Ringmaster ASGD: The First Asynchronous SGD with Optimal Time Complexity

Artavazd Maranjyan

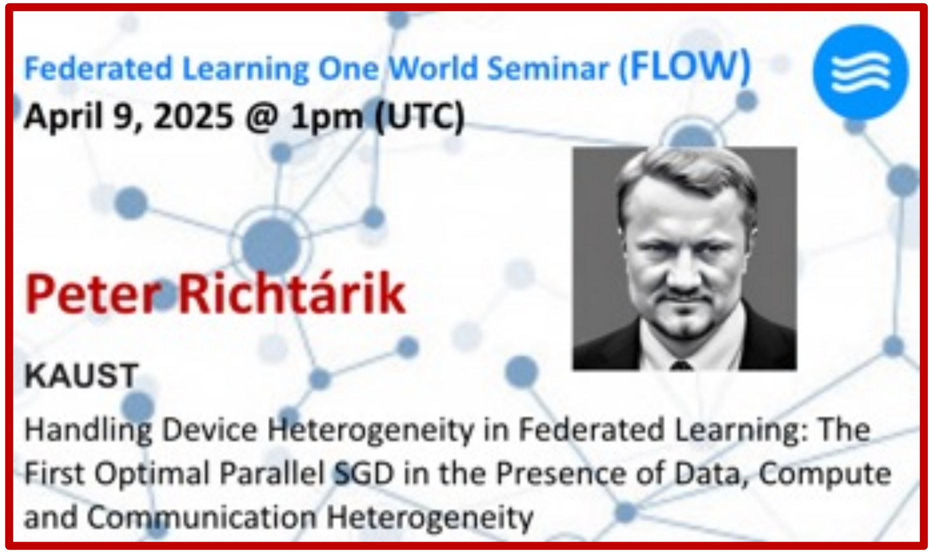
FLOW Talk #126
16 April 2025

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

Ringmaster ASGD: The First Asynchronous SGD with Optimal Time Complexity

- Problem setup
  - Optimization objective
  - Heterogenous system
  - Method (SGD)
- Different ways of parallelizing SGD
  - Synchronized approaches
  - Asynchronous SGD
- Problems of ASGD
- Ringmaster ASGD

Series of talks on Asynchronous SGD



Part 1

The core optimization problem in Machine Learning (and beyond)

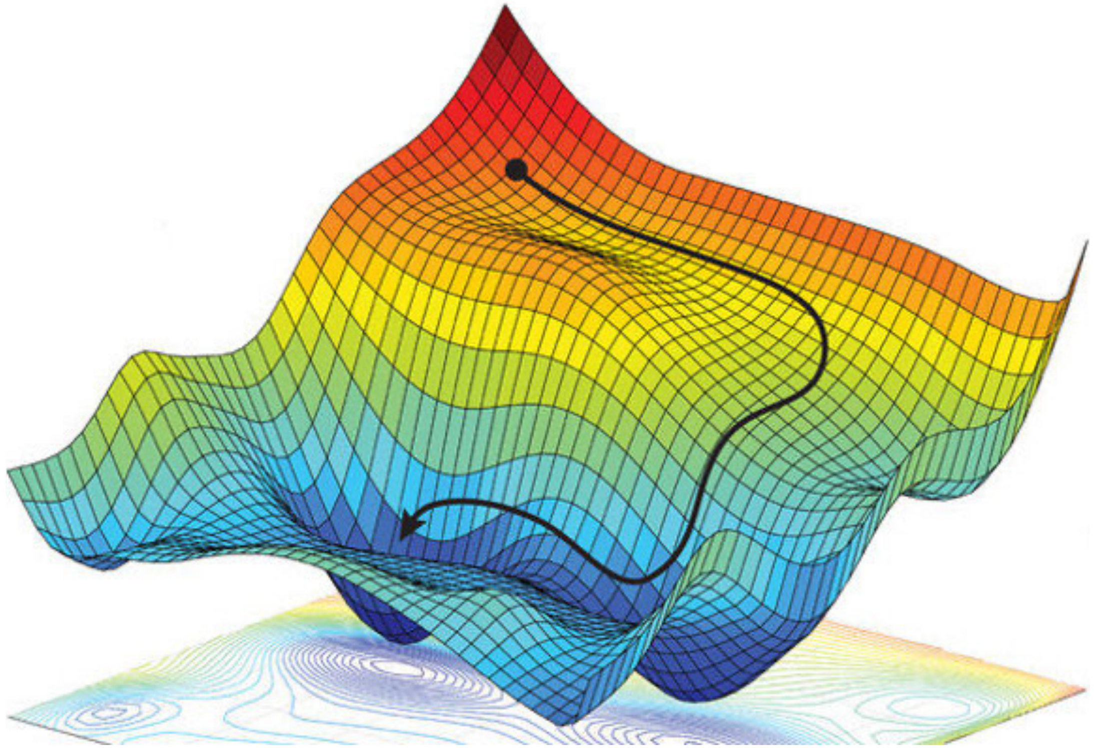
$$\min_{x \in \mathbb{R}^d} \left\{ f(x) := \mathbb{E}_{\xi \sim \mathcal{D}} \left[f(x; \xi) \right] \right\}$$

Loss of a data sample ξ

The distribution of the training dataset

$$\mathcal{D} = \mathrm{Uniform}([m]) \qquad \frac{1}{m} \sum_{i=1}^{m} f(x; \xi_i)$$

A common method in ML is Stochastic Gradient Descent (SGD)



Stepsize / Learning rate

$$x^{k+1} = x^k - \gamma g(x^k)$$

Unbiased gradient estimator, e.g.,

$$\nabla f(x^k; \xi^k)$$

$$\frac{1}{B}\sum_{i=1}^{B} \nabla f(x^k; \xi_i^k)$$

How to parallelize SGD in heterogeneous systems?

$\nabla f(x;\xi)$

Compute time = τ_1

τ_1

$\nabla f(x;\xi)$

Compute time = τ_2

τ_2

$\nabla f(x;\xi)$

Compute time = τ_3

τ_3

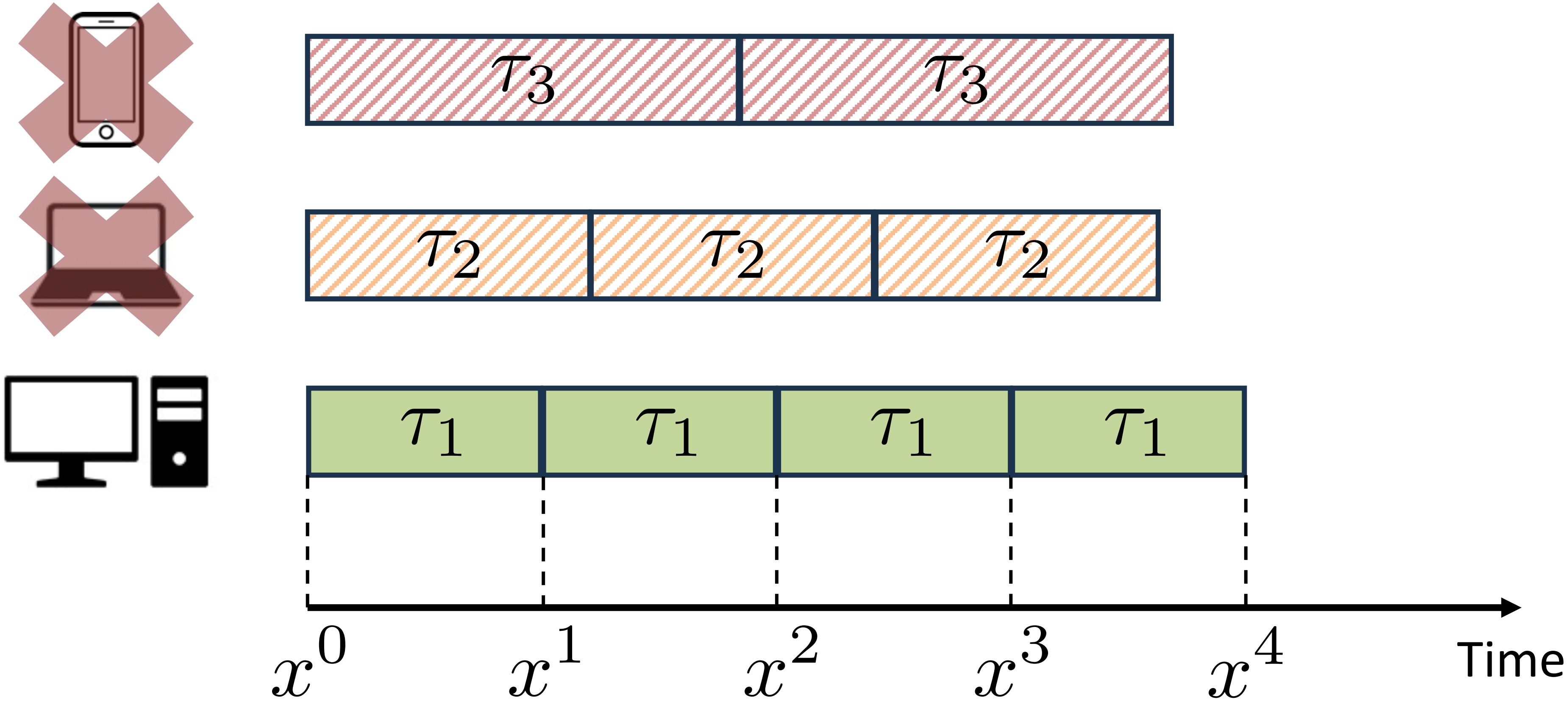
$\mathbb{E}[g(x^k)] = \nabla f(x^k)$

Server

$$x^{k+1} = x^k - \gamma g(x^k)$$

How to construct?

