2020 IEEE International Conference on Acoustics, Speech, and Signal Processing

Adaptive Distributed Stochastic Gradient Descent for Minimizing Delay in the Presence of Stragglers

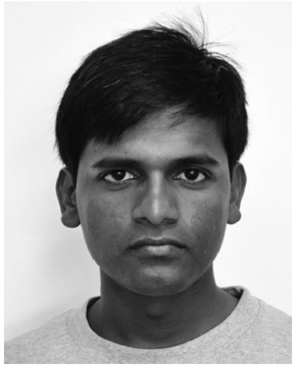
Serge Kas Hanna

Email: serge.k.hanna@rutgers.edu
Website: tiny.cc/serge-kas-hanna

RUTGERS THE STATE UNIVERSITY OF NEW JERSEY

Joint work with

Rawad Bitar, TUM



Parimal Parag, IISC

Venkat Dasari, US Army RL

Salim El Rouayheb, Rutgers

Distributed Computing and Applications

The Age of Big Data

Internet of Things (IoT)

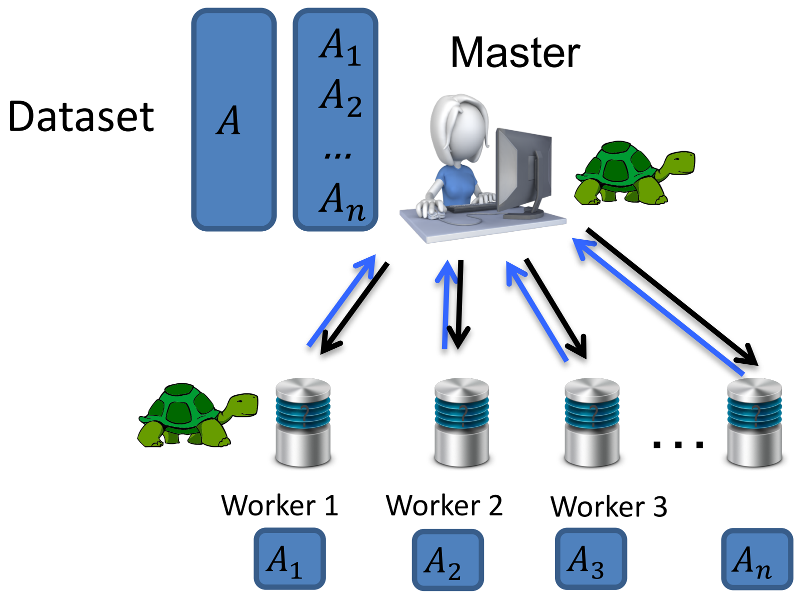
Cloud computing

Outsourcing computations to companies

Distributed Machine Learning

Focus of this talk: Distributed Machine Learning

Speeding Up Distributed Machine Learning

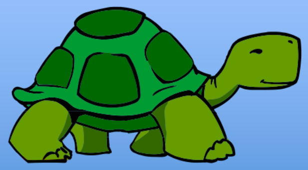


Master wants to run a ML algorithm on a **large** dataset A

Learning process can be made **faster** by outsourcing computations to worker nodes

Workers who perform **local computations** and communicate results back to master

Challenge:



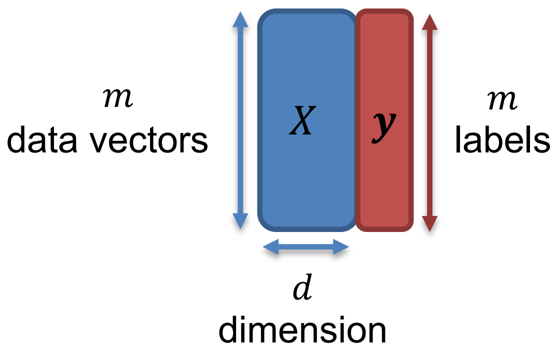
Stragglers

Stragglers: slow or unresponsive workers can significantly delay the learning process

Master is as fast as the slowest worker!

