Unified Analysis of Asynchronous-type Algorithms

KAUST Rising Stars in AI Symposium 2024

Rustem Islamov
University of Basel

Mher Safaryan

Postdoctoral fellow

Institute of Science and Technology Austria

Dan Alistarh

Professor of Computer Science

Institute of Science and Technology Austria

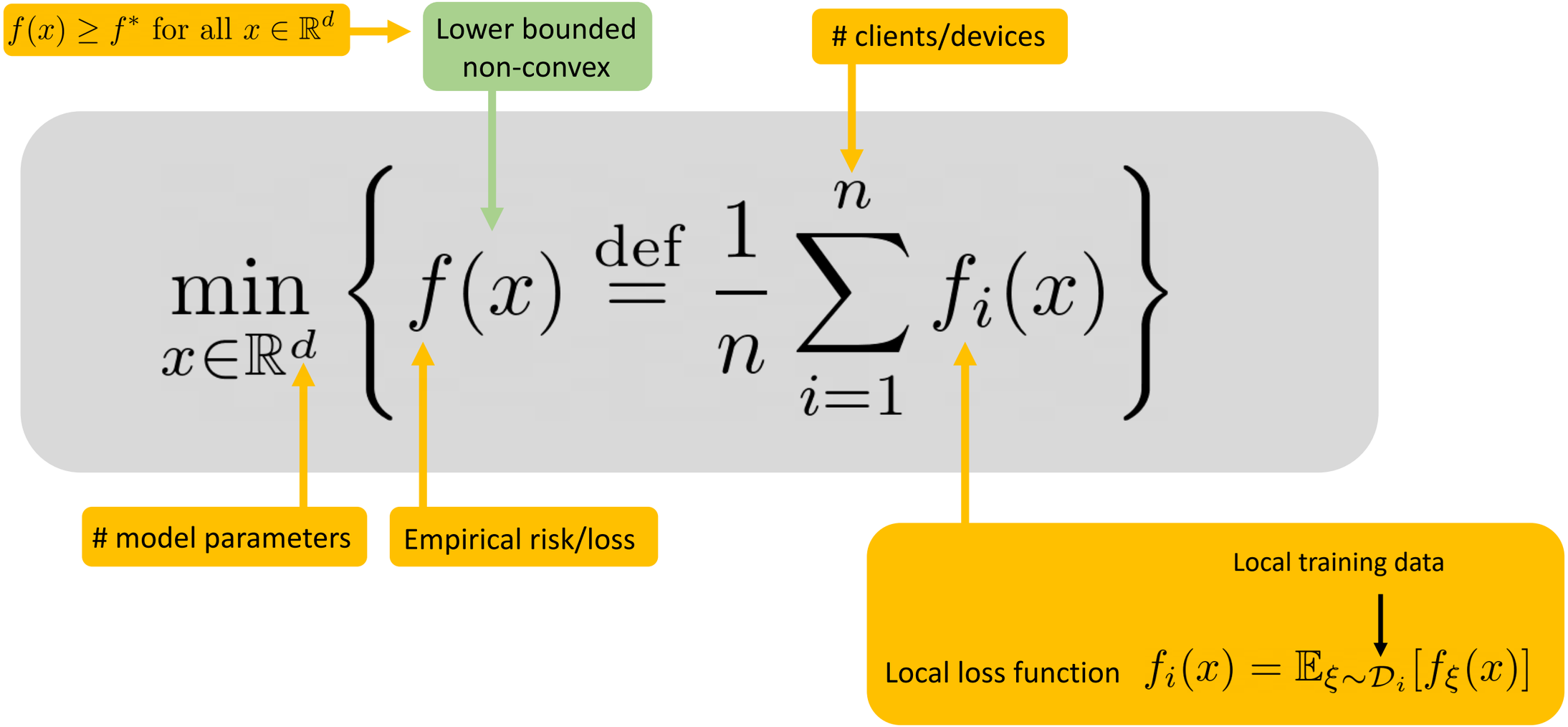
Outline

1. The Problem Formulation
2. Theoretical Analysis
3. Numerical Experiments

Outline

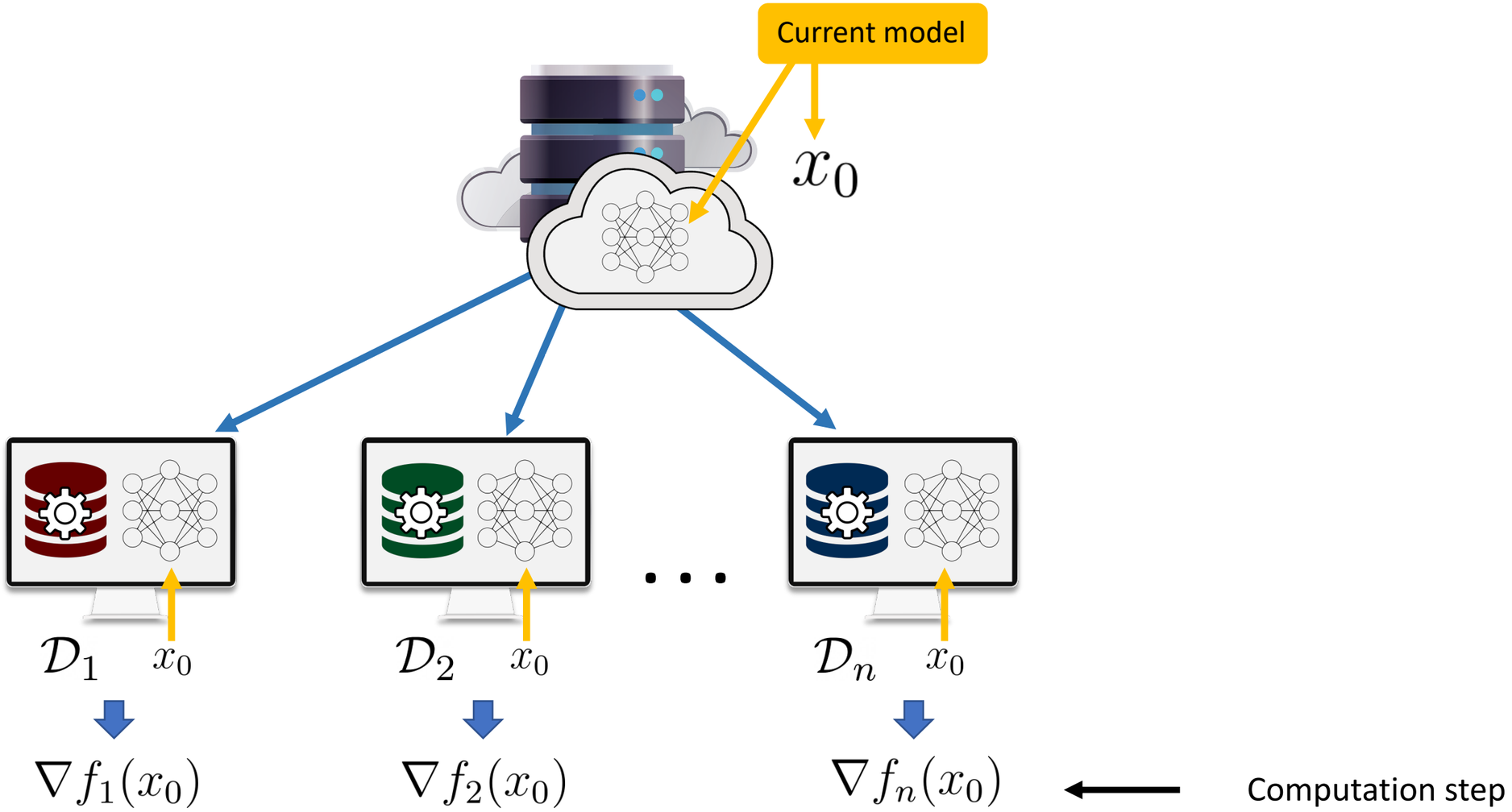
1. **The Problem Formulation**
2. Theoretical Analysis
3. Numerical Experiments

