Identification-based first-order algorithms for distributed learning

Dmitry Grishchenko

ICCOPT 2019

6 August 2019, Berlin

Collaborators

Collaborators

Collaborators

My supervisors

F. Iutzeler
LJK

J. Malick
CNRS, LJK

Collaborators

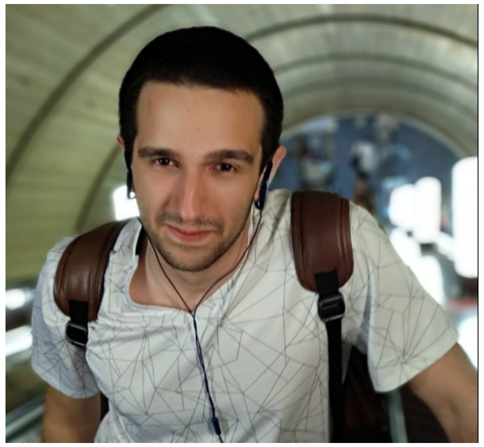
My supervisors

F. Iutzeler
LJK

J. Malick
CNRS, LJK

My student

L. Kochiev
MIPT



Problem

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{M} f_i(x) + r(x)$$

Problem

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{M} f_i(x) + r(x)$$

convex
L-smooth

convex
non-smooth

Problem

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{M} f_i(x) + r(x)$$

convex
L-smooth

convex
non-smooth

In ML context

$$f_i(x) = \sum_{j \in S_i} \pi_i l_j(x)$$

l_j – loss associated with j^{th} example
S_i – i^{th} set of examples
π_i – proportion of examples in i^{th} set

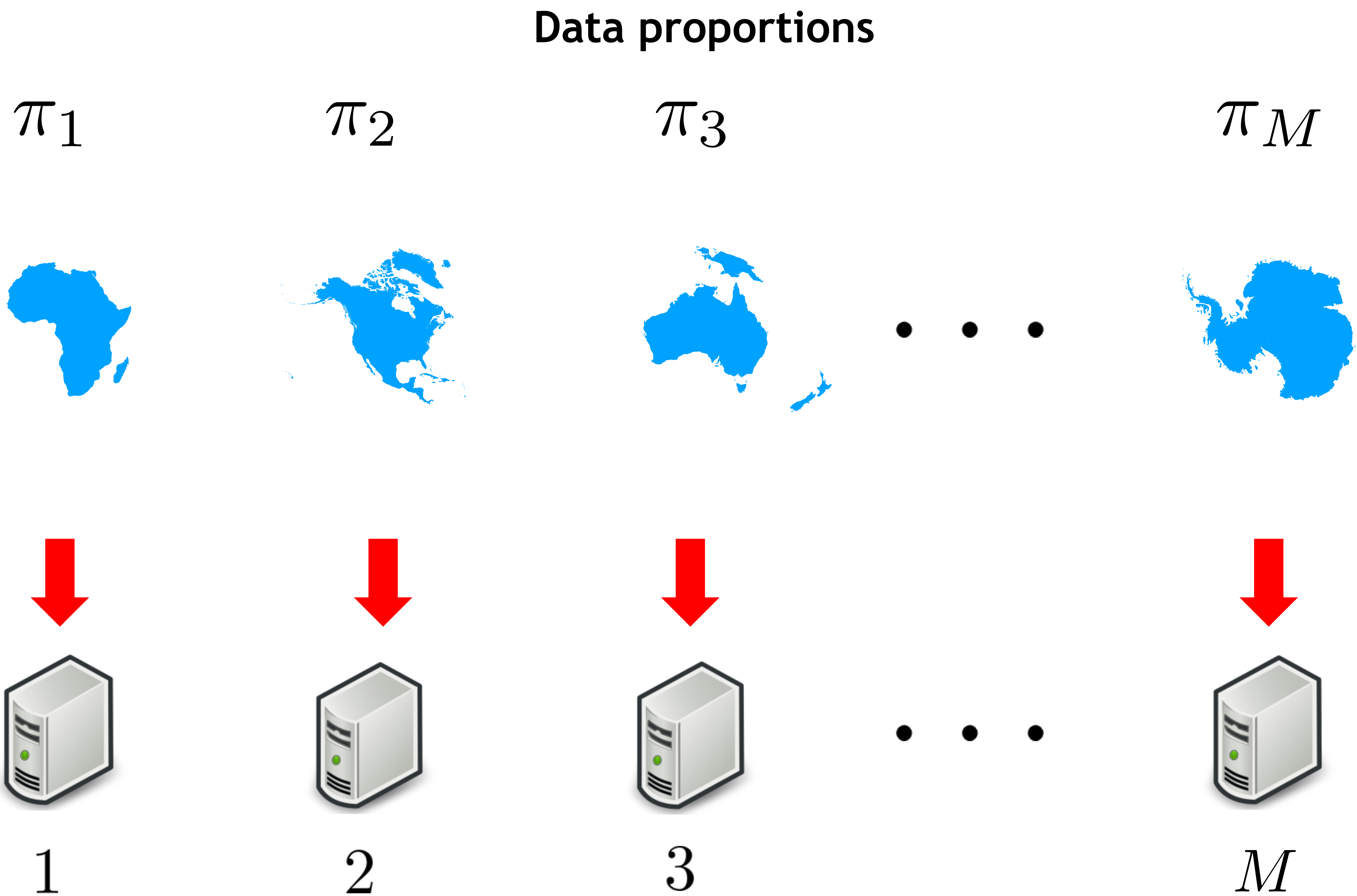
Distribute It!