Distributed Machine Learning

- Master has dataset $X \in \mathbb{R}^{m \times d}$, labels $\boldsymbol{y} \in \mathbb{R}^{\boldsymbol{m}}$ and wants to learn a model $\boldsymbol{w}^* \in \mathbb{R}^d$ that best represents $\boldsymbol{y}$ as a function of X



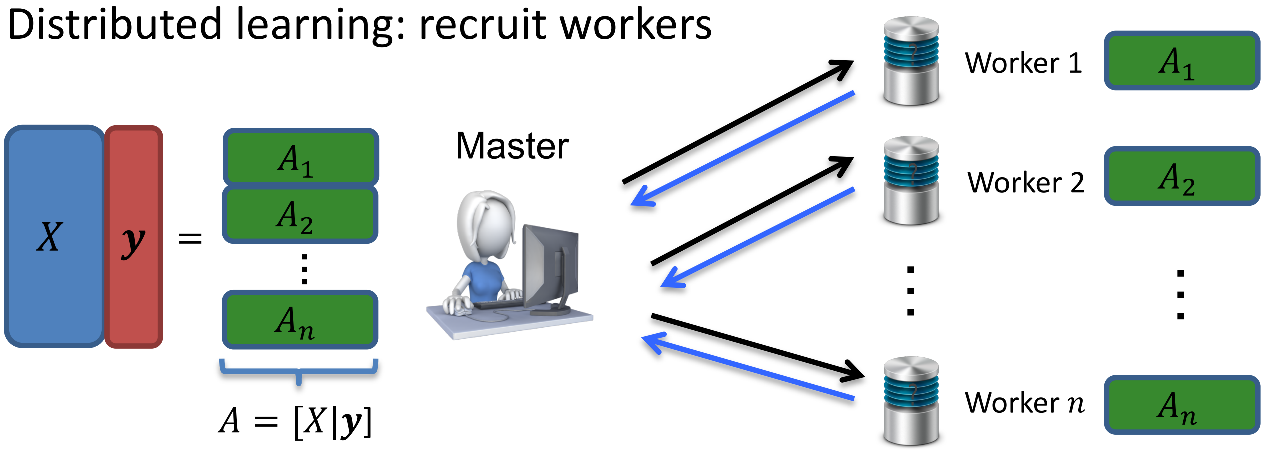
Optimization problem

Find $\boldsymbol{w}^* \in \mathbb{R}^d$ that minimizes a certain loss function F

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} F(X, \boldsymbol{y}, \boldsymbol{w})$$

- When the dataset is **large** $(m \gg)$, computation is a bottleneck

- Distributed learning: recruit workers



1) Distribute data to n workers

2) Workers compute on local data & send to master

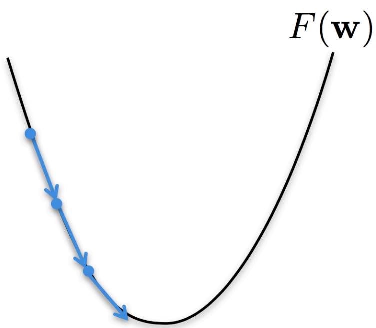
3) Master aggregates responses & updates model

GD, SGD & batch SGD

- Gradient Descent (GD), choose $\boldsymbol{w}_0$ randomly then iterate

$$\boldsymbol{w}_{j+1} = \boldsymbol{w}_j - \eta \nabla F(A, \boldsymbol{w}_j),$$

where η is the step size and ∇F is the gradient of F



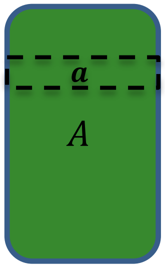
$$\boldsymbol{w}_{j+1} = \boldsymbol{w}_j - \eta \nabla F(\boldsymbol{w})$$

- When dataset A is **large**, computing $\nabla F(A, \boldsymbol{w})$ is cumbersome

- Stochastic Gradient Descent (SGD): at each iteration, update $\boldsymbol{w}_j$ based on one row of $A \in \mathbb{R}^{d+1}$ that is chosen **uniformly at random**

randomly chosen data vector from A

$$\boldsymbol{w}_{j+1} = \boldsymbol{w}_j - \eta \nabla F(\boldsymbol{a}, \boldsymbol{w}_j),$$

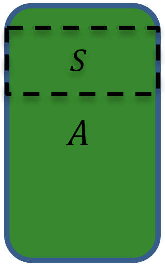


sample 1 row at random

- Batch SGD: choose a batch of $s < m$ data vectors uniformly at random

random batch of data vectors

$$\boldsymbol{w}_{j+1} = \boldsymbol{w}_j - \eta \nabla F(S, \boldsymbol{w}_j),$$

