Doubly Random Parallel Stochastic Methods for Large-Scale Learning

Aryan Mokhtari, Alec Koppel, and Alejandro Ribeiro
University of Pennsylvania, Philadelphia, PA

American Control Conference

Boston, MA, July 7, 2016

Large-Scale Parameter Estimation

- Learning $\Rightarrow$ params $\mathbf{x}^* \in \mathbb{R}^p$ that minimize expected risk $F(\mathbf{x})$
- $f : \mathbb{R}^p \to \mathbb{R}$ $\Rightarrow$ convex loss, quantifies merit of statistical model

  $\Rightarrow \boldsymbol{\theta}$ is random variable representing data

$$\mathbf{x}^* := \underset{\mathbf{x}}{\operatorname{argmin}}\, F(\mathbf{x}) := \underset{\mathbf{x}}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{x}, \boldsymbol{\theta})]$$

Large-Scale Parameter Estimation

- Learning $\Rightarrow$ params $\mathbf{x}^* \in \mathbb{R}^p$ that minimize expected risk $F(\mathbf{x})$
- $f : \mathbb{R}^p \to \mathbb{R}$ $\Rightarrow$ convex loss, quantifies merit of statistical model
  $\Rightarrow$ θ is random variable representing data
- Suppose N i.i.d. samples θ_n of stationary dist. of θ
  $\Rightarrow$ $f_n(\mathbf{x}) := f(\mathbf{x}, \theta_n)$ loss associated with n-th sample

$$\mathbf{x}^* := \underset{\mathbf{x}}{\operatorname{argmin}} F(\mathbf{x}) := \underset{\mathbf{x}}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} f_n(\mathbf{x})$$

- Example problems:
  $\Rightarrow$ support vector machines
  $\Rightarrow$ logistic regression
  $\Rightarrow$ matrix completion

Large-Scale Parameter Estimation

- Learning $\Rightarrow$ params $\mathbf{x}^* \in \mathbb{R}^p$ that minimize expected risk $F(\mathbf{x})$
- $f : \mathbb{R}^p \to \mathbb{R}$ $\Rightarrow$ convex loss, quantifies merit of statistical model
  $\Rightarrow$ θ is random variable representing data
- Suppose N i.i.d. samples θ_n of stationary dist. of θ
  $\Rightarrow$ $f_n(\mathbf{x}) := f(\mathbf{x}, \theta_n)$ loss associated with n-th sample

$$\mathbf{x}^* := \operatorname*{argmin}_{\mathbf{x}} F(\mathbf{x}) := \operatorname*{argmin}_{\mathbf{x}} \frac{1}{N} \sum_{n=1}^{N} f_n(\mathbf{x})$$

- Example problems:
  $\Rightarrow$ support vector machines
  $\Rightarrow$ logistic regression
  $\Rightarrow$ matrix completion
- Focus: feature dimension p and sample size N are huge-scale
  $\Rightarrow$ e.g., $p = \mathcal{O}(N)$

Background

- Optimization for large N: stochastic approximation
  - $\Rightarrow$ stochastic first-order (SGD, SAG, SVRG, etc.)
  - $\Rightarrow$ stochastic quasi-Newton (RES, SQN, oLBFGS)
- Optimization for large p: block coordinate methods
  - $\Rightarrow$ block coordinate descent
  - $\Rightarrow$ stochastic coordinate descent
- Optimization for large p *and* N
  - $\Rightarrow$ asynchronous block SGD w/ sparsity (Hogwild!)
  - $\Rightarrow$ This work: operate on random subsets of features & samples
  - $\Rightarrow$ no block separability in gradient computations as in Hogwild!

Large Sample Size Setting

- Recall the problem

$$\mathbf{x}^* := \underset{\mathbf{x}}{\operatorname{argmin}}\, F(\mathbf{x}) := \underset{\mathbf{x}}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{x}, \boldsymbol{\theta})]$$

- N is very large $\Rightarrow$ can't afford gradient or Newton methods
  $\Rightarrow$ solution: stochastic methods

- Classically solved with stochastic gradient method

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \gamma^t \nabla_{\mathbf{x}} f(\mathbf{x}^t, \boldsymbol{\theta}^t)$$

  $\Rightarrow$ descend using stochastic gradient rather than true gradient

  $\Rightarrow$ breaks bottleneck in N $\Rightarrow$ operate on one sample at a time

- Nice analytical properties for convex and strongly convex cases

  $\Rightarrow$ Converges sublinearly in mean, converges to optimum a.s.