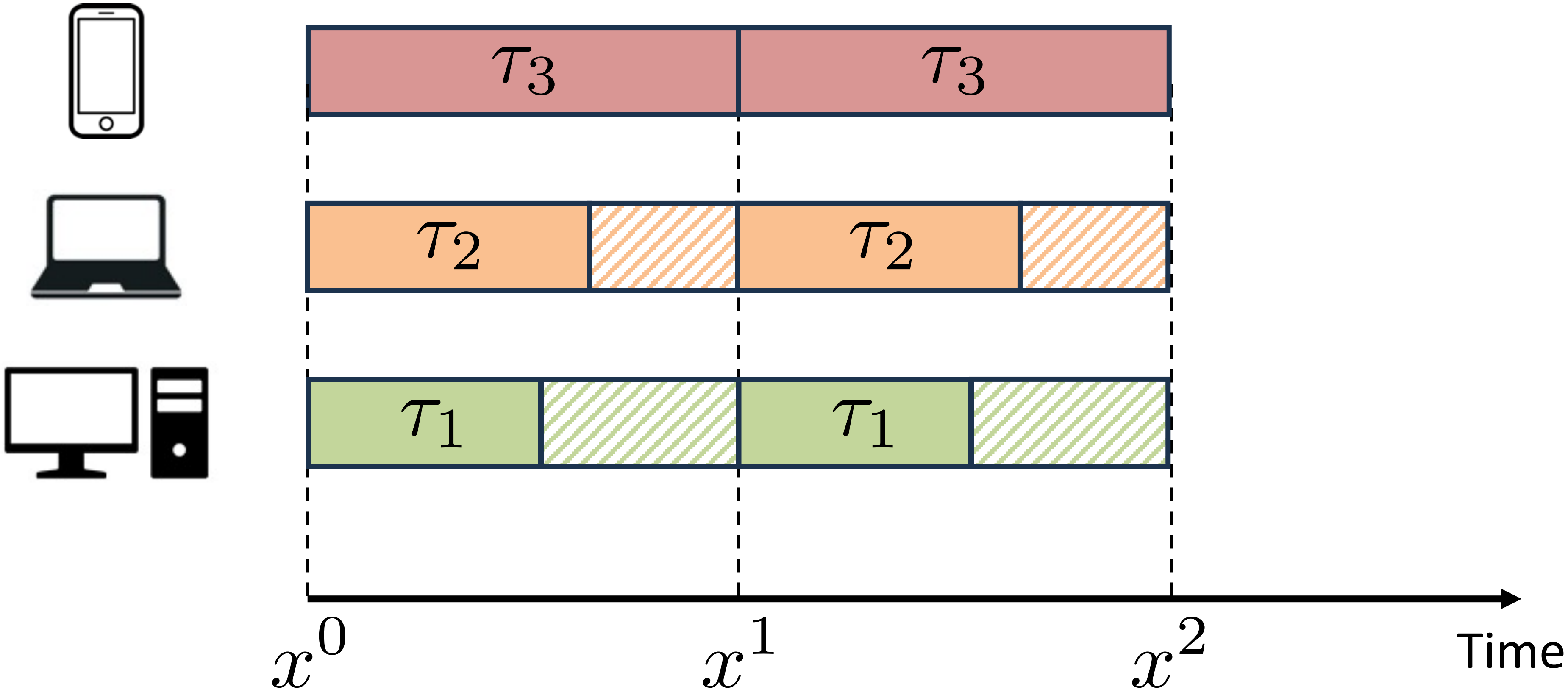
Hero SGD:
The fastest worker does it all



$$x^{k+1} = x^k - \gamma \nabla f(x^k; \xi^k)$$

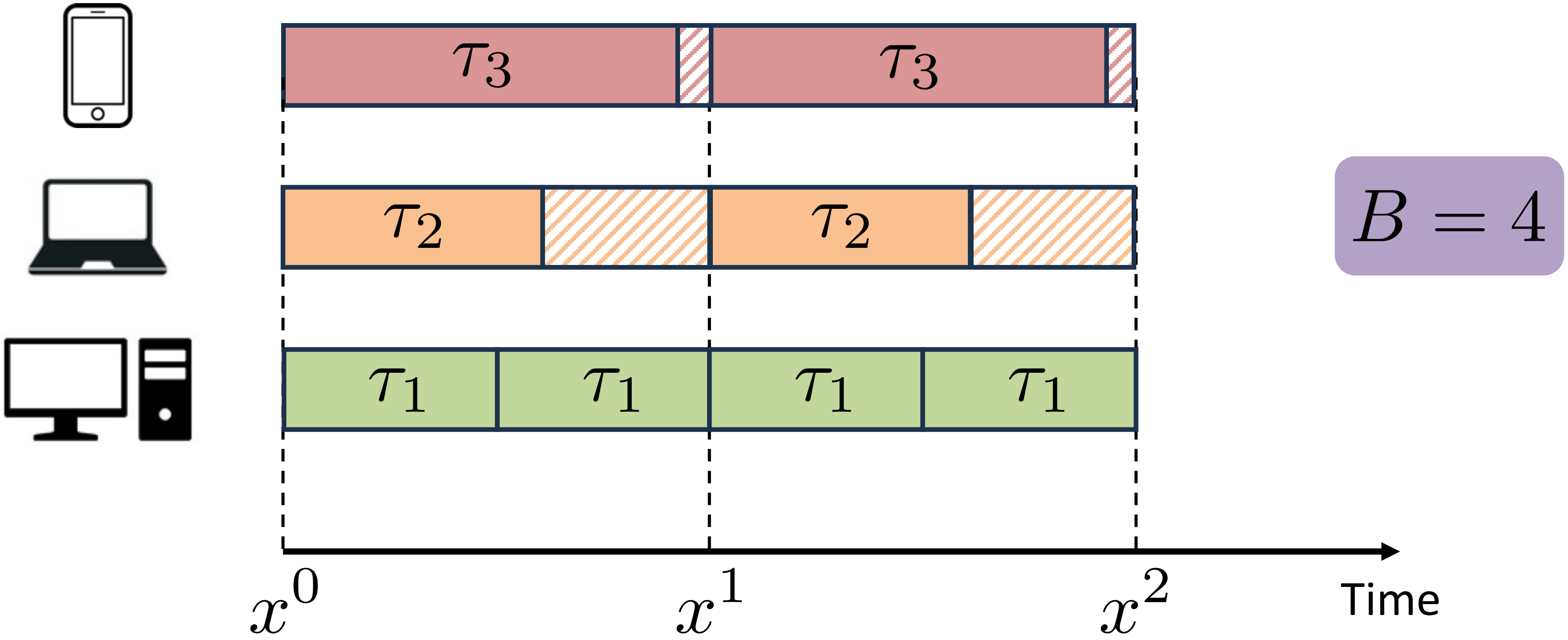
Minibatch SGD:
Each worker does one job only

$n = 3$



$$x^{k+1} = x^k - \gamma \boxed{\frac{1}{n} \sum_{i=1}^{n} \nabla f(x^k; \xi_i^k)}$$

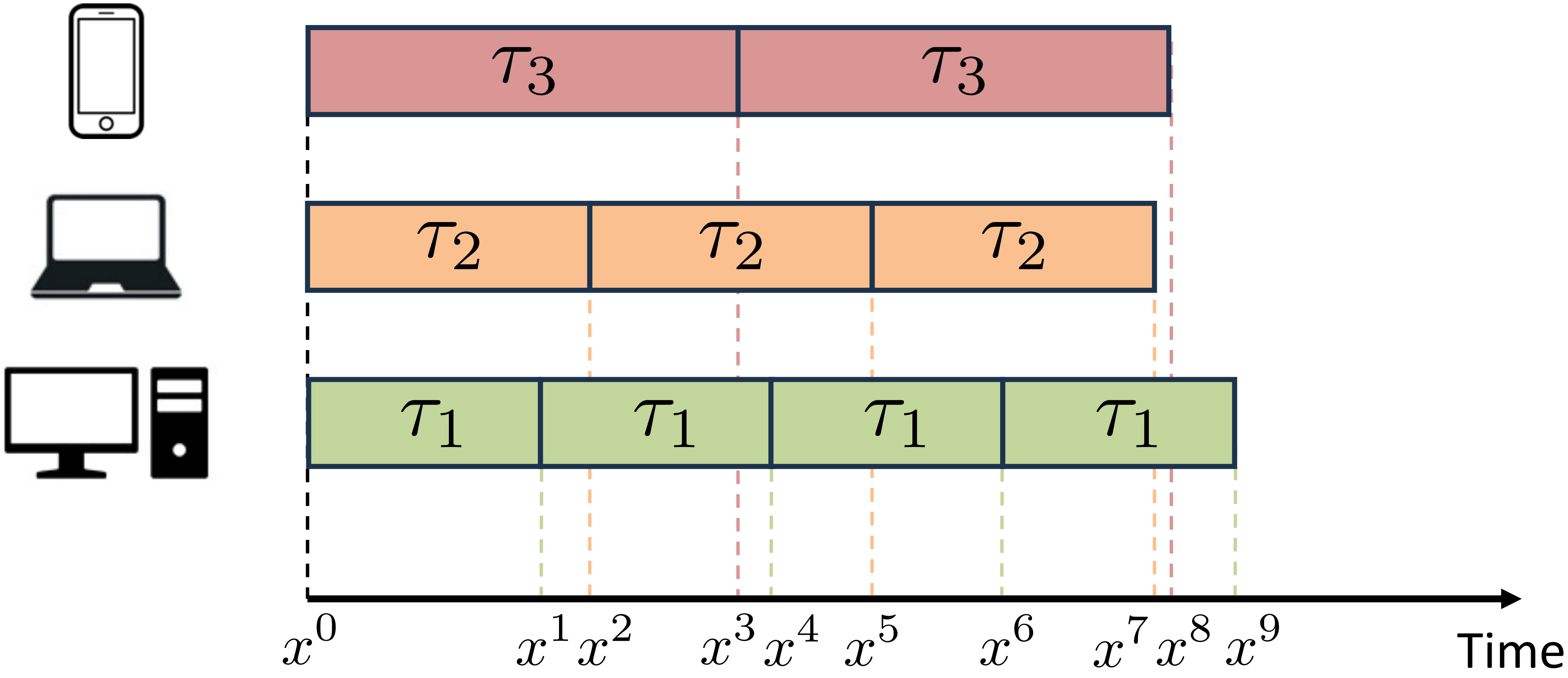
Rennala SGD:
Asynchronous batch collection



$$x^{k+1} = x^k - \gamma \frac{1}{B} \sum_{j=1}^{B} \nabla f(x^k; \xi_j^k)$$

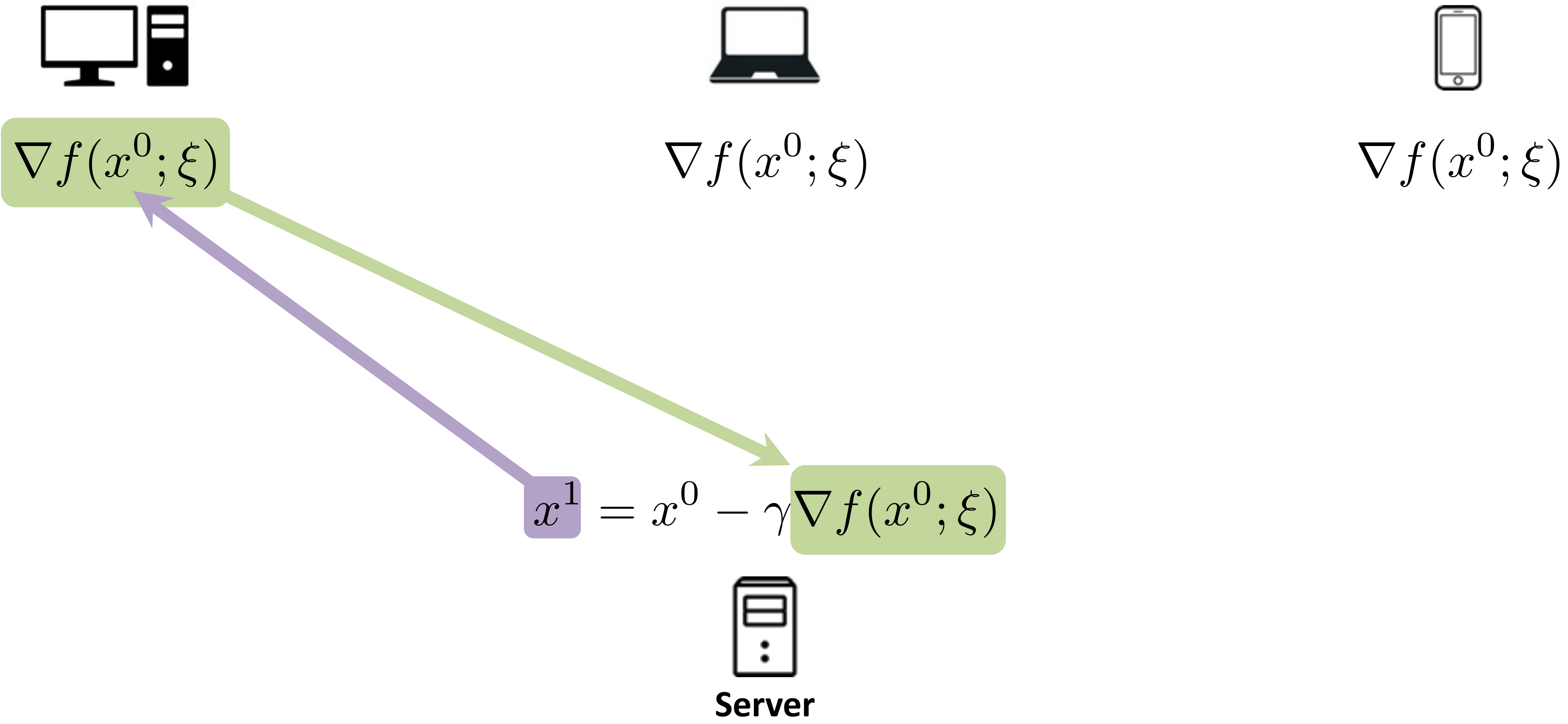
Tyurin and Richtárik (2024)