Distribute It!

Data proportions

π_1 π_2 π_3 π_M

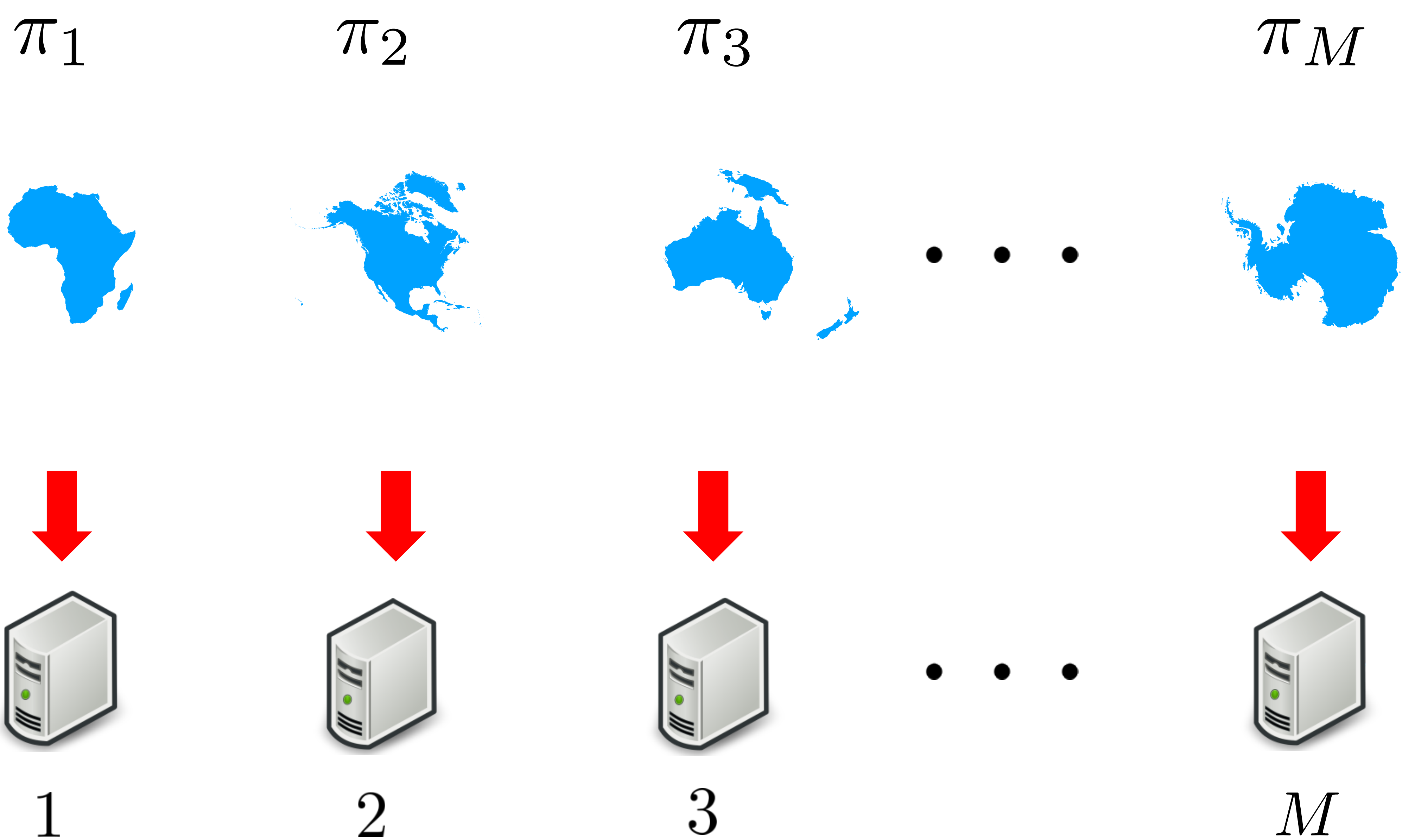
Distribute It!