The Problem: Centralized Distributed Training

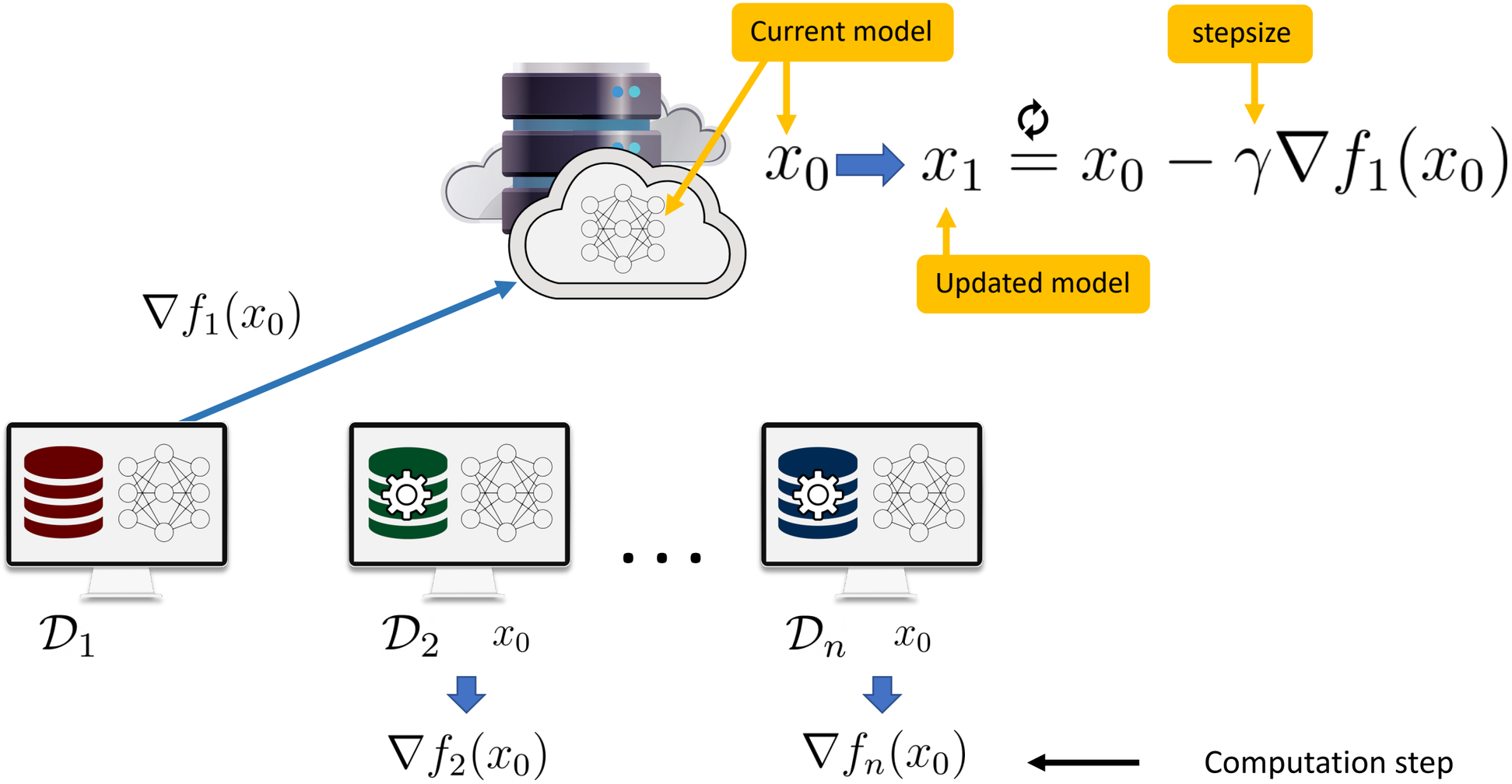


Asynchronous