Asynchronous SGD
Remove the synchronization

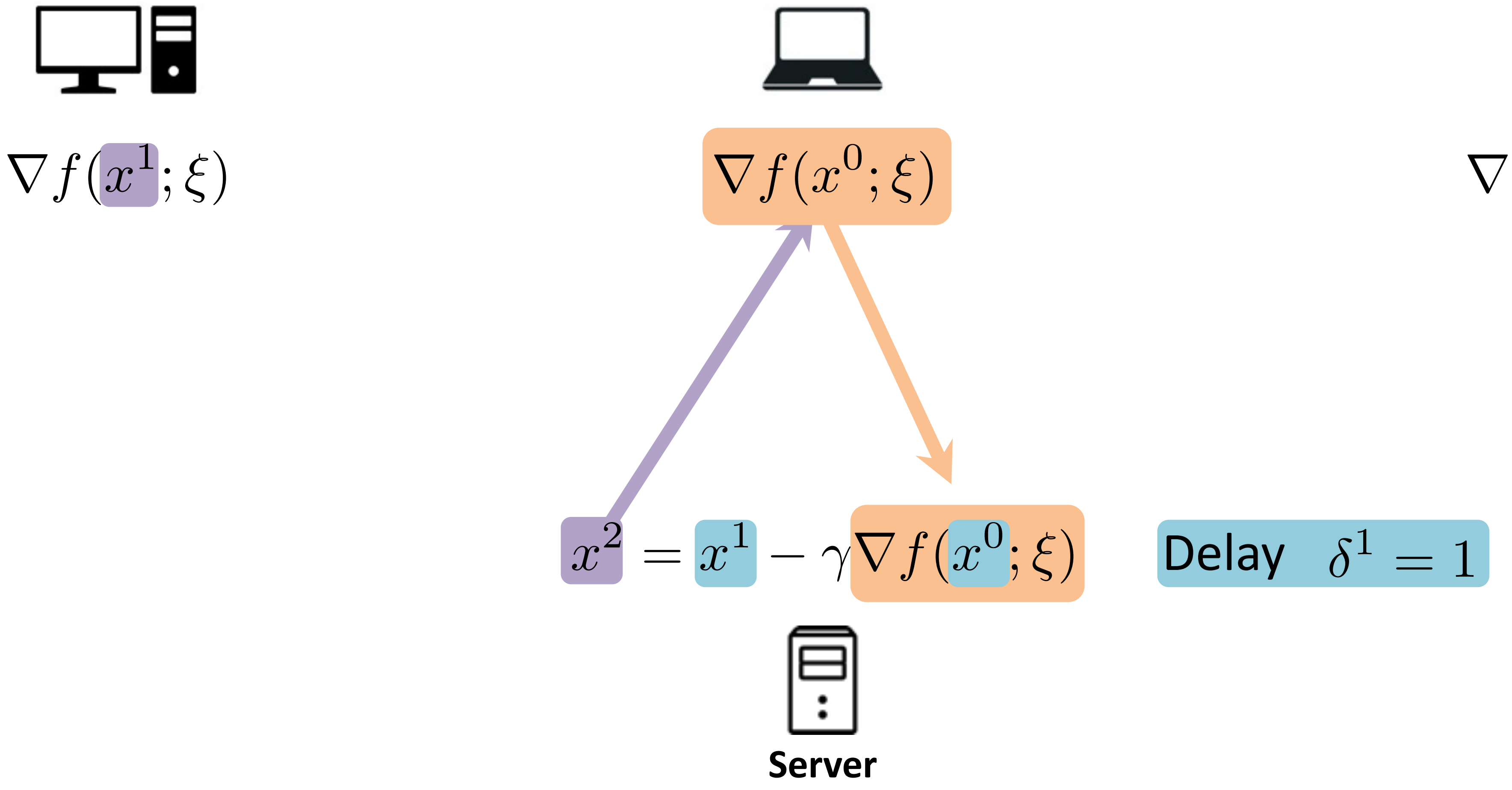


$$x^{k+1} = x^k - \gamma \cancel{g(x^k)}$$

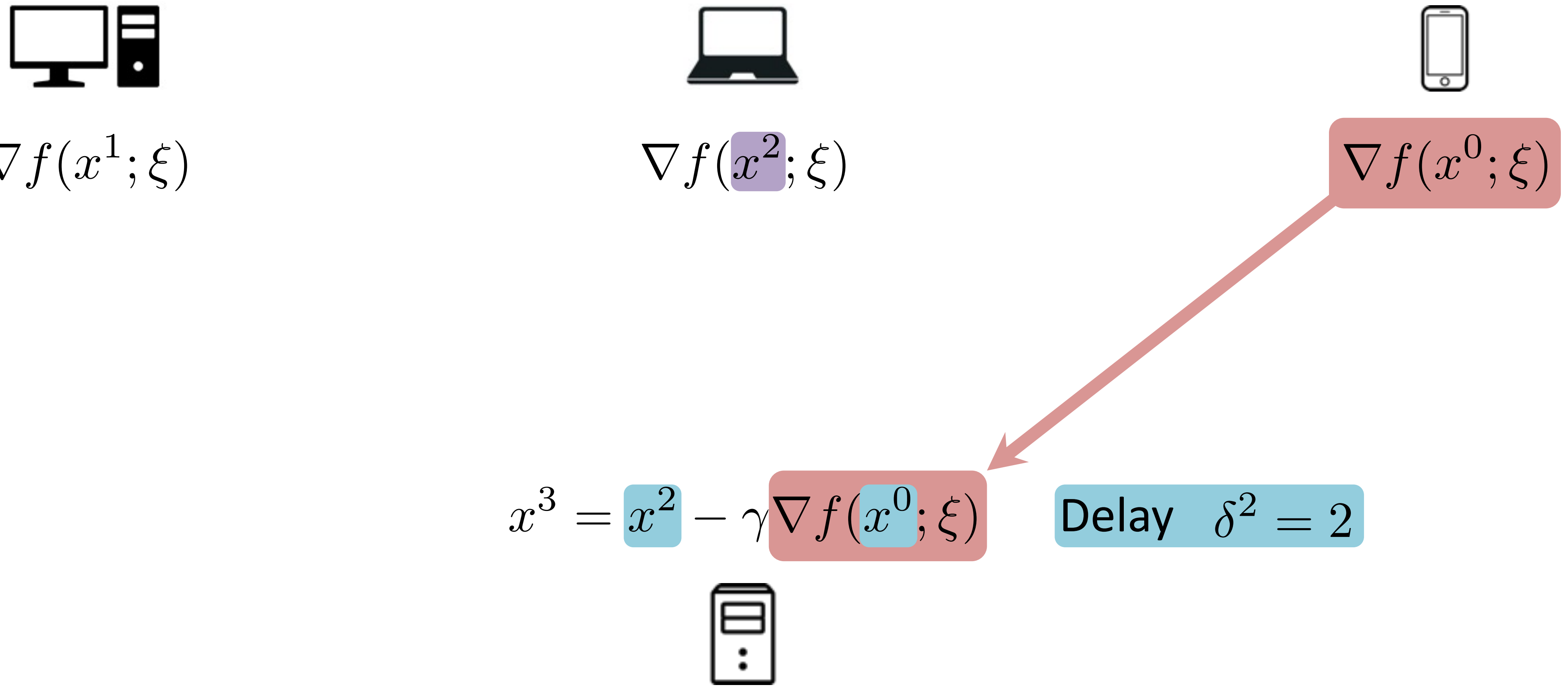
Updates of Asynchronous SGD has delayed stochastic gradients



Updates of Asynchronous SGD has delayed stochastic gradients



Updates of Asynchronous SGD has delayed stochastic gradients



$\nabla f(x^1;\xi)$ $\nabla f(x^2;\xi)$ $\nabla f(x^0;\xi)$

$$x^3 = x^2 - \gamma \nabla f(x^0;\xi)$$

Delay $\delta^2 = 2$

Server

Updates of Asynchronous SGD has delayed stochastic gradients

$$x^{k+1} = x^k - \gamma \nabla f(x^{k-\delta^k}; \xi)$$

Delay δ^k

Server

Feng Niu, Benjamin Recht, Christopher Re, Stephen J. Wright.
HOGWILD!: A lock-free approach to parallelizing stochastic gradient descent.
NeurIPS 2011

NeurIPS 2020 Test of Time Award

More than a decade of research on Asynchronous SGD

"First" ASGD

2011

Delay-adaptive ASGD

2022

Koloskova et al.
Mishchenko et al.