High-Dimensional Parameter Spaces

- Suppose the feature dimension $p = \mathcal{O}(N)$. For this case:
  - $\Rightarrow$ Computational complexity per iteration $\mathcal{O}(p)$ $\Rightarrow$ very large!
  - $\Rightarrow$ Stochastic gradient update is computationally demanding
- Focus: break bottleneck in p in stochastic approx. methods
- We do this by partitioning vector $\mathbf{x}$ into B blocks of size p_b
  - $\Rightarrow$ block stochastic approximation on random subsets of blocks
  - $\Rightarrow$ executed by a collection of $\mathcal{I}$ parallel processors
- Results in a doubly stochastic parallel method (RAPSA)
- Propose asynchronous extensions
- Establish convergence properties comparable to SGD

Parallelizing over Features

- Break regressor $\mathbf{x}$ into B distinct blocks $\mathbf{x}_b$ of size $p_b << p$
- Associate w/ each block an i.i.d. sample of random variable θ:

$$\begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_B \end{bmatrix} \longleftrightarrow \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_B \end{bmatrix}$$

Parallelizing over Features

- Break regressor $\mathbf{x}$ into B distinct blocks $\mathbf{x}_b$ of size $p_b << p$
- Associate w/ each block an i.i.d. sample of random variable θ:

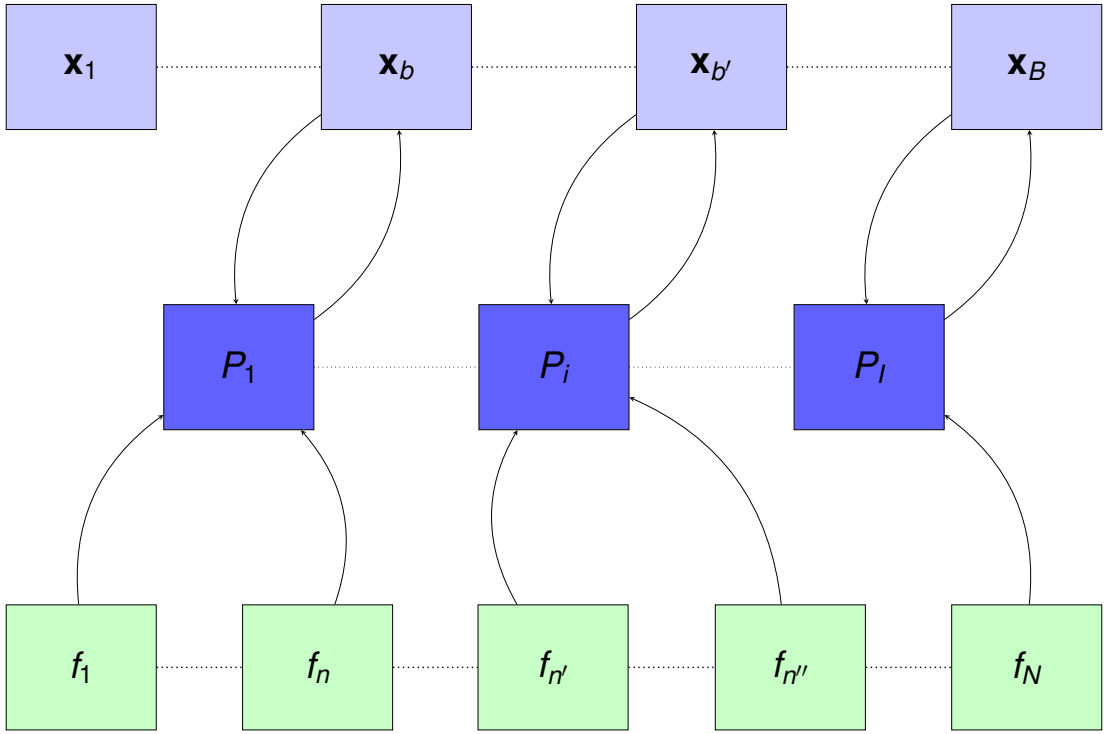
$$\begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_B \end{bmatrix} \longleftrightarrow \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_B \end{bmatrix}$$