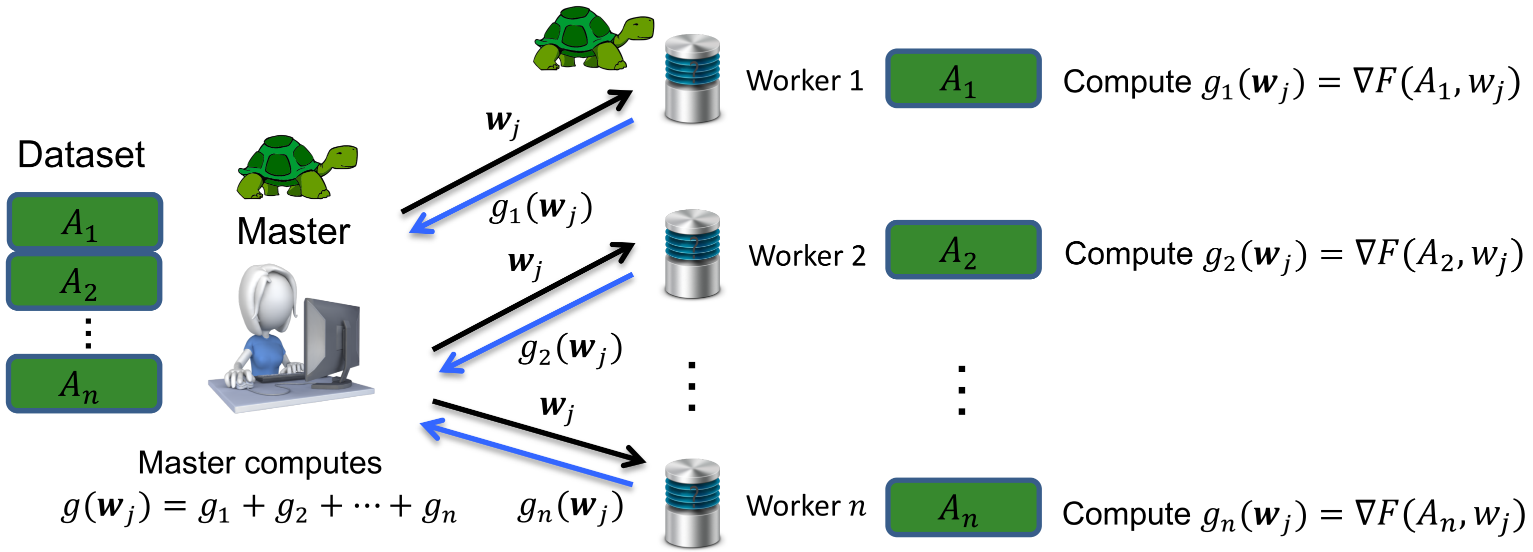


sample batch of s rows at random

- SGD & Batch SGD can converge to $\boldsymbol{w}^*$ with a higher number of iterations

Synchronous Distributed GD

- Distributed GD: each worker computes a partial gradient on its local data



- At iteration j:
  1. Master sends the current model $\boldsymbol{w}_j$ to all workers
  2. Workers compute their partial gradients and send them to the master
  3. Master aggregates the partial gradients by summing them to obtain full gradient
- Aggregation with simple summation works if ∇F is additively separable, e.g. $\mathcal{L}_2$ loss
- Straggler problem: Master is as fast as the slowest worker

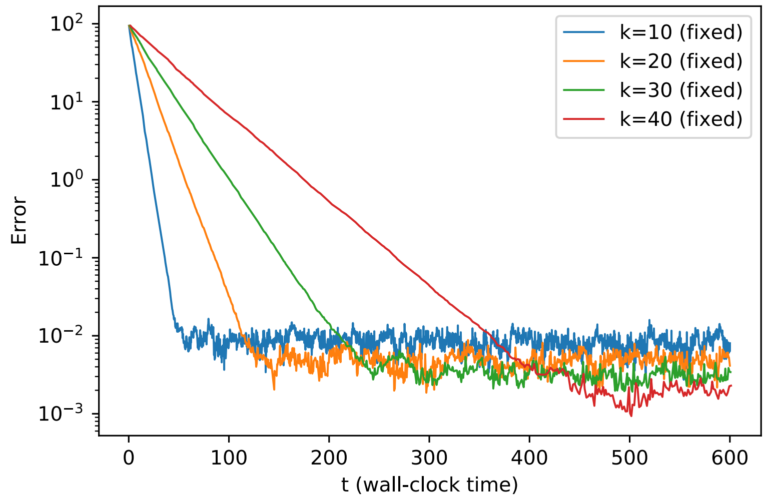
Speeding up Distributed GD: Previous Work