Semi-asynchronous SGD

2023

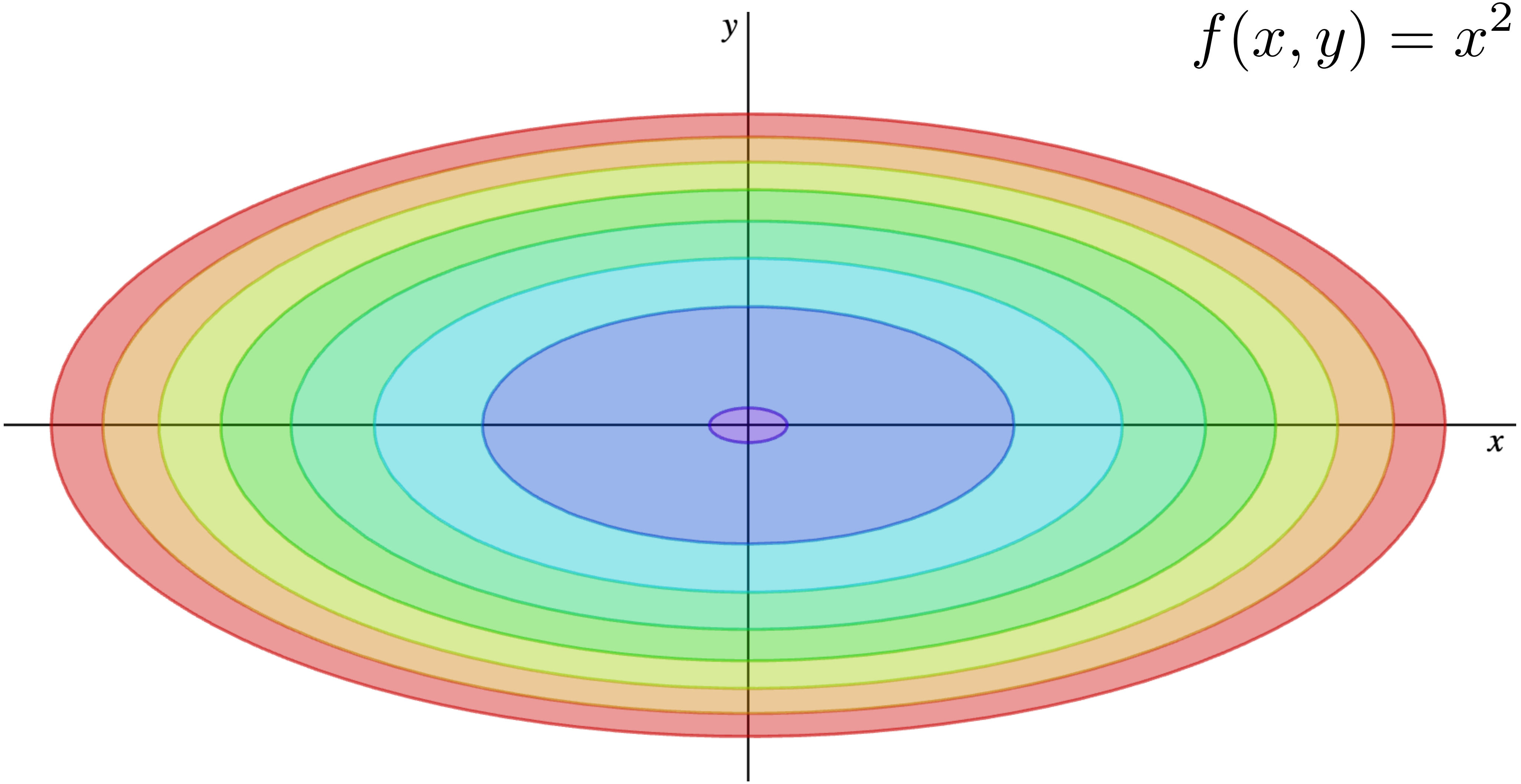
Rennala SGD

fully asynchronous SGD

2025

Asynchronous SGD can get wild: delays can degrade performance

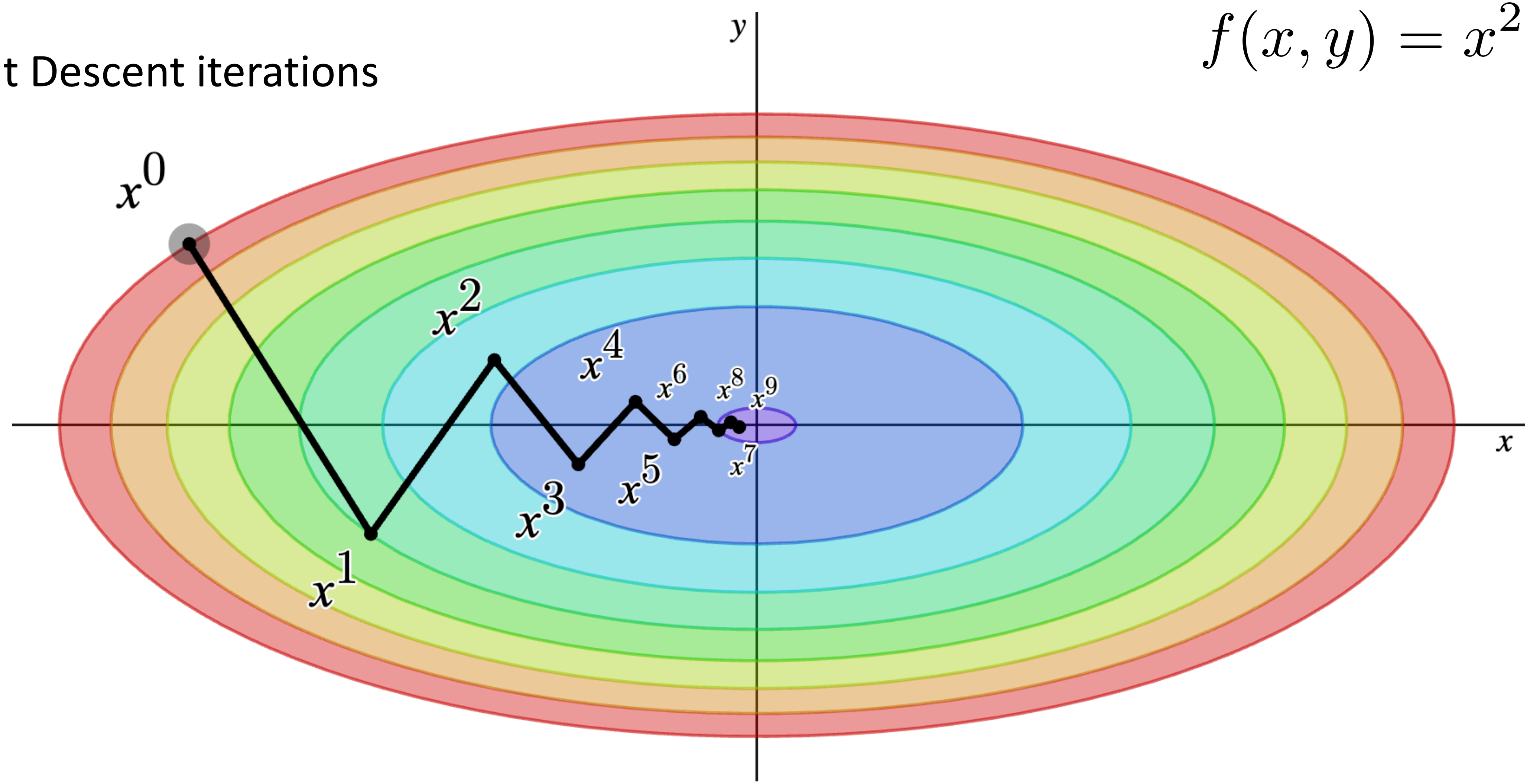
$$f(x, y) = x^2 + 5y^2$$



Asynchronous SGD can get wild: delays can degrade performance

Gradient Descent iterations

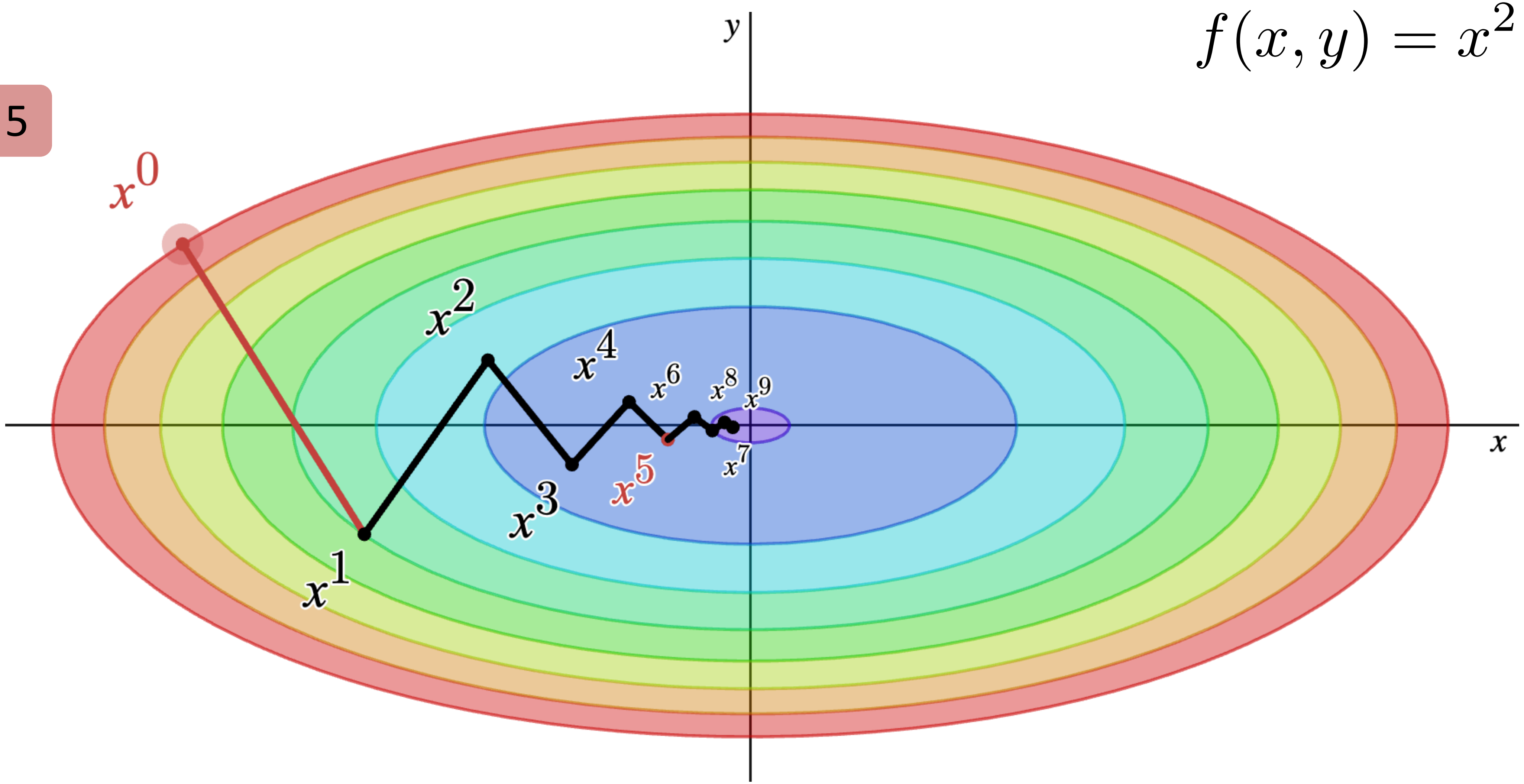