- Collection of $\mathcal{I} \ll B$ processors work in parallel

  $\Rightarrow$ Each processor randomly chooses a block $\mathbf{x}_b$
- Rather than parallel SGD, gradient update on only *some* blocks

  $\Rightarrow$ Processor P_i updates block $\mathbf{x}_b$ w/ stochastic subset of data.
- Advantages of both stochastic coordinate descent and SGD

RAPSA: Random Parallel Stochastic Algorithm



- processor P_i picks block $b_i^t \in [B]$ at random, sample subset θ_i^t
- Executes block SGD

$$\mathbf{x}_b^{t+1} = \mathbf{x}_b^t - \gamma^t \nabla_{\mathbf{x}_b} f(\mathbf{x}^t, \theta_i^t) \,, \quad b = b_i^t.$$

- Block selection $\Rightarrow$ no processors operate on the same block
- Processors use shared memory on common time index.

Extension: Asynchronous RAPSA

- Up to now, parallel processors operate on common time index
  - $\Rightarrow$ all nodes must wait for the one with longest clock time
  - $\Rightarrow$ bottleneck is substantial in applications (e.g. deep learning)
- This common clock requirement is unnecessary
- Let's consider case where processors don't wait for one another
  - $\Rightarrow$ asynchronous parallel algorithm
  - $\Rightarrow$ as long as the amount of asynchronicity τ is bounded by Δ

Extension: Asynchronous RAPSA

- Assume at time t only one processor executes an update.

  $\Rightarrow$ break ties at random
- Asynchronous RAPSA:

  $\Rightarrow$ Processor P_i picks uniformly $b_i^t \in [B]$ at random at time t
- Execute stochastic descent step using delayed gradient

$$\mathbf{x}_b^{t+1} = \mathbf{x}_b^t - \gamma^t \nabla_{\mathbf{x}_b} f(\mathbf{x}^{t-\tau}, \mathbf{\Theta}_i^{t-\tau}) \qquad b = b_i^t.$$

- $\tau \Rightarrow$ delay due to asynchronicity

  $\Rightarrow$ $t - \tau$ is the last time block b was updated
- Similar architecture to Hogwild!, but $\mathbf{\Theta}$ not assumed sparse

Technical Conditions

- Instantaneous objective functions $f(\mathbf{x}, \theta)$ $\Rightarrow$ differentiable
- Average function $F(\mathbf{x}) = \mathbb{E}_{\theta}[(\mathbf{x}, \theta)]$ $\Rightarrow$ m-strongly convex
- Average objective gradients $\nabla F(\mathbf{x})$ $\Rightarrow$ M-Lipschitz continuous,
  $\Rightarrow$ For all $\mathbf{x}, \hat{\mathbf{x}} \in \mathbb{R}^p$, it holds

$$\|\nabla F(\mathbf{x}) - \nabla F(\hat{\mathbf{x}})\| \leq M \|\mathbf{x} - \hat{\mathbf{x}}\|.$$

- Stochastic gradient has finite variance
  $\Rightarrow$ for a constant K, all $\mathbf{x}$, we have

$$\mathbb{E}_{\theta}\left[\|\nabla f(\mathbf{x}^t, \theta^t)\|^2 \,\middle|\, \mathbf{x}^t\right] \leq K.$$