Distribute It!

Centralized

Data proportions



Parallel or Distributed?

Parallel or Distributed?

	Parallel	Distributed

Parallel or Distributed?

	Parallel	Distributed
Machines		

Parallel or Distributed?

	Parallel	Distributed
Machines	Single	Multiple

Parallel or Distributed?

	Parallel	Distributed
Machines	Single	Multiple
Storage		

Parallel or Distributed?

	Parallel	Distributed
Machines	Single	Multiple
Storage	Limited	Unlimited

Parallel or Distributed?

	Parallel	Distributed
Machines	Single	Multiple
Storage	Limited	Unlimited
Communications		

Parallel or Distributed?

	Parallel	Distributed
Machines	Single	Multiple
Storage	Limited	Unlimited
Communications	Free	Bottleneck

Parallel or Distributed?

	Parallel	Distributed
Machines	Single	
Storage	Limited	Unlimited
Communications	Free	Bottleneck

Multiple

Parallel or Distributed?

	Parallel	Distributed
Machines	Single	
Storage	Limited	Unlimited
Communications	Free	Bottleneck

Multiple

Private data

Parallel or Distributed?

	Parallel	Distributed
Machines	Single	
Storage	Limited	Unlimited
Communications	Free	Bottleneck

Multiple

Private data

Machines = Phones (Federated Learning)

Parallel or Distributed?

	Parallel	Distributed
Machines	Single	
Storage	Limited	Unlimited
Communications	Free	**Bottleneck**

Multiple

Private data

Machines = Phones (Federated Learning)

Parallel or Distributed?

	Parallel	Distributed
Machines	Single	
Storage	Limited	Unlimited
Communications	Free	**Bottleneck**

Multiple

Private data

Machines = Phones (Federated Learning)

Result

Parallel or Distributed?

	Parallel	Distributed
Machines	Single	
Storage	Limited	Unlimited
Communications	Free	**Bottleneck**

Multiple

Private data

Machines = Phones (Federated Learning)

Result <u>Computations are cheaper than communications</u>

Parallel or Distributed?

	Parallel	Distributed
Machines	Single	
Storage	Limited	Unlimited
Communications	Free	**Bottleneck**

Multiple

Private data

Machines = Phones (Federated Learning)

Result <u>Computations are cheaper than communications</u>

Let's make communications cheaper!

The Size Is Important!

The Size Is Important!

The Size Is Important!

Not so many fans

The Size Is Important!

Not so many fans → Everyone can go to the stadium and watch the game

The Size Is Important!

Not so many fans → **Everyone can go to the stadium and watch the game**

The Size Is Important!

~~Not so many fans~~ → Everyone can go to the stadium and watch the game

A lot of fans

The Size Is Important!

~~Not so many fans~~ ~~Everyone can go to the stadium and watch the game~~

A lot of fans → Need to make a ballot for the tickets

The Size Is Important!

~~Not so many fans~~

~~Everyone can go to the stadium and watch the game~~

A lot of fans → Need to make a <u>ballot</u> for the tickets

<u>**sparsification**</u>

Synchronous?

Synchronous?

Drawbacks of synchronous algorithms

Synchronous?

Drawbacks of synchronous algorithms

Wasting of time

All machines have to wait the
moment, when everyone finishes

Synchronous?

Drawbacks of synchronous algorithms

Wasting of time

All machines have to wait the
moment, when everyone finishes

A lot of communications in one time

Master machine communicates with
all machines in the same time

Synchronous?

Drawbacks of synchronous algorithms

Wasting of time

All machines have to wait the moment, when everyone finishes

A lot of communications in one time

Master machine communicates with all machines in the same time

Let's kill 2 birds with one stone

ASynchronous?

Drawbacks of synchronous algorithms

Wasting of time

All machines have to wait the moment, when everyone finishes

A lot of communications in one time

Master machine communicates with all machines in the same time

Let's kill 2 birds with one stone

ASynchronous?