$$f(x, y) = x^2 + 5y^2$$



Asynchronous SGD can get wild: delays can degrade performance

$$f(x, y) = x^2 + 5y^2$$

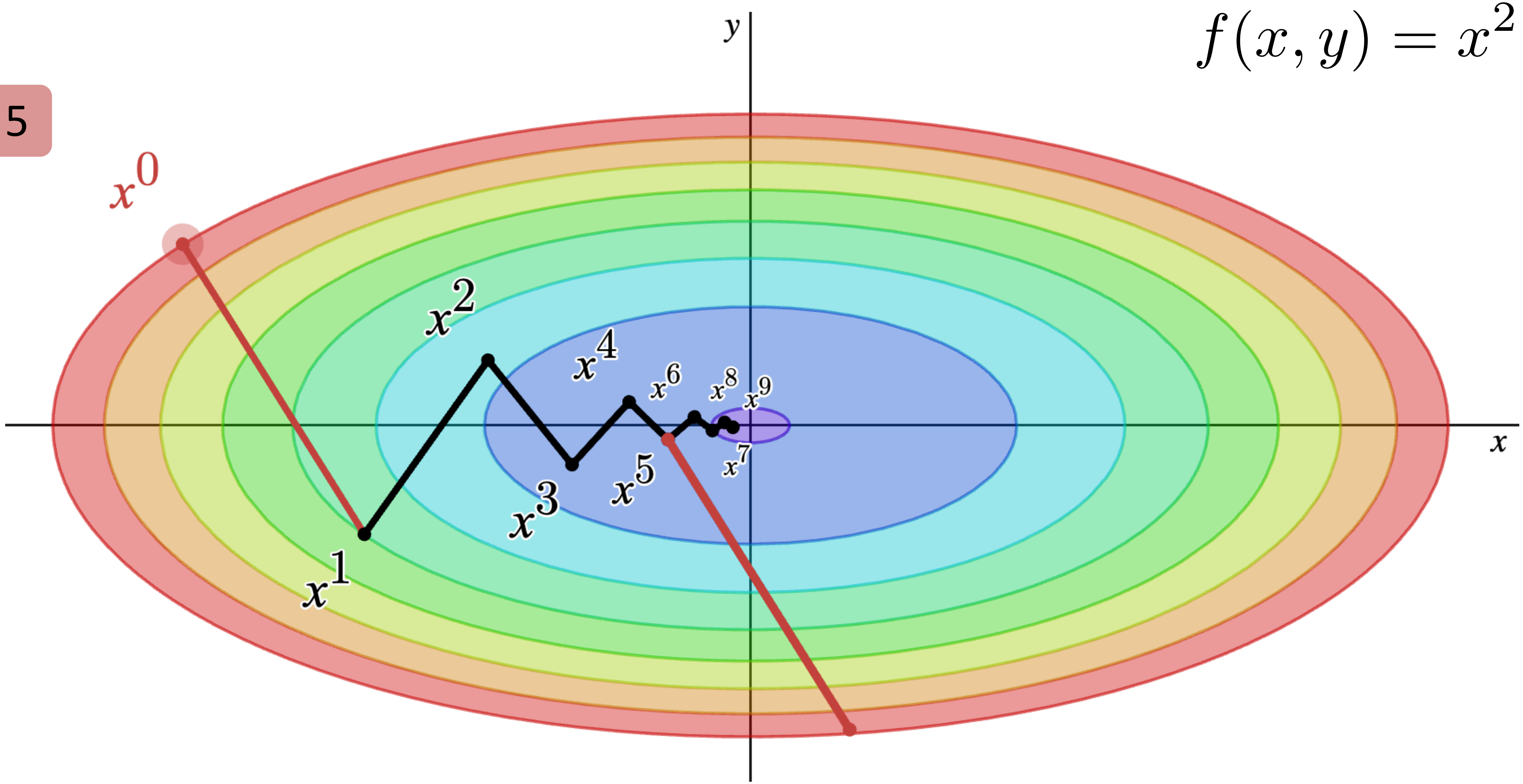
Delay = 5



Asynchronous SGD can get wild: delays can degrade performance

Delay = 5

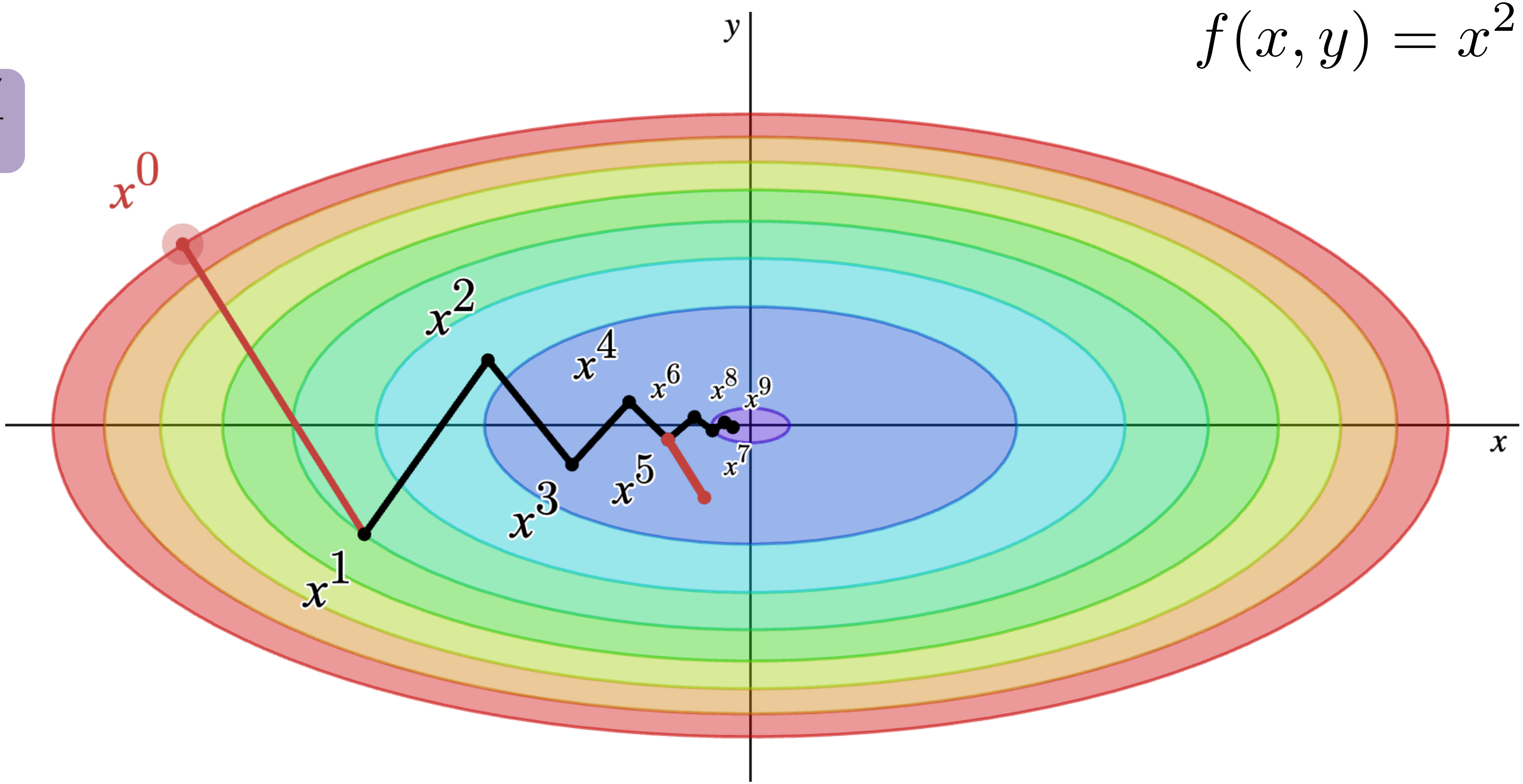
$$f(x, y) = x^2 + 5y^2$$



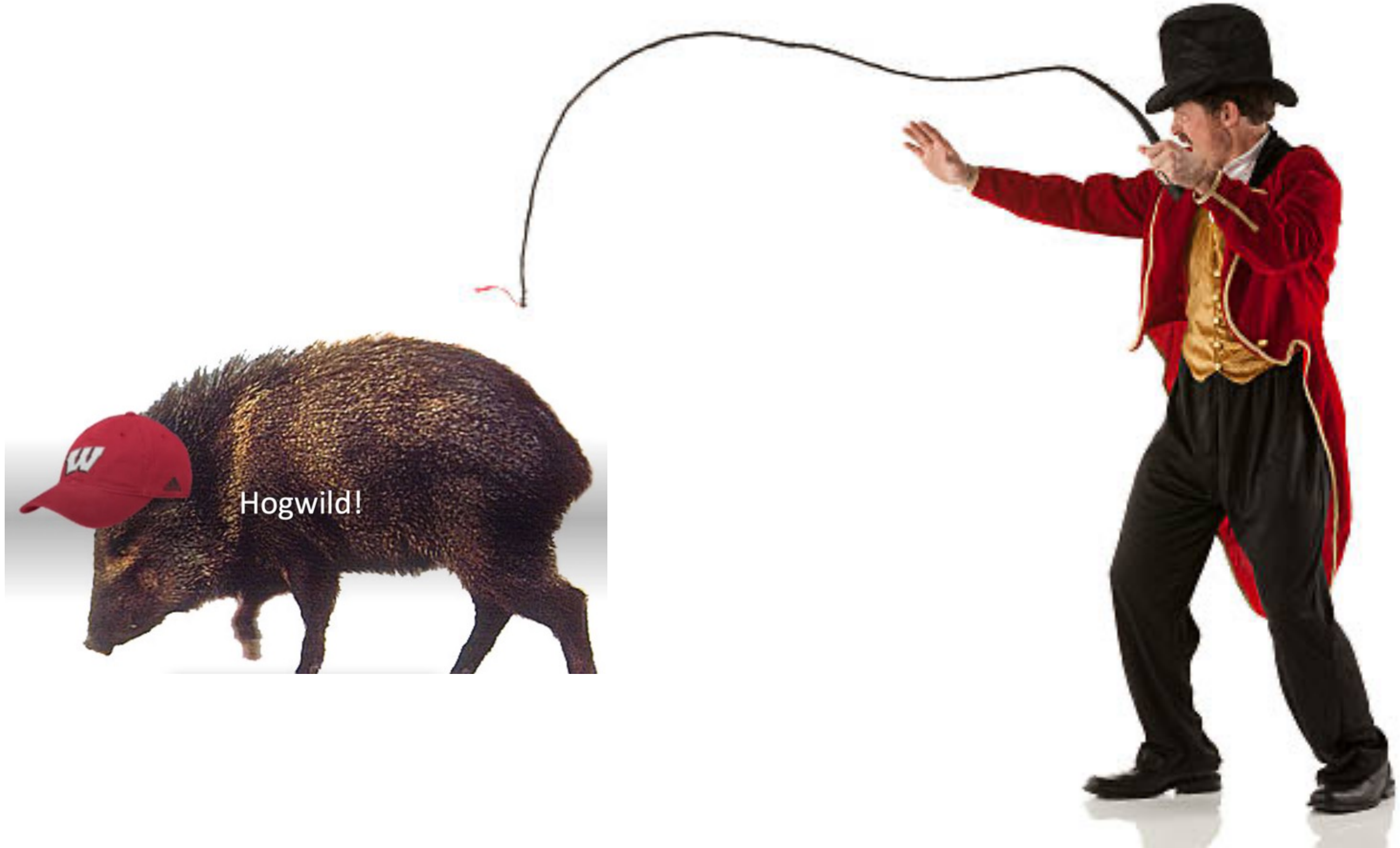
How to fix this?
Make the stepsize smaller