- Standard conditions in stochastic approximation literature

Convergence of RAPSA

Theorem

(i) The synchronous RAPSA sequence $\{\mathbf{x}^t\}$*, with diminishing step-size rules* $\gamma^t = \mathcal{O}(1/t)$ *converges a.s. to optimal* $\mathbf{x}^*$*,*

$$\lim_{t\to\infty} \|\mathbf{x}^t - \mathbf{x}^*\|^2 = 0 \qquad a.s.$$

(ii) If step-size is such that $\gamma^t := \gamma^0 T^0/(t+T^0)$ *and* $2mr\gamma^0 T^0 > 1$*, then the error sequence* $\mathbb{E}[F(\mathbf{x}^t) - F(\mathbf{x}^*)]$ *converges to null as* $\mathcal{O}(1/t)$*,*

$$\mathbb{E}[F(\mathbf{x}^t) - F(\mathbf{x}^*)] \leq \frac{C}{t+T^0},$$

$\Rightarrow$ *Constant* C *is defined as*

$$C = \max\left\{\frac{rMK(\gamma^0 T^0)^2}{4mr\gamma^0 T^0 - 2},\ T^0(F(\mathbf{x}^0) - F(\mathbf{x}^*))\right\}.$$

Convergence of RAPSA

- Almost sure convergence to optimum using diminishing step-size

- A.s. convergence to nbhd. of optimum w/ constant step-size
- Linear convergence on average to optimal objective

  $\Rightarrow$ provided step-size is chosen as sufficiently small constant

Convergence of Asynchronous RAPSA

Theorem

Suppose the level of asynchronicity satisfies $\tau \leq \Delta$. *(i) Asynchronous RAPSA seq.* $\{\mathbf{x}^t\}$ *converges a.s. to optimal* $\mathbf{x}^*$

$\Rightarrow$ *using diminishing step-size rules* $\gamma^t = \mathcal{O}(1/t)$, *i.e.*

$$\liminf_{t\to\infty} \|\mathbf{x}^t - \mathbf{x}^*\|^2 \;=\; 0 \qquad a.s.$$

(ii) If step-size satisfies $\gamma^t := \gamma^0 T^0/(t+T^0)$ *with* $2mr\gamma^0 T^0 > 1$, *then expected error sequence* $\mathbb{E}F(\mathbf{x}^t) - F(\mathbf{x}^*)$ *converges to null as* $\mathcal{O}(1/t)$,