- Coding theoretic approach: Gradient coding [Tandon et al. ‘17], [Yu et al. ‘17], [Halbawi et al. ‘18], [Kumar et al. ‘18], …
  - Main idea: Distribute data redundantly and encode the partial gradients
  - Responses from stragglers are treated as erasures and the full gradient is decoded from responses of non-stragglers
- Approximate gradient coding: [Chen et. al ‘17], [Wang et al. ‘19], [Bitar et al. ‘19], …
  - Main idea: master does not need to compute exact gradient, e.g. SGD
  - Ignore the response of stragglers and obtain an estimate of the full gradient
  - **Fastest-$\boldsymbol{k}$ SGD**: wait for the responses of the **fastest** $k < n$ workers and ignore the responses of the $n - k$ stragglers
- Mixed Strategies: [Charles et al. ‘17], [Maity et al. ‘18], …

Fastest-k SGD

- Our question: how to choose the value of k in fastest-k SGD with fixed step size?
- Numerical example on synthetic data: linear regression, $\mathcal{L}_2$ loss function

Error vs Time of Fastest-k SGD



n = 50 workers
m = 2000 data points
d = 10 dimension
Response time of workers iid $\sim \exp(1)$

Key observation

Error-runtime trade-off: **convergence is faster** for small k but **accuracy is lower**

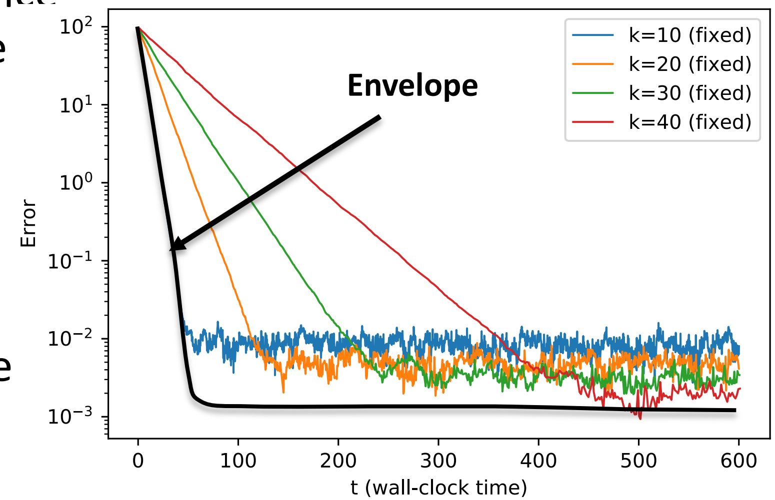
- What does theory say?

Theorem [Murata 1998]: SGD with fixed step size goes through an **exponential phase** where error decreases exponentially, then enters a **stationary phase** where $\boldsymbol{w}_j$ oscillates around $\boldsymbol{w}^*$

- Previous work on fastest-k SGD: Analysis by [Bottou et al. '18] & [Duta et al. '18] for predetermined (fixed) k

Our Contribution: Adaptive fastest-k SGD

- Our goal: speed up distributed SGD in the presence of stragglers, i.e., achieve lower error is less time



- Approach: adapt the value of k throughout the runtime to maximize time spent in exponential decrease

- Adaptive: start with smallest k and then increase k gradually every time error hits a plateau

- Challenge: in practice we do not know the error because we do not know $\boldsymbol{w}^*$

- Our results:

  1. Theoretical:
     - Derive an upper bound on the error of fastest-k SGD as a function of time
     - Determine the bound-optimal switching times
  2. Practical: Devise an algorithm for adaptive fastest-k SGD based on a statistical heuristic

Our Theoretical Results

Theorem 1 [Error vs. Time of fastest-k SGD]: Under certain assumptions on the loss function, the error of fastest-k SGD after wall-clock time t with fixed step size satisfies

$$\mathbb{E}[F(\boldsymbol{w}_t) - F(\boldsymbol{w}^*) | J(t)] \leq \frac{\eta L \sigma^2}{2cks} + (1 - \eta c)^{\frac{t}{\mu_k}(1-\epsilon)} \left(F(\boldsymbol{w}_0) - F(\boldsymbol{w}^*) - \frac{\eta L \sigma^2}{2cks} \right),$$

with high probability for large t, where $0 < \epsilon \ll 1$ is a constant error term, $J(t)$ is the number of iterations completed in time t, and μ_k is the average of the k^{th} order statistic of the random response times.