$$\gamma = \frac{\gamma}{5}$$

$$f(x, y) = x^2 + 5y^2$$



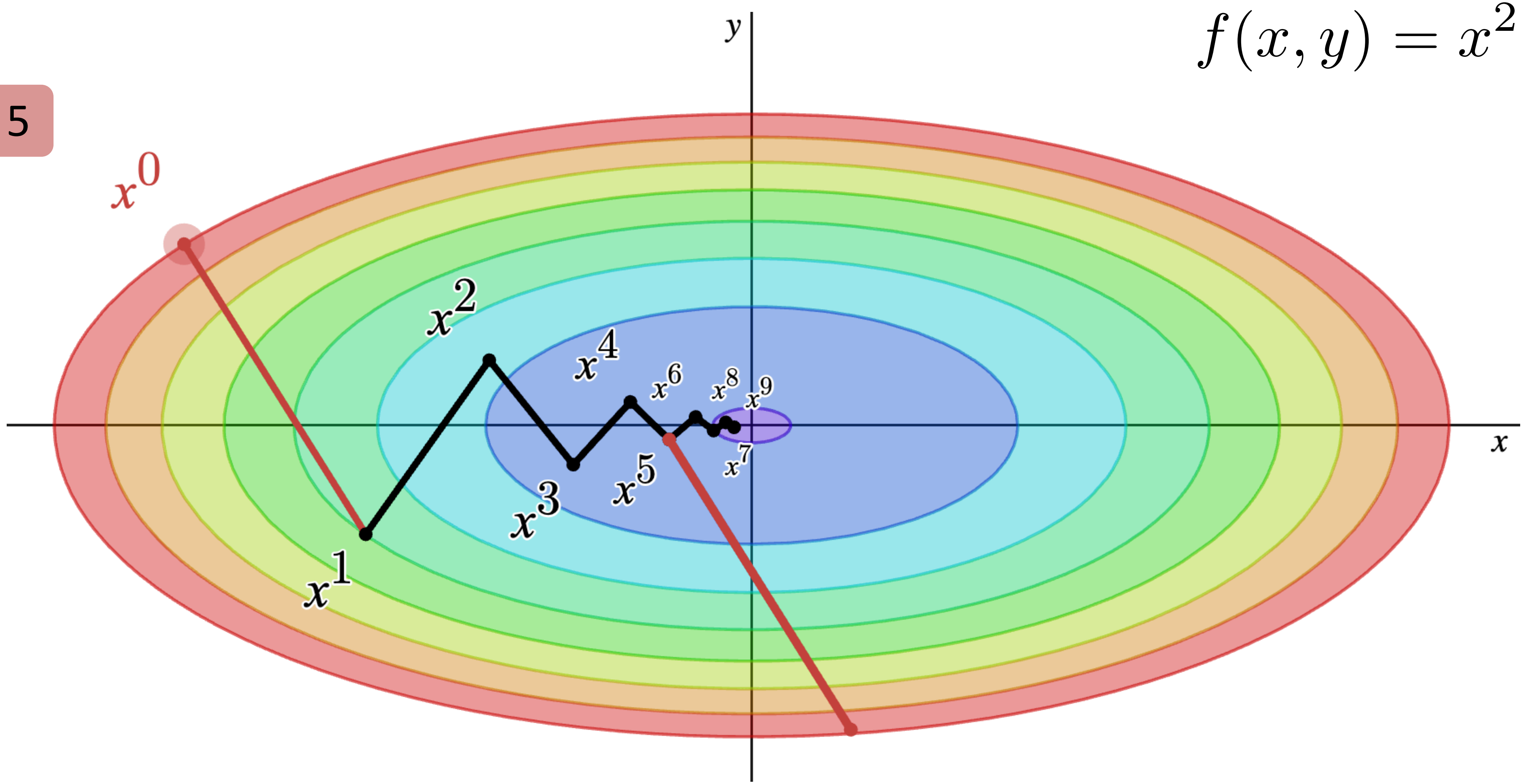
Asynchronous SGD is too wild: Ringmaster ASGD *tames* it



The smaller the delay, the better the gradient

$$f(x, y) = x^2 + 5y^2$$

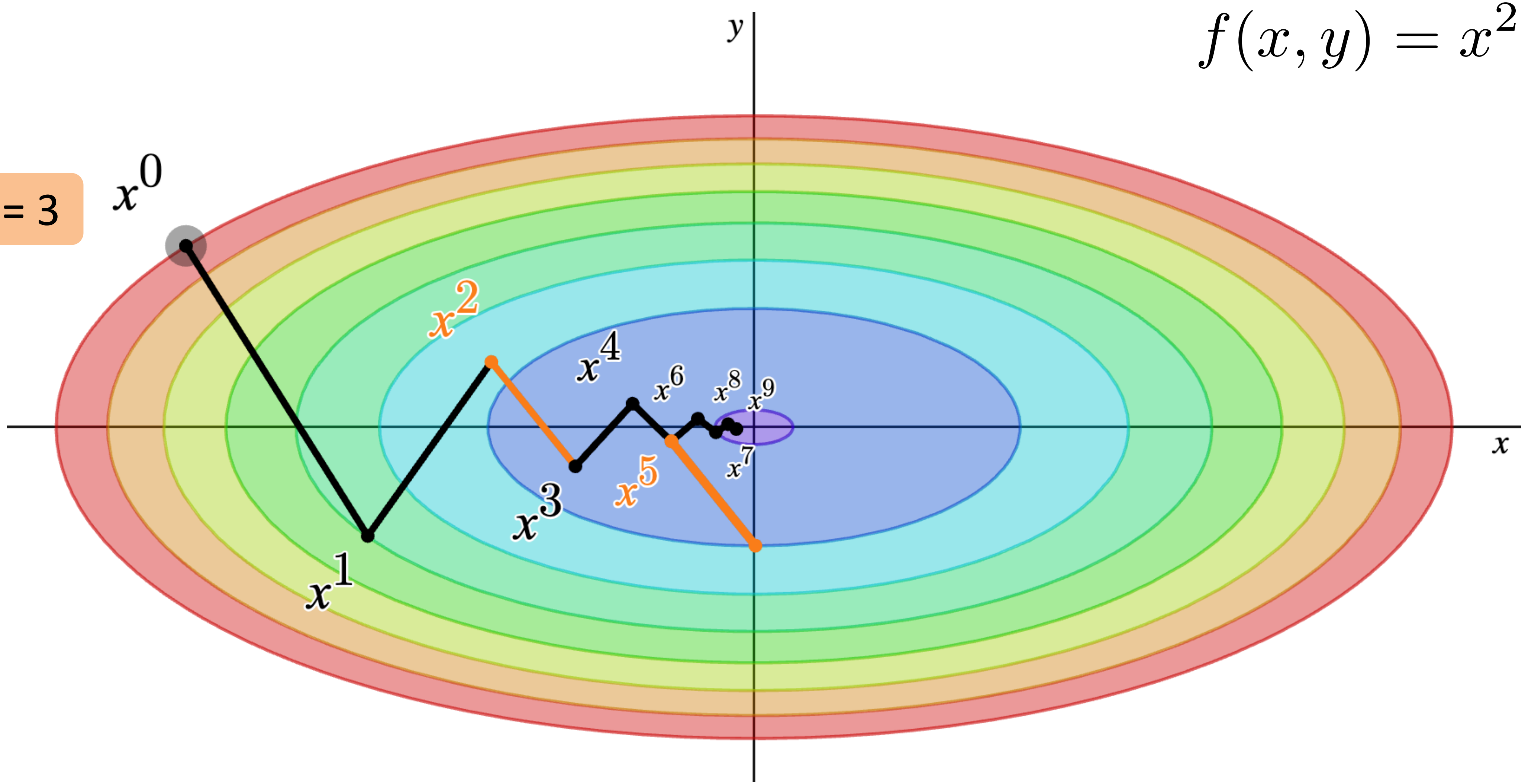
Delay = 5



The smaller the delay, the better the gradient

$$f(x, y) = x^2 + 5y^2$$

Delay = 3



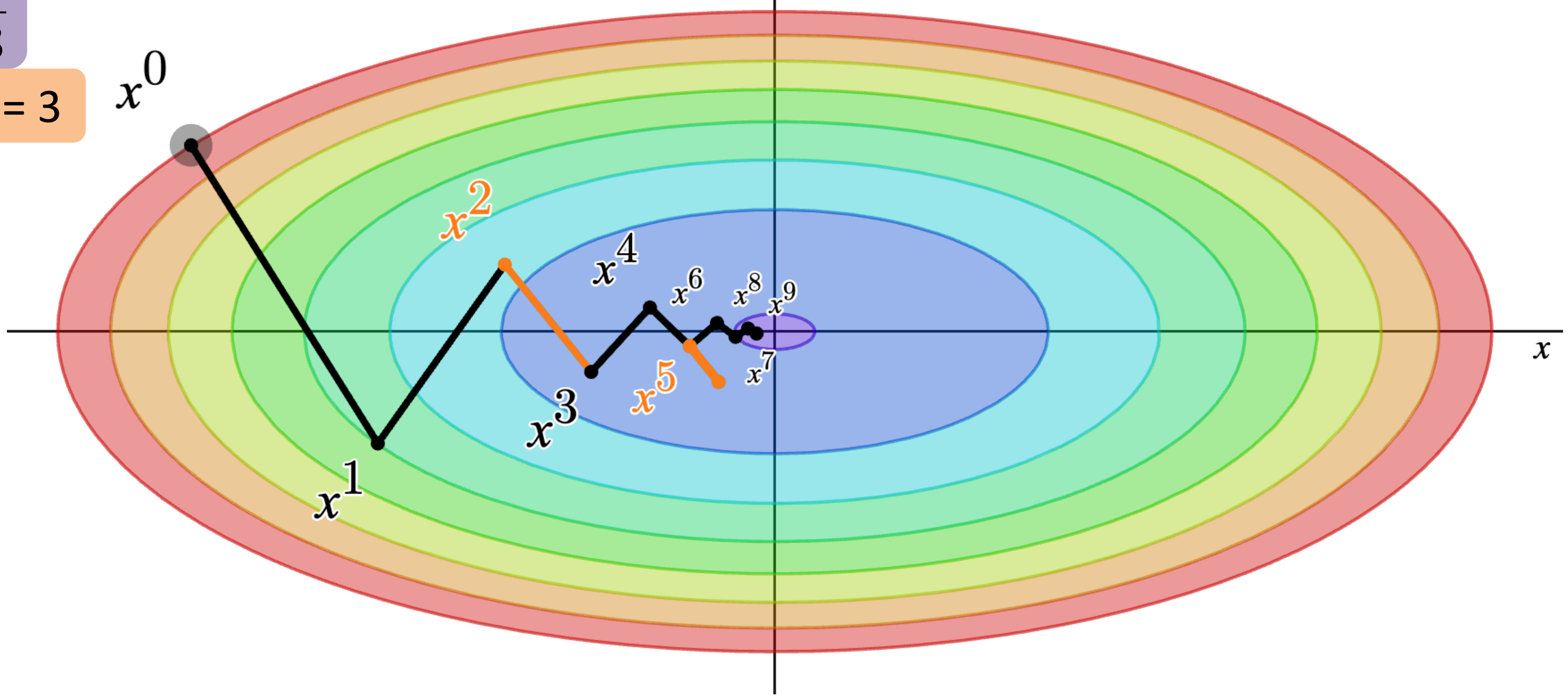
The smaller the delay, the better the gradient

How can we reduce the delay?