$$\mathbb{E}F(\mathbf{x}^t) - F(\mathbf{x}^*) \;\leq\; \frac{C}{t+T^0},$$

$\Rightarrow$ *Constant* C *is defined as in synchronous case.*

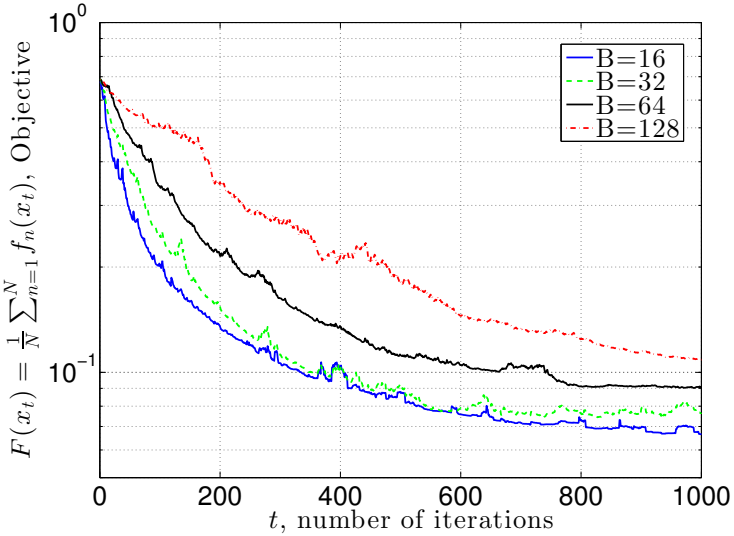
Logistic Regression Example

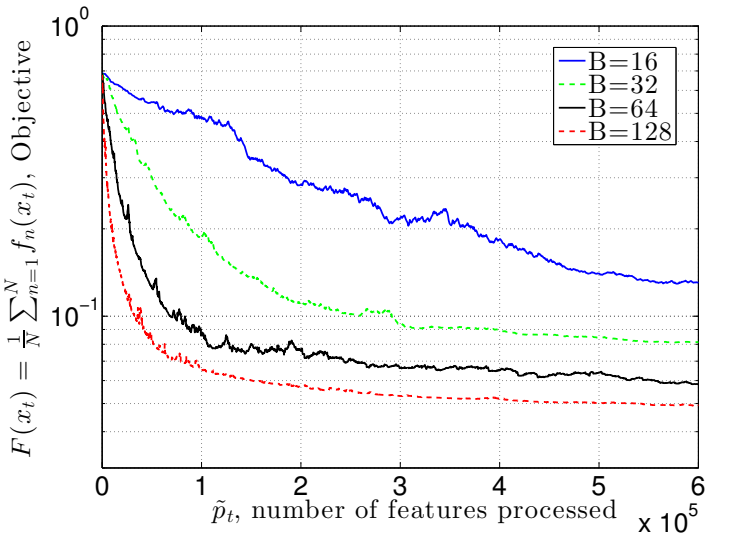
- $\mathbf{z} \in \mathbb{R}^p \Rightarrow$ feature vector encoding image pixel intensities
  $\Rightarrow$ label $y \in \{-1, 1\} \Rightarrow$ whether image contains digit 0 or 8
- Learning a hand-written digit detector $\Rightarrow$ logistic regression
  $\Rightarrow \mathbf{x} \in \mathbb{R}^p \Rightarrow$ relate samples $\mathbf{z}_n \in \mathbb{R}^p$ to labels $y_n \in \{-1, 1\}$
- ERM problem associated with training set $\mathcal{T} = \{(\mathbf{z}_n, y_n)\}_{n=1}^N$
  $\Rightarrow$ Find $\mathbf{x}^*$ as ℓ_2 regularized maximum likelihood estimate

$$\mathbf{x}^* := \underset{\mathbf{x} \in \mathbb{R}^p}{\operatorname{argmin}} \frac{\lambda}{2}\|\mathbf{x}\|^2 + \frac{1}{N}\sum_{n=1}^{N} \log(1 + \exp(-y_n \mathbf{x}^T \mathbf{z}_n)) \ ,$$

  $\Rightarrow$ Logistic transformation of odds ratio for label being -1 or 1
- We use an $N = 1.76 \times 10^4$ subset of MNIST with labels 0 and 8
  $\Rightarrow$ Feature vectors $\mathbf{z}_n \in \mathbb{R}^p$ are $p = 28^2 = 784$ pixel images