Theorem 2 [Bound-optimal switching times]: The bound optimal switching times t_k, $k = 1, \dots, n-1$, at which the master should switch from waiting for the fastest k workers to waiting for the fastest $k + 1$ workers are given by

$$\begin{aligned} t_k = t_{k-1} + \frac{\mu_k}{-\ln(1-\eta c)} [&\ln(\mu_{k+1} - \mu_k) - \ln(\eta L \sigma^2 \mu_k) \\ &+ \ln(2ck(k+1)s\left(F(\boldsymbol{w}_{t_{k-1}}) - F(\boldsymbol{w}^*)\right) - \eta L (k+1)\sigma^2] \end{aligned}$$

where $t_0 = 0$.

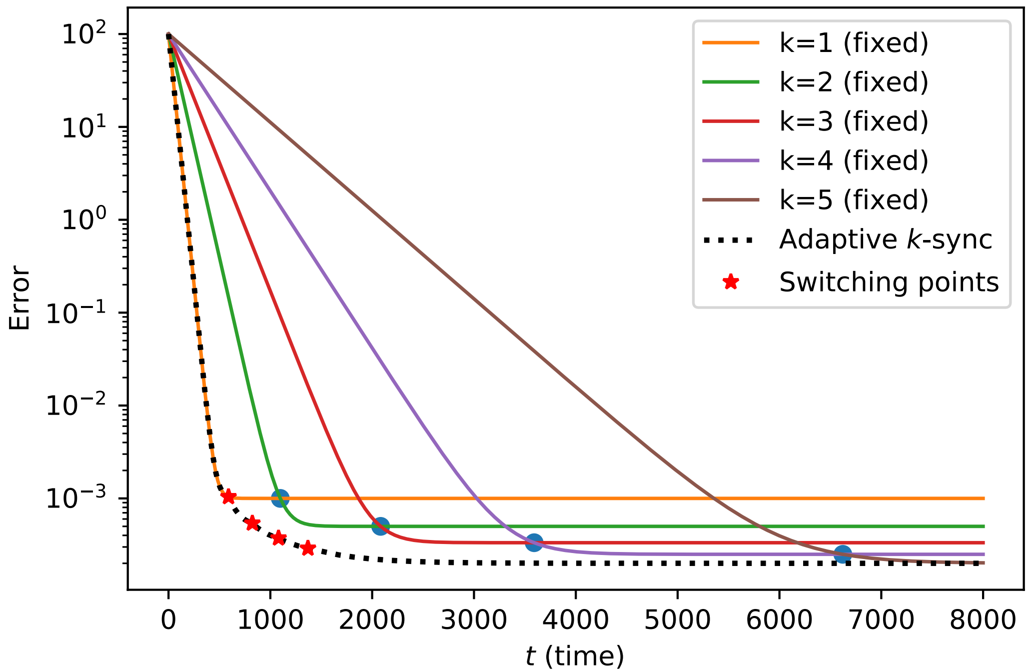
Example on Theorem 2

Theorem 2 [Bound-optimal switching times]: The bound optimal switching times t_k, ..., are given by

$$t_k = t_{k-1} + \frac{\mu_k}{-\ln(1-\eta c)} [\ln(\mu_{k+1} - \mu_k) - \ln(\eta L \sigma^2 \mu_k) + \ln(2ck(k+1)s\left(F(\mathbf{w}_{t_{k-1}}) - F(\mathbf{w}^*)\right) - \eta L(k+1)\sigma^2]$$

where $t_0 = 0$.