Distributed Gradient Descent

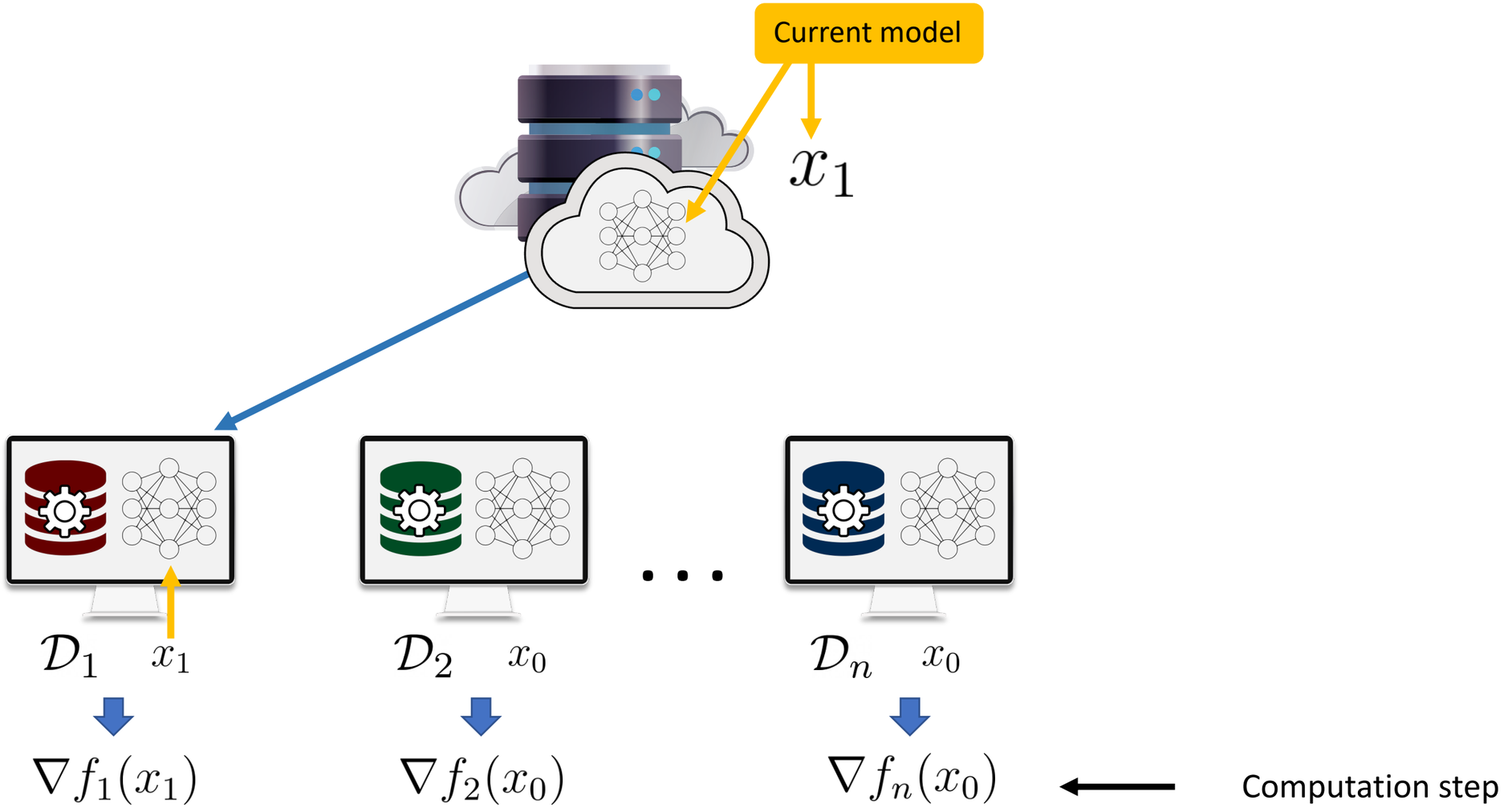
Asynchronous Distributed Gradient Descent