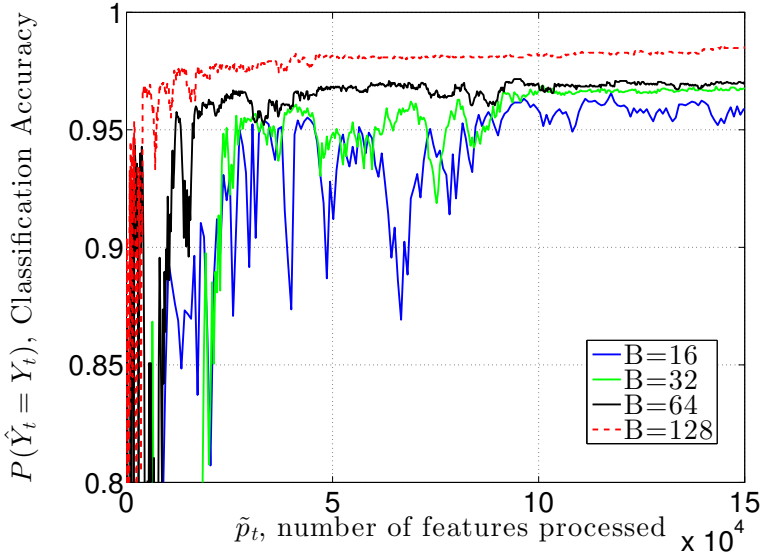
Logistic Regression Example

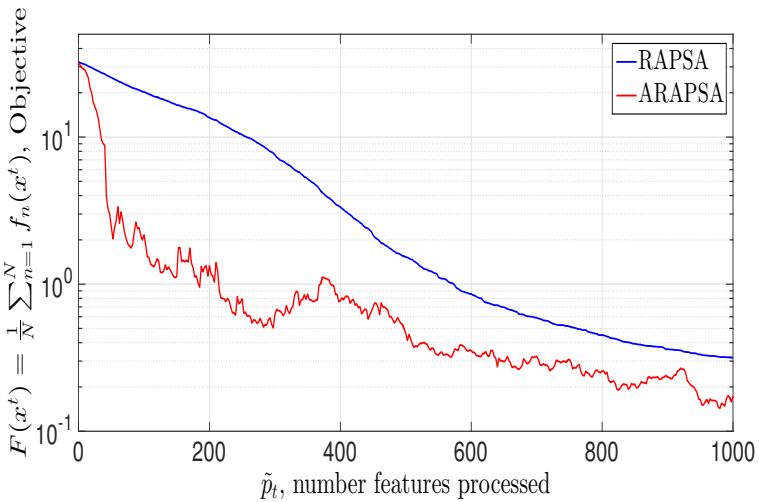




- RAPSA on binary subset of MNIST
  - $\Rightarrow$ hybrid step-size $\gamma^t = \min(10^{-2.5}, 10^{-2.5}\tilde{T}_0/t)$, $\tilde{T}_0 = 525$
  - $\Rightarrow$ no mini-batching $L = 1$.
  - $\Rightarrow$ block size $p_b = p/4$
- Define $\tilde{p}_t = prtL$
  - $\Rightarrow$ no. of features processed per iteration
- Performance w.r.t. prop. of **x** updated
  - $\Rightarrow$ faster when full **x** is used with iteration t
  - $\Rightarrow$ faster with *fewer* entries of **x** with $\tilde{p}_t$

Logistic Regression Example





- Consider a test set of size $\tilde{N} = 5.88 \times 10^3$
- Classification accuracy $\approx 95\%$
  - $\Rightarrow$ across different values of B
  - $\Rightarrow$ using fewest entries of $\mathbf{x}$ is best

- Now we fix $B = 64$
  - $\Rightarrow$ $1/4$ of $\mathbf{x}$ is updated per iteration
- mini-batch size $L = 10$, step-size $\epsilon = 10^{-1}$
- "Accelerated" RAPSA $\approx 3x$ RAPSA rate
  - $\Rightarrow$ ARAPSA $\Rightarrow$ block-wise oL-BFGS

Conclusions

- Classic stochastic approximation $\Rightarrow$ can't handle $p = \mathcal{O}(N)$
- RAPSA breaks bottleneck in p

  $\Rightarrow$ Operates on random subsets of samples and features
- Can be implemented on a parallel computing architecture
- No coordination among distinct computing nodes required
- Quasi-Newton extension $\Rightarrow$ empirically superior convergence
- Convergence of synchronous and asynchronous RAPSA

  $\Rightarrow$ Under standard technical conditions
- Benefits of both stochastic coordinate descent and SGD
- Developing GPU implementation $\Rightarrow$ computational speedup

References

- A. Mokhtari, A. Koppel, and A. Ribeiro, “Doubly Random Parallel Stochastic Methods for Large Scale Learning,” American Control Conference, July. 2016.
- A. Mokhtari, A. Koppel, and A. Ribeiro, “A Class of Parallel Doubly Stochastic Algorithms for Large-Scale Learning,” Journal of Machine Learning Research (Submitted), June. 2016. [Preprint on ArXiv]

`http://seas.upenn.edu/~akoppel/`

Extension: Accelerated RAPSA

- First-order stochastic approximation methods $\Rightarrow$ converge slowly
- In stochastic setting, Newton's method impractical
  $\Rightarrow$ requires inverting Hessian $\mathbf{H} = \nabla^2 F$, an $p \times p$ dim. matrix
  $\Rightarrow$ Quasi-Newton methods approximate this Hessian inverse
- We develop an online block-coordinate Quasi-Newton method
  $\Rightarrow \mathcal{I}$ processors execute stochastic approx. updates in parallel