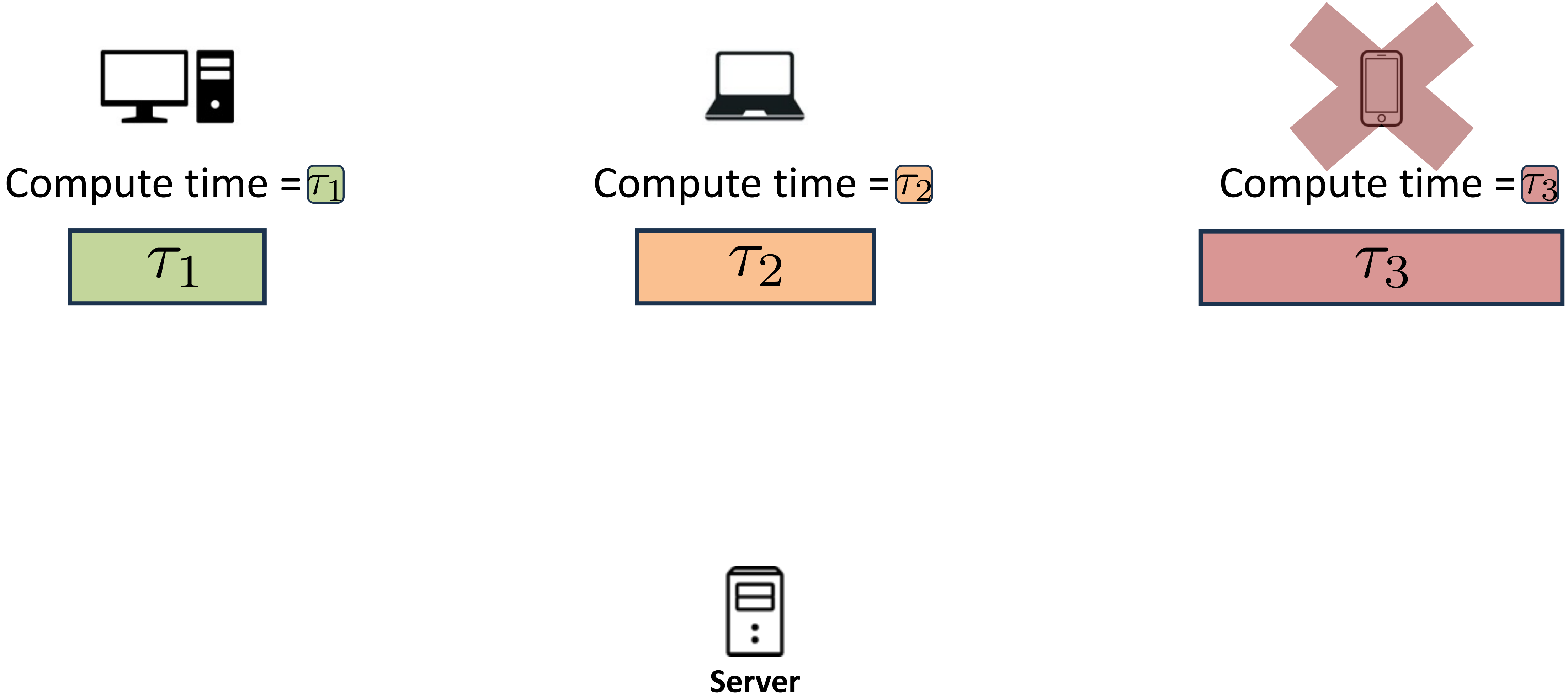
$$f(x, y) = x^2 + 5y^2$$

$\gamma = \frac{\gamma}{3}$

Delay = 3



Naive approach:
Remove slow workers



Naive approach: Remove slow workers

$$\mathbb{E}\left[\|\nabla f(x;\xi) - \nabla f(x)\|^2\right] \leq \sigma^2$$

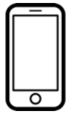
Use only the first $m_\star = \arg\min_{m \in [n]} \left\{ \left(\frac{1}{m} \sum_{i=1}^{m} \frac{1}{\tau_i} \right)^{-1} \left(1 + \frac{\sigma^2}{m\varepsilon} \right) \right\}$ fastest workers

$$\mathbb{E}\left[\|\nabla f(x)\|^2\right] \leq \varepsilon$$

Problem: τ_i-s may be unknown and dynamic

Ringmaster ASGD:
Have a threshold on delays





$$\nabla f\left(x^{k-\delta^k}; \xi_i^{k-\delta^k}\right)$$

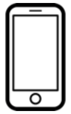
If: $\delta^k < R$

$$x^{k+1} = x^k - \gamma \nabla f\left(x^{k-\delta^k}; \xi_i^{k-\delta^k}\right)$$

Else: Ignore the gradient and send the current point x^k to the worker

Server

Ringmaster ASGD:
Have a threshold on delays



$\nabla f\left(x^k; \xi_i^k\right)$

How to choose the delay threshold R

If: $\delta^k < R$

$$x^{k+1} = x^k - \gamma \nabla f\left(x^{k-\delta^k}; \xi_i^{k-\delta^k}\right)$$

Else: Ignore the gradient and send the current point x^k to the worker

Server

Certain threshold choices in Ringmaster ASGD recover previous methods

$$R = \max\left\{1, \left\lceil \frac{\sigma^2}{\varepsilon} \right\rceil\right\}$$

$R = 1$
Hero SGD

Sweet spot

$R = \infty$
HOGWILD!

Theoretical results validate our intuition

$$R = \max\left\{1, \left\lceil \frac{\sigma^2}{\varepsilon} \right\rceil\right\} \qquad \gamma = \min\left\{\frac{1}{2RL}, \frac{\varepsilon}{4L\sigma^2}\right\}$$

$$\mathcal{O}\left(\frac{R}{\varepsilon} + \frac{\sigma^2}{\varepsilon^2}\right)$$

Number of iterations

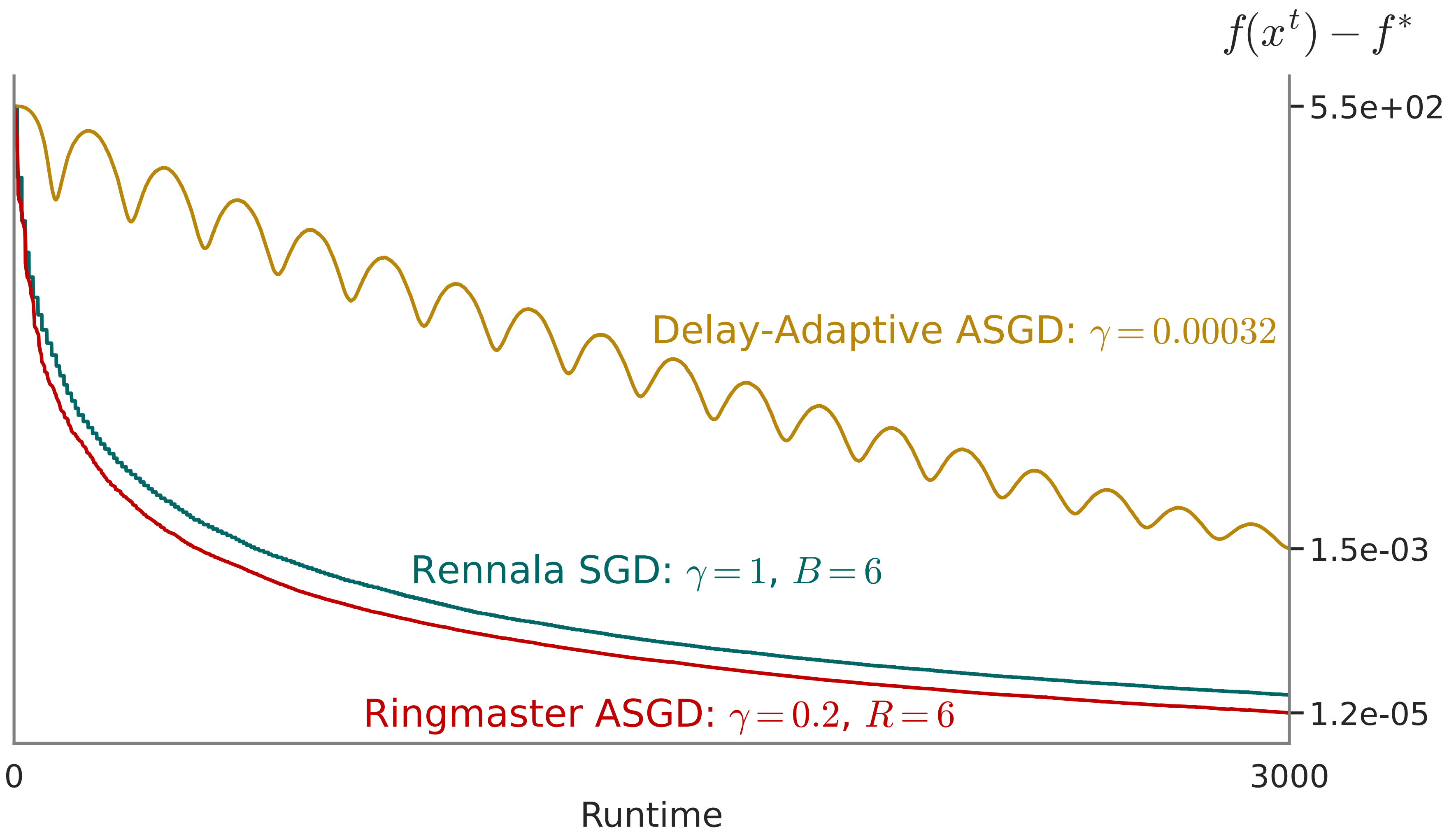
$$\mathcal{O}\left(\min_{m\in[n]}\left[\underbrace{\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}}_{\text{non-decreasing}}\left(\frac{1}{\varepsilon} + \underbrace{\frac{\sigma^2}{m\varepsilon^2}}_{\text{decreasing}}\right)\right]\right)$$

Time complexity

non-decreasing

decreasing

Ringmaster ASGD outperforms existing baselines



Recap of what we have covered

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) := \mathbb{E}_{\xi \sim \mathcal{D}} \left[f(x;\xi) \right] \right\}$$