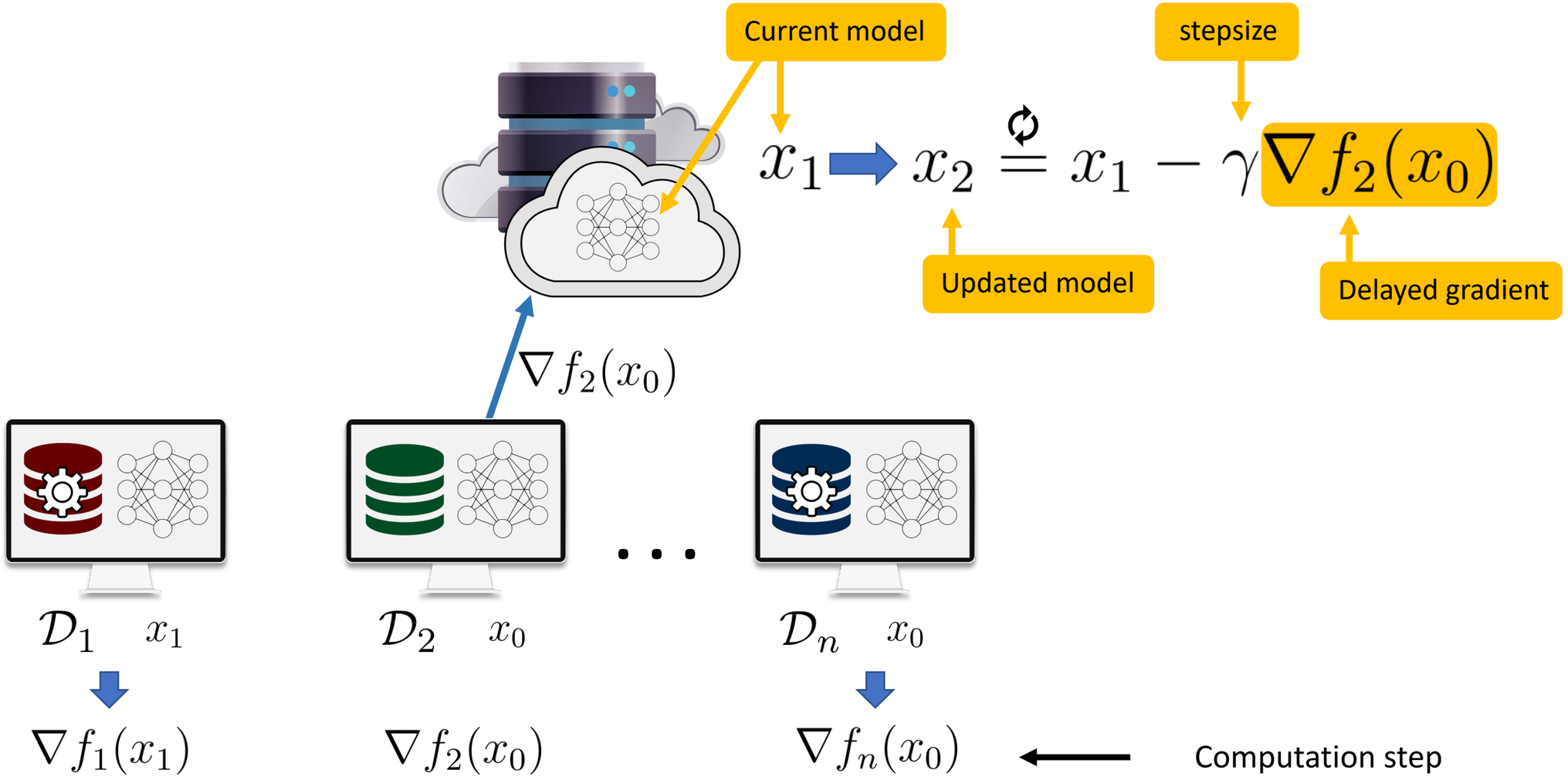
Asynchronous Distributed Gradient Descent