Extension: Accelerated RAPSA

- First-order stochastic approximation methods $\Rightarrow$ converge slowly
- In stochastic setting, Newton's method impractical
  $\Rightarrow$ requires inverting Hessian $\mathbf{H} = \nabla^2 F$, an $p \times p$ dim. matrix
  $\Rightarrow$ Quasi-Newton methods approximate this Hessian inverse
- We develop an online block-coordinate Quasi-Newton method
  $\Rightarrow$ $\mathcal{I}$ processors execute stochastic approx. updates in parallel
- Consider RAPSA update at processor $i \in \{1, \dots, I\}$
  $\Rightarrow$ $\mathcal{I}$ selects block index $b_i^t \in \{1, \dots, B\}$ uniformly at random

$$\mathbf{x}_b^{t+1} = \mathbf{x}_b^t - \gamma^t \nabla_{\mathbf{x}_b} f(\mathbf{x}^t, \mathbf{\Theta}_i^t) \ , \quad b = b_i^t .$$

- Modify stochastic descent step by "pre-conditioning" matrix $\hat{\mathbf{B}}_b^t$
  $\Rightarrow$ $\hat{\mathbf{B}}_b^t \approx [\nabla^2_{\mathbf{x}_b} F(\mathbf{x}_b^t)]^{-1}$ in a certain sense

Extension: Accelerated RAPSA

- First-order stochastic approximation methods $\Rightarrow$ converge slowly
- In stochastic setting, Newton's method impractical
  $\Rightarrow$ requires inverting Hessian $\mathbf{H} = \nabla^2 F$, an $p \times p$ dim. matrix
  $\Rightarrow$ Quasi-Newton methods approximate this Hessian inverse
- We develop an online block-coordinate Quasi-Newton method
  $\Rightarrow$ $\mathcal{I}$ processors execute stochastic approx. updates in parallel
- Accelerated RAPSA (ARAPSA): processor $i \in \{1, \dots, I\}$
  $\Rightarrow$ selects block index $b_i^t \in \{1, \dots, B\}$ uniformly at random

$$\mathbf{x}_b^{t+1} = \mathbf{x}_b^t - \gamma^t \hat{\mathbf{B}}_b^t \nabla_{\mathbf{x}_b} f(\mathbf{x}^t, \mathbf{\Theta}_i^t) \ , \quad b = b_i^t.$$

- $\hat{\mathbf{B}}_b^t \approx [\nabla^2_{\mathbf{x}_b} F(\mathbf{x}_b^t)]^{-1}$ is block Hessian inverse approximation

Hessian Approximation

- $\hat{\mathbf{H}}_b^t$ is block Hessian approximation, $\hat{\mathbf{B}}_b^t = [\hat{\mathbf{H}}_b^t]^{-1}$
- Specify this matrix by considering gradient and var. variations

$$\mathbf{v}_b^t = \mathbf{x}_b^{t+1} - \mathbf{x}_b^t, \qquad \hat{\mathbf{r}}_b^t = \nabla_{\mathbf{x}_b} f(\mathbf{x}^{t+1}, \mathbf{\Theta}_b^t) - \nabla_{\mathbf{x}_b} f(\mathbf{x}^t, \mathbf{\Theta}_b^t).$$

- True Hessian $\mathbf{H}_b^t$ associated w/ block var. $\mathbf{x}_b$

  $\Rightarrow$ has inverse which satisfies secant condition $(\mathbf{H}_b^t)^{-1}\mathbf{v}_b^t = \hat{\mathbf{r}}_b^t$

  $\Rightarrow$ "shouldn't change much" $\Rightarrow$ measured via differential entropy

$$\begin{aligned} \hat{\mathbf{H}}_b^{t+1} = \text{argmin} \quad & \text{tr}(\hat{\mathbf{H}}_b^t)^{-1}\mathbf{Z}) - \log\det(\hat{\mathbf{H}}_b^t)^{-1}\mathbf{Z}) - n \\ \text{s. t.} \quad & \mathbf{Z}\mathbf{v}_b^t = \hat{\mathbf{r}}_b^t, \quad \mathbf{Z} \succeq \mathbf{0} \end{aligned}$$