- Example with iid exponential response times: evaluate upper bound and apply Thm 2

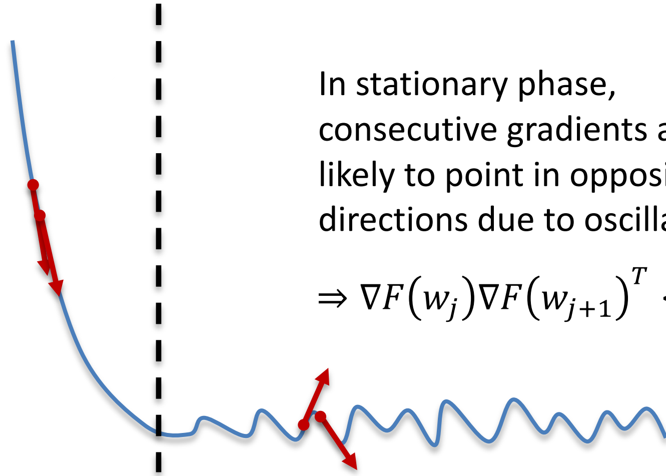


Algorithm for Adaptive fastest-k SGD

- Start with $k = 1$ and then increase k every time a phase transition is detected

- Phase transition detection: monitor the sign of consecutive gradients

Exponential phase | Stationary phase



In exponential phase, consecutive gradients are likely to point in the same direction

$$\Rightarrow \nabla F(w_j)\nabla F(w_{j+1})^T > 0$$

In stationary phase, consecutive gradients are likely to point in opposite directions due to oscillation

$$\Rightarrow \nabla F(w_j)\nabla F(w_{j+1})^T < 0$$

Stochastic approximation: [Pflug 1990]

Detect phase transition: [Chee and Toulis '18]

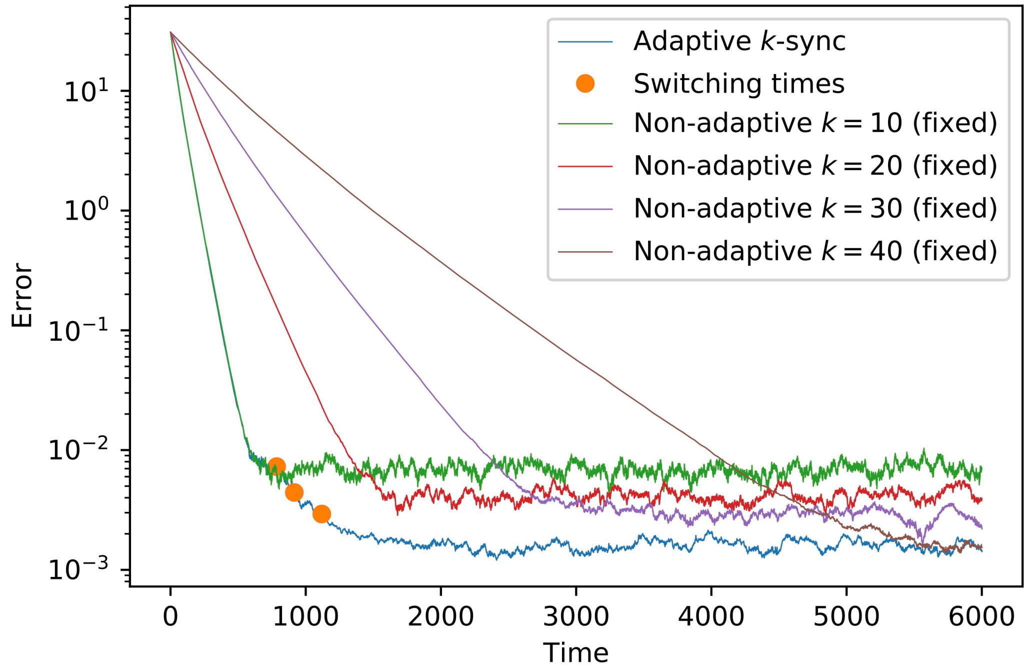
- Initialize a counter to zero and update:

$$counter = \begin{cases} counter + 1, & if\ \nabla F(w_j)\nabla F(w_{j+1})^T < 0 \\ counter - 1, & if\ \nabla F(w_j)\nabla F(w_{j+1})^T > 0 \end{cases}$$

- Declare a phase transition if counter goes above a certain threshold & increase k

Simulation Results: Non-adaptive vs Adaptive Fastest-k SGD

- Simulation on synthetic data X:
  - Generate X: pick m data vectors chosen uniformly at random from $\{1,2,\ldots,10\}^d$
  - Pick $\boldsymbol{w}^\star$ uniformly at random from $\{1,2,\ldots,100\}^d$
  - Generate labels: $\boldsymbol{y} \sim \mathcal{N}(X\boldsymbol{w}^\star, 1)$
  - Loss function: $\mathcal{L}_2$ loss (least square errors)
  - Workers' response times are iid $\sim \exp(1)$ and independent across iterations
- Simulation results on adaptive fastest-k SGD for $n = 50$ workers