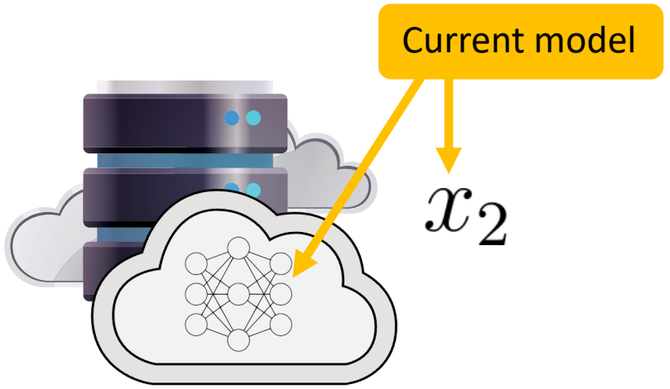
Asynchronous Distributed Gradient Descent

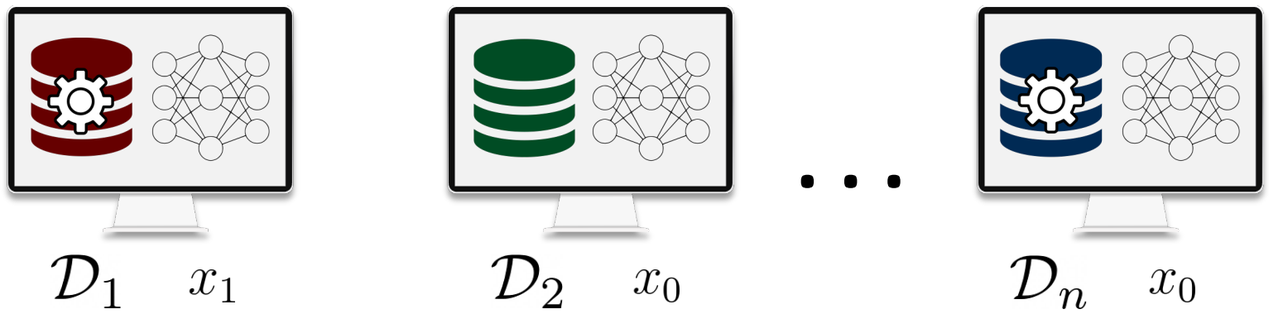


Asynchronous Distributed Gradient Descent



Asynchronous Distributed Gradient Descent





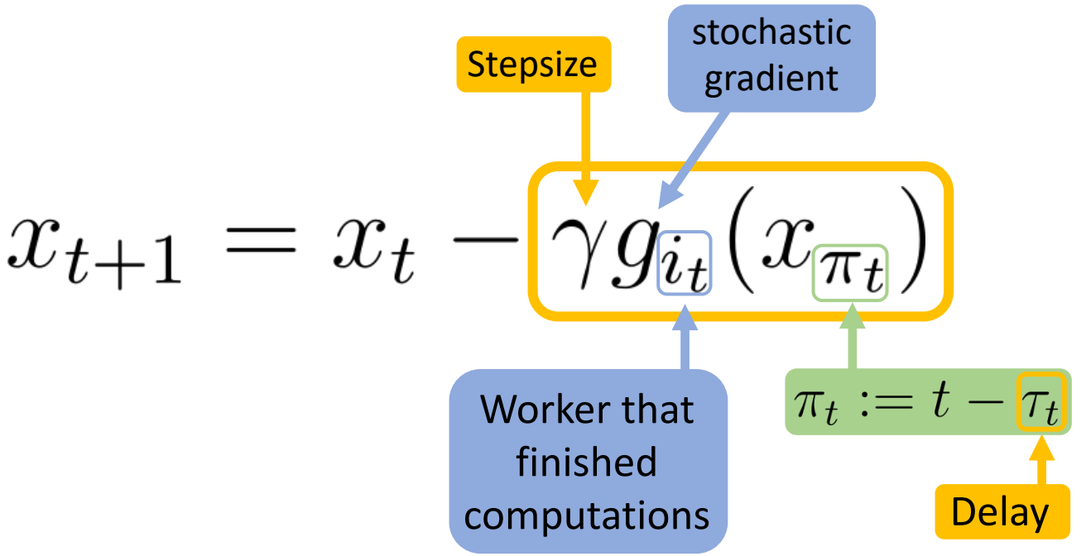
We save time by **asychronizing** the communication, but it comes with a cost of applying **delayed gradients**

Computation step

Outline

1. The Problem Formulation
2. Theoretical Analysis
3. Numerical Experiments

General Asynchronous Algorithm



1. The characteristics of a particular asynchronous method depend on how sequences $\{i_t\}$ and $\{\pi_t\}$ are defined.

2. In order to analyze a particular asynchronous method, we only need to understand how sequences $\{i_t\}$ and $\{\pi_t\}$ affect the method.

3. Knowing 2, we can design more efficient asynchronous SGD-type algorithms.

Having the above stated, we propose a **unified framework, called AsGrad,** to analyze asynchronous SGD-type algorithms.

Theoretical Analysis: Main Result

The analysis is performed utilizing the **restarting sequence**

$$\widetilde{x}_0 = x_0, \quad \widetilde{x}_{t+1} = \begin{cases} \widetilde{x}_t - \gamma \nabla f(x_t) & \text{if } t+1 \neq \tau k \text{ for any } k \geq 1, \\ x_{t+1} & \text{if } t+1 = \tau k \text{ for some } k \geq 1. \end{cases}$$

Choice of τ : $\Theta(1/\gamma)$

Under standard assumptions we can guarantee to find a stationary point such that