- Secant condition interpretation

  $\Rightarrow$ stoch. grad. of quad. approx. of objective is similar over time

Block Online BFGS

- Block variant of Online BFGS $\Rightarrow$ approximate Hessian inverse
- Derived as closed-form solution of opt. prob. on previous slide

$$\hat{\mathbf{H}}_b^{t+1} = \hat{\mathbf{H}}_b^t + \frac{\hat{\mathbf{r}}_b^t (\hat{\mathbf{r}}_b^t)^T}{(\mathbf{v}_b^t)^T \hat{\mathbf{r}}_b^t} - \frac{\hat{\mathbf{B}}_b^t \mathbf{v}_b^t (\mathbf{v}_b^t)^T \hat{\mathbf{H}}_b^t}{(\mathbf{v}_b^t)^T \hat{\mathbf{H}}_b^t \mathbf{v}_b^t}$$

$\Rightarrow$ apply Sherman-Morrison matrix inversion Lemma to the result

$$[\hat{\mathbf{H}}_b^{t+1}]^{-1} = \hat{\mathbf{B}}_b^{t+1} = [\mathbf{Z}_b^t]^T \hat{\mathbf{B}}_b^t \mathbf{Z}_b^t + \rho_b^t \mathbf{v}_b^t (\mathbf{v}_b^t)^T$$

with scalar ρ_b^t and matrix $\mathbf{Z}_b^t$ defined as

$$\rho_b^t = \frac{1}{(\mathbf{v}_b^t)^T \mathbf{r}_b^t}, \qquad \mathbf{Z}_b^t = \mathbf{I}_{p_b} - \rho_b^t \mathbf{r}_b^t (\mathbf{v}_b^t)^T$$

- $\hat{\mathbf{B}}_b^{t+1}$ depends on $\hat{\mathbf{B}}_b^u$ for $u < t+1$

  $\Rightarrow$ at time $t+1$, must recurse over all $u < t+1$ to compute $\hat{\mathbf{B}}_b^{t+1}$

  $\Rightarrow$ Let's truncate update at $t+1$ to only past τ iterations

BoL-BFGS: Block Online Limited Memory BFGS

- $\tau \Rightarrow$ memory for block online Limited Memory BFGS (oL-BFGS)

  $\Rightarrow$ use past τ pairs of curvature information $\{\mathbf{v}_b^u, \mathbf{r}_b^u\}_{u=t-\tau}^{t-1}$

  $\Rightarrow$ Approximate matrix $\hat{\mathbf{B}}_b^t$ is computed by initializing as

$$\hat{\mathbf{B}}_b^{t,0} := \eta_b^t \mathbf{I}\,, \qquad \eta_b^t := \frac{(\mathbf{v}_b^{t-1})^T \hat{\mathbf{r}}_b^{t-1}}{\|\hat{\mathbf{r}}_b^{t-1}\|^2},$$

- Approx. Hessian inverse $\Rightarrow \tau$ recursive applications of update

$$\hat{\mathbf{B}}_b^{t,u+1} = (\hat{\mathbf{Z}}_i^{t-\tau+u})^T \hat{\mathbf{B}}_b^{t,u} (\hat{\mathbf{Z}}_b^{t-\tau+u}) + \hat{\rho}_b^{t-\tau+u} (\mathbf{v}_b^{t-\tau+u})(\mathbf{v}_b^{t-\tau+u})^T,$$

- Matrices $\hat{\mathbf{Z}}_b^{t-\tau+u}$, constant $\hat{\rho}_b^{t-\tau+u}$ for $u = 0, \ldots, \tau - 1$ defined as

$$\hat{\rho}_b^{t-\tau+u} = \frac{1}{(\mathbf{v}_b^{t-\tau+u})^T \hat{\mathbf{r}}_b^{t-\tau+u}} \text{ and } \hat{\mathbf{Z}}_b^{t-\tau+u} = \mathbf{I} - \hat{\rho}_b^{t-\tau+u} \hat{\mathbf{r}}_b^{t-\tau+u} (\mathbf{v}_b^{t-\tau+u})^T.$$