$\nabla f(x;\xi)$

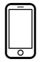
Compute time = τ_1

τ_1

$\nabla f(x;\xi)$

Compute time = τ_2

τ_2



$\nabla f(x;\xi)$

Compute time = τ_3

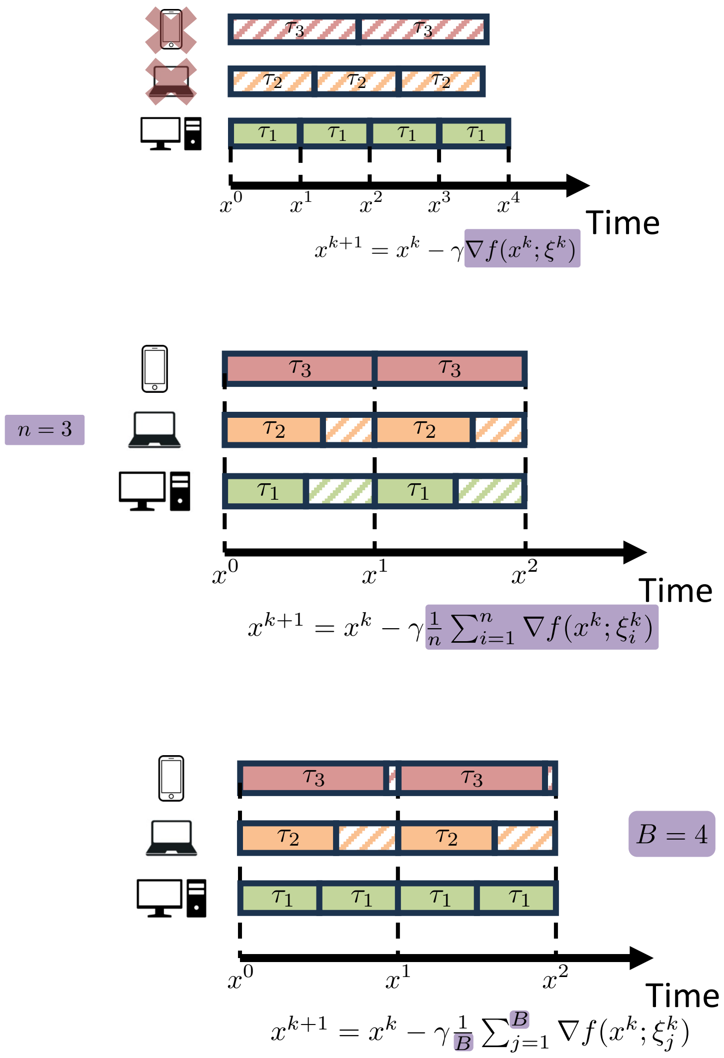
τ_3

$$x^{k+1} = x^k - \gamma g(x^k)$$

Problem setup

- Optimization objective
- Heterogenous system
- Method (SGD)

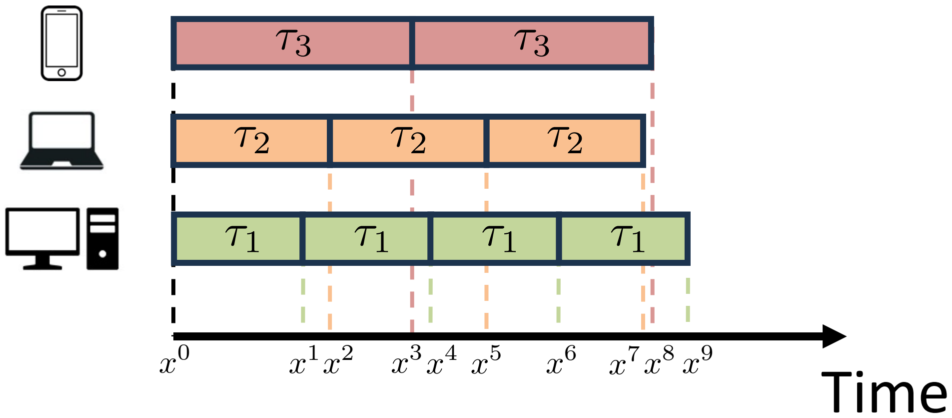
Recap of what we have covered



Problem setup
- Optimization objective
- Heterogenous system
- Method (SGD)

Different ways of parallelizing SGD
- Synchronized approaches

Recap of what we have covered



$$x^{k+1} = x^k - \cancel{\gamma g(x^k)}$$

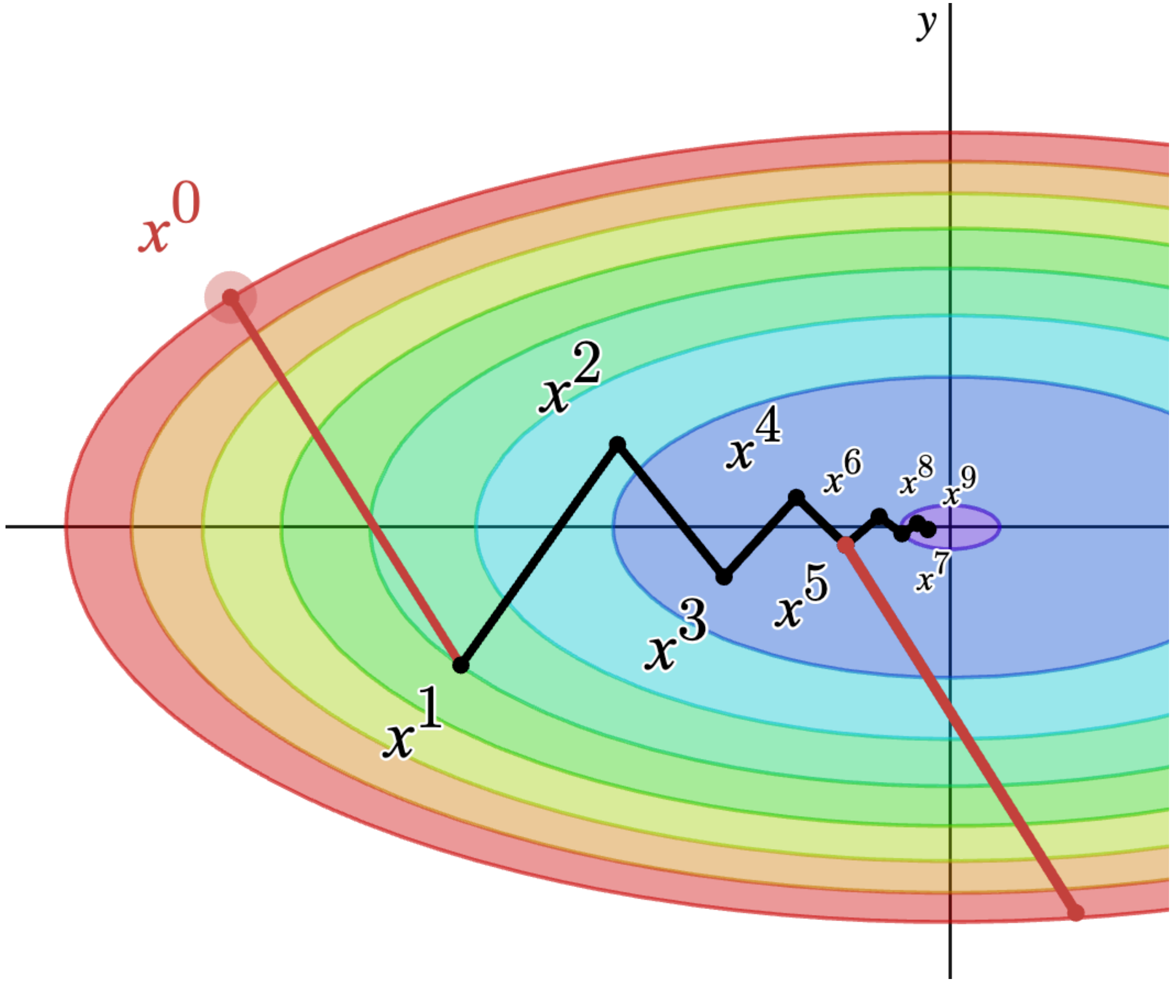
Problem setup

- Optimization objective
- Heterogenous system
- Method (SGD)

Different ways of parallelizing SGD

- Synchronized approaches
- Asynchronous SGD

Recap of what we have covered

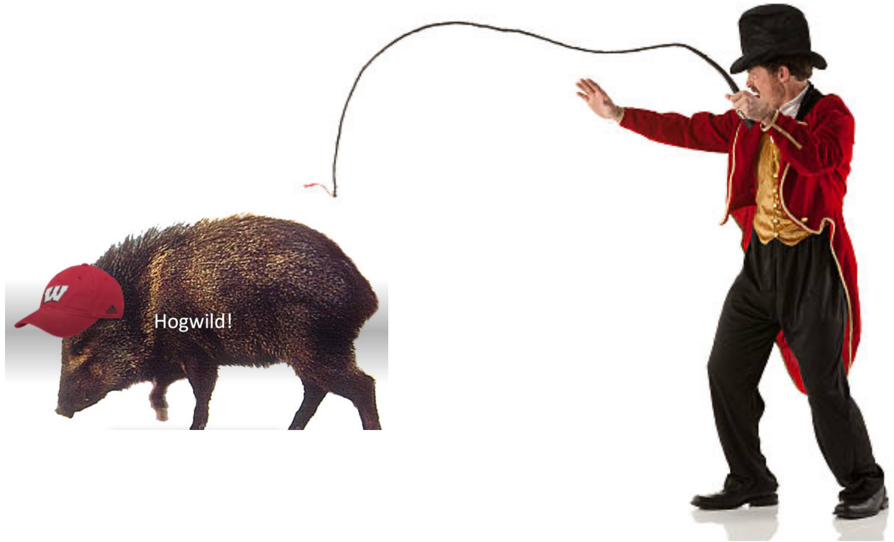


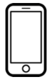
Problem setup
- Optimization objective
- Heterogenous system
- Method (SGD)

Different ways of parallelizing SGD
- Synchronized approaches
- Asynchronous SGD

Problems of ASGD

Recap of what we have covered





$\nabla f\left(x^k; \xi_i^k\right)$

If: $\delta^k < R$

$$x^{k+1} = x^k - \gamma \nabla f\left(x^{k-\delta^k}; \xi_i^{k-\delta^k}\right)$$

Else: Ignore the gradient and send the current point x^k to the worker

Server

- Problem setup
  - Optimization objective
  - Heterogenous system
  - Method (SGD)
- Different ways of parallelizing SGD
  - Synchronized approaches
  - Asynchronous SGD
- Problems of ASGD
- **Ringmaster ASGD**

Alexander Tyurin
Skoltech

Peter Richtárik
KAUST

Closely related papers

Artavazd Maranjyan, Omar Shaikh Omar, Peter Richtárik (2024)
MindFlayer: Efficient asynchronous parallel SGD in the presence of heterogeneous and random worker compute times

Artavazd Maranjyan, El Mehdi Saad, Peter Richtarik, and Francesco Orabona (2025)
ATA: Adaptive Task Allocation for Efficient Resource Management in Distributed Machine Learning

There's still a lot we don't know about asynchronous SGD

"First" ASGD

fully asynchronous SGD

What's next?

2011

2025