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla f(x_t)\|^2\right] \leq \mathcal{O}\left(\frac{1}{\gamma T} + \gamma\sigma^2 + \gamma^2\left[\frac{1}{\lfloor T/\tau \rfloor}\sum_{k=0}^{\lfloor T/\tau \rfloor}\sigma_{k,\tau}^2 + \frac{1}{T}\nu^2\right]\right)$$

Noise variance

Delay variance of $\{i_t\}$

Hides numerical constants

Sequence correlation of $\{i_t\}$

Stepsize restrictions:
$20L\gamma\sqrt{\tau_{\max}\tau_C} \leq 1$
$6L\gamma \leq 1$

1. The closer the sequence $\{i_t\}$ to uniform sampling, the better convergence is.
2. The closer the sequence $\{\pi_t\}$ to $\{t\}$, the better convergence is.

Theoretical Analysis: Special Cases

1. **Pure Asynchronous SGD** (new job is assigned to the *same worker* who finished computations)

$$\mathcal{O}\left(\frac{\tau_C}{T} + \left(\frac{\sigma^2}{T}\right)^{1/2} + \left(\frac{\tau_C G}{T}\right)^{2/3} + \zeta^2\right)$$

τ_C maximum number of active workers

ζ^2 gradient similarity

G gradient bound

2. **Random Asynchronous SGD** (new job is assigned to randomly selected worker)

$$\mathcal{O}\left(\frac{\tau_C}{T} + \left(\frac{\sigma^2}{T}\right)^{1/2} + \left(\frac{\zeta^2}{T}\right)^{1/2} + \left(\frac{\tau_C G}{T}\right)^{2/3}\right)$$

3. **Shuffle Asynchronous SGD [ours]** (new job is assigned according to reshuffled order of workers)

$$\mathcal{O}\left(\frac{n}{T} + \left(\frac{\sigma^2}{T}\right)^{1/2} + \left(\frac{\sqrt{n}\zeta}{T}\right)^{2/3} + \left(\frac{\tau_C G}{T}\right)^{2/3}\right)$$

Our results improve over the results of previous works or match them in terms of the dependencies of the rate on maximum delay.