$n = 50$ workers

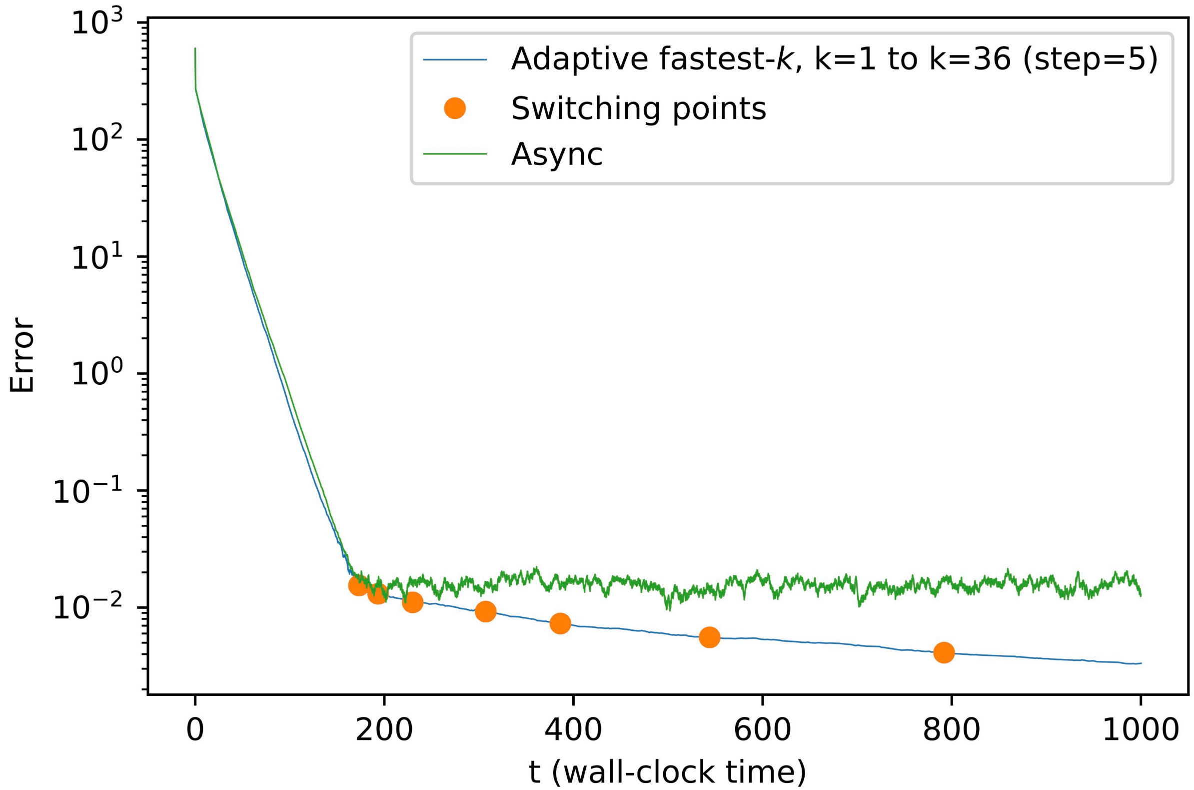
$m = 2000$ data vectors

$d = 100$ dimension

$\eta = 0.005$ step size

Simulation Results: Async vs Adaptive Fastest-k SGD

- Asynchronous Stochastic Gradient Descent: update the model $\boldsymbol{w}_j$ and send new model $\boldsymbol{w}_{j+1}$ every time a worker finishes it's partial gradient computation
- Workers who have not finished continue working on the old model
- Simulation results:



$n = 50$ workers

$m = 2000$ data vectors

$d = 100$ dimension

$\eta = 0.002$ step size

Summary and Future Work

- Speeding up distributed machine learning
  - Straggler problem
  - Adaptive fastest-k SGD for minimizing delay in the presence of stragglers
  - Theoretical results: bounds on the error & bound-optimal switching times
  - Novel realizable algorithm based on statistical heuristic
  - Numerical results showing gain with respect to non-adaptive SGD
- Future work
  - Simulations or real data (MNIST, CFAR, etc.)
  - Variable step size
  - Mixed strategies: coding + adaptivity