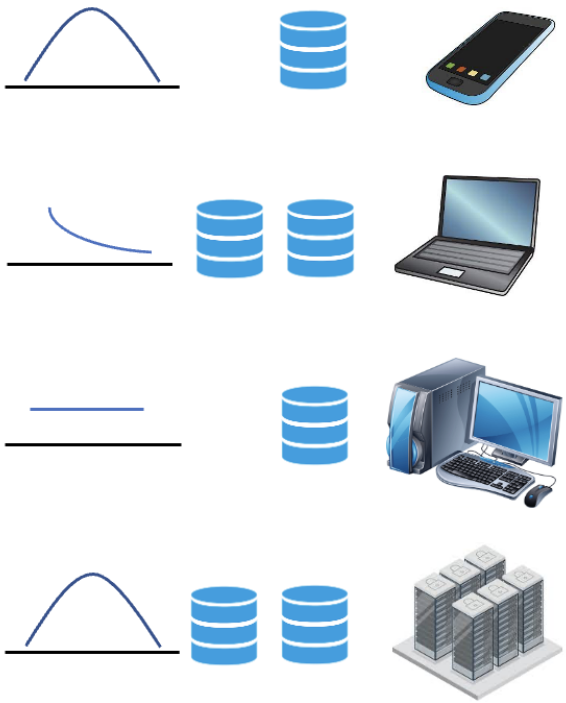
where

- x: concatenation of local variables $[x_1^T, x_2^T, \ldots, x_N^T]$
- $\bar{x}$: global consensus variable

Each agent has an augmented Lagrangian:

$$\mathcal{L}_i(x_i, \bar{x}, z_i) := f_i(x_i) + g(\bar{x}^k) + \langle z_i^k, x_i - \bar{x}^k \rangle + \frac{\eta}{2}\left\|x_i - \bar{x}^k\right\|^2$$

Client-side



▷ Client side

for each client $i \in \mathcal{S}_k$ **do**

receive $\bar{x}^k$ from the server.

$$x_i^{k+1} \approx \arg\min_{x_i} \mathcal{L}_i\left(x_i, \bar{x}^k, z_i^k\right)$$

$$z_i^{k+1} = z_i^k + \eta\left(x_i^{k+1} - \bar{x}^k\right) \quad \diamondsuit \text{Dual updates}$$

$$\hat{x}_i^{k+1} = x_i^{k+1} + \frac{1}{\eta} z_i^{k+1}$$

send $\Delta\hat{x}_i^k = \hat{x}_i^{k+1} - \hat{x}_i^k$ back to the server

end for

Client-side

Approximation

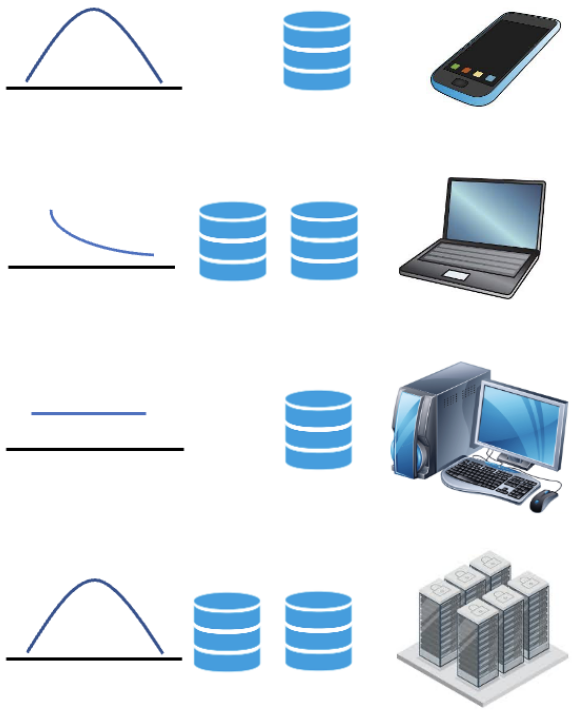
Clients do not have to minimize $\mathcal{L}_i$ precisely:

$$\left\| x_i^{k+1} - \arg \min_{x_i} \mathcal{L}_i(x_i, \bar{x}^k, z_i^k) \right\| \leq \epsilon_{i,k+1}$$

Partial Participation

At iteration k only a subset of clients $\mathcal{S}_k$ need to send local updates

Server-side



▷ Server side

aggregation $\tilde{x}^{k+1} = \tilde{x}^k + \frac{1}{n}\sum_{i\in\mathcal{S}_k} \Delta\hat{x}_i^k$

update $\bar{x}^{k+1} = \text{prox}_{g/\eta}\left(\tilde{x}^{k+1}\right)$

Analysis

Convergence (Informal):

When $g \equiv 0$, we have

$$\frac{1}{K+1}\sum_{k=0}^{K}\mathbb{E}\left[\left\|\nabla f(\bar{x}^k)\right\|^2\right] \leq \underbrace{\frac{c_1[F(x^0)-f^\star]}{K+1}}_{(1)} + \underbrace{\frac{1}{N(K+1)}l(\epsilon_{i,k},\epsilon_{i,k+1})}_{(2)}$$

where

$$l(\epsilon_{i,k},\epsilon_{i,k+1}) := \sum_{k=0}^{K}\sum_{i=1}^{n}(c_2\epsilon_{i,k}^2 + c_3\epsilon_{i,k+1}^2)$$

.