Outline

1. The Problem Formulation
2. Theoretical Analysis
3. Numerical Experiments

Experiments

We consider non-convex Logistic Regression

$$f_i(x) = \frac{1}{m}\sum_{j=1}^{m} \log\left(1 + e^{-b_{ij}a_{ij}^\top x}\right) + \lambda r(x), \quad r(x) = \sum_{j=1}^{d} \frac{x_j^2}{1+x_j^2}$$

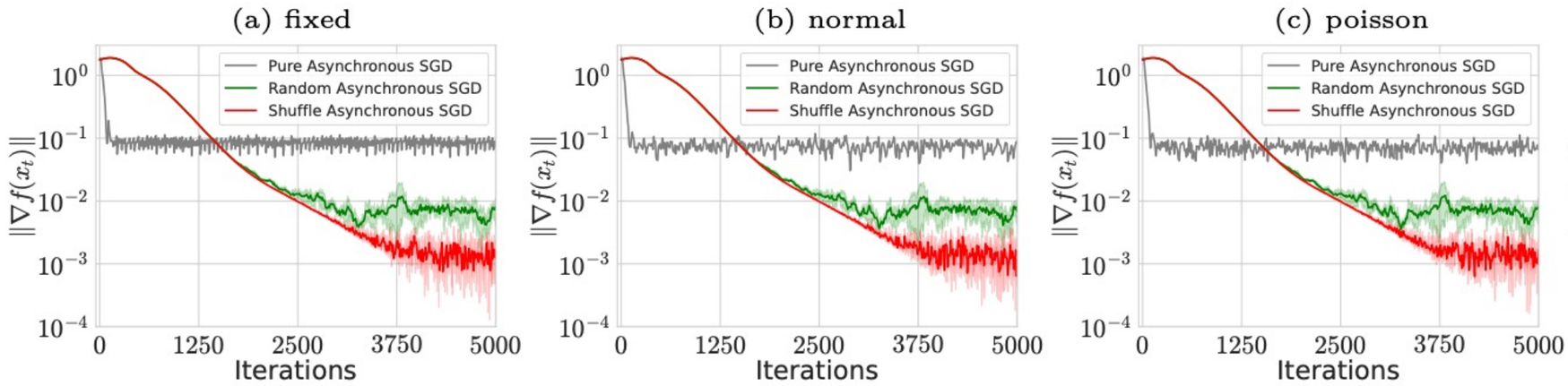
Training data points

Regularization parameter

Gradient computation time simulations:

- **Fixed:** worker *i* spends s_i seconds to compute one gradient;
- **Poisson:** worker *i spends* s_i *seconds to compute one gradient where* $s \sim \mathrm{Po}(s_i)$*;*
- **Normal:** worker *i* spends $|s| + 1$ seconds to compute one gradient where $s \sim \mathcal{N}(s_i, s_i)$.

Experiments



We observe that

- Random and Shuffled Asynchronous SGD is always faster than Pure Asynchronous SGD, and both of them find a stationary point with smaller error;
- Random and Shuffled Asynchronous SGD converge with similar speed, but Shuffled Asynchronous SGD finds a stationary point with smaller gradient norm;

Thank you

Rustem Islamov, Mher Safaryan, Dan Alistarh, AsGrad: A Sharp Unified Analysis of Asynchronous-SGD algorithms, International Conference on Artificial Intelligence and Statistics (AISTATS 2024)