- (1): initial optimality gap
- (2): cost of working with approximate solutions
- Impact of partial participation reflected in the constants (omitted)

Analysis

Convergence (Informal):

If the sum of the accuracies is bounded by $D > 0$, then FedADMM requires

$$K = \left\lfloor \frac{c_1[F(x_0) - F^\star] + (c_2 + c_3)D}{\epsilon^2} \right\rfloor \equiv O(\epsilon^{-2})$$

to achieve an ϵ-suboptimal stationary point.

- All the above analysis can be extended to include g

Conclusions

- Randomized algorithms are widely used in ML & scientific computing
- Demonstrated their use in simple control applications
- Power applications?!

Acknowlegements

- **Han Wang**, Columbia University
- **Siddartha Marella**, Columbia University
- **NSF:** 2144634, **DoE:** DE-SC0022234

james.anderson@columbia.edu www.columbia.edu/∼ja3451

Bibliography

- **[HMT]**: Halko, Martinsson, and Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate decompositions*. SIAM Review, 53.2, 2011.
- **[OO]**: Oymak and Ozay, *Non-asymptotic identification of LTI systems from a single trajectory*. Proc. of the American Control Conference, 2019.
- **[PW]**: Pilanci and Wainwright, *Iterative Hessian sketch: Fast and accurate solution approximation for constrained least-squares*. Journal of Machine Learning Research 17, 2016.
- **[BP]**: Bartan and Pilanci, *Distributed averaging methods for randomized second order optimization*, arXiv:2002.0654, 2020.

Backup slides

Stage 1: Comments

Slowly decaying spectrum

Can boost accuracy by incorporating **power iterations**. Based on the observation:

$$W := (AA^*)^q A$$

has the same singular vectors as A. But

$$\sigma_j(W) = \sigma_j(A)^{2q+1}, \quad j = 1, 2, \ldots$$

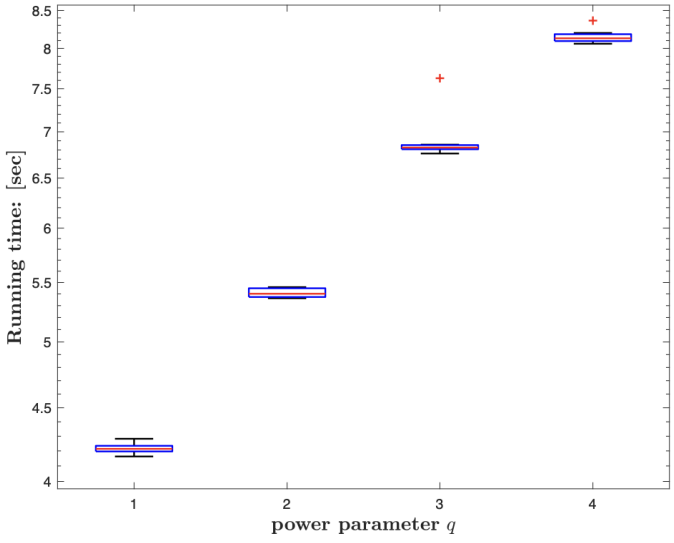
- Replace $Y = A\Omega$ with $Y = W\Omega$
- Complicates the error bound – will show it later

Target rank selection

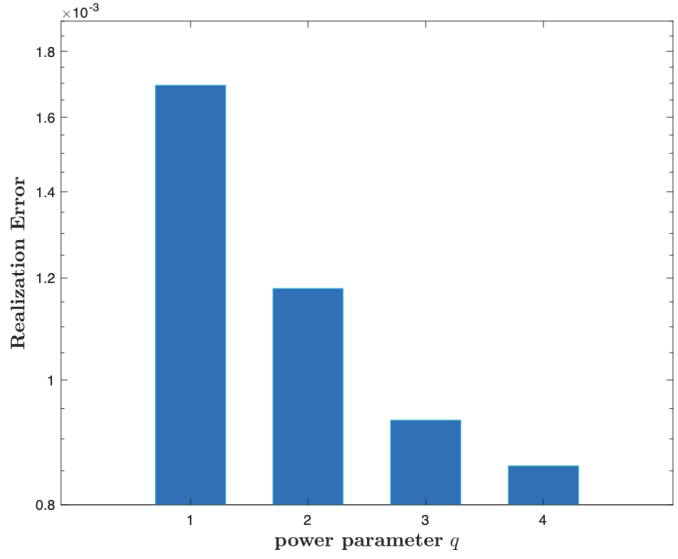
Straight forward to adaptively construct the basis vectors Q until tolerance is met. See **[HMT]** for details.

Additional Numerical Experiments

Power Iterations



(c) Running time of stochastic Ho-Kalman Algorithm with varying power parameter q. The oversampling parameter l is 10.



(d) Realization error of the stochastic Ho-Kalman Algorithm with varying power parameter q. The oversampling parameter l is 10.

- Example 4: $(n, m, p, T) = (100, 80, 50, 500)$