Drawbacks of synchronous algorithms

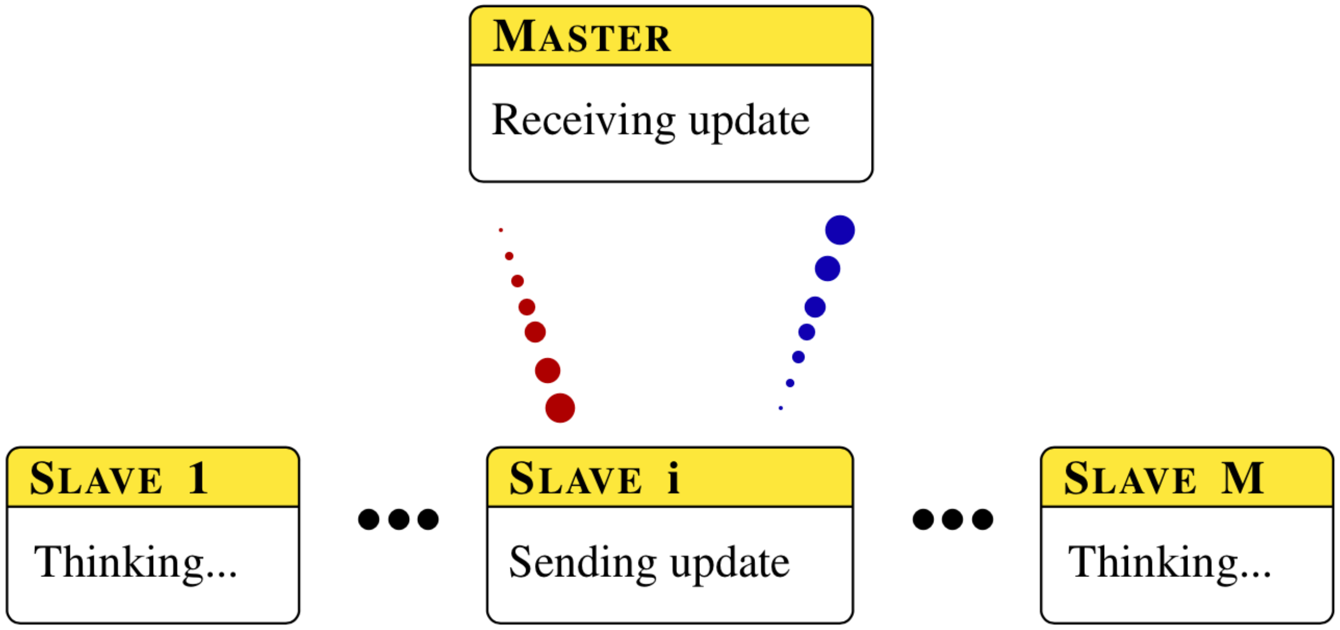
Wasting of time

All machines have to wait the moment, when everyone finishes

A lot of communications in one time

Master machine communicates with all machines in the same time

Let’s kill 2 birds with one stone



Back to Basics

Problems of the form

$$\min_{x \in \mathbb{R}^n} f(x) + r(x)$$

convex
L-smooth

convex
non-smooth

Back to Basics

convex
L-smooth

Problems of the form

$$\min_{x \in \mathbb{R}^n} f(x) + r(x)$$

convex
non-smooth

Call for

First-order proximal methods (for example Proximal Gradient Descent)

Back to Basics

convex
L-smooth

Problems of the form

$$\min_{x \in \mathbb{R}^n} f(x) + r(x)$$

convex
non-smooth

Call for

First-order proximal methods (for example Proximal Gradient Descent)

$$x^{k+1} = \underset{\gamma g}{\mathbf{prox}}(x^k - \gamma \nabla f(x^k))$$

Back to Basics

Problems of the form

$$\min_{x \in \mathbb{R}^n} f(x) + r(x)$$

convex
L-smooth

convex
non-smooth

Call for

First-order proximal methods (for example Proximal Gradient Descent)

$$x^{k+1} = \mathbf{prox}_{\gamma g}(x^k - \gamma \nabla f(x^k))$$

where $\mathbf{prox}_{\gamma g}(x) := \operatorname*{argmin}_u \left\{ g(u) + \frac{1}{2\gamma}\|u - x\|^2 \right\}$

Back to Basics

Problems of the form

$$\min_{x \in \mathbb{R}^n} f(x) + r(x)$$

convex
L-smooth

convex
non-smooth

Call for

First-order proximal methods (for example Proximal Gradient Descent)

$$x^{k+1} = \underset{\gamma g}{\mathbf{prox}}(x^k - \gamma \nabla f(x^k))$$

where $\underset{\gamma g}{\mathbf{prox}}(x) := \underset{u}{\operatorname{argmin}} \left\{ g(u) + \frac{1}{2\gamma}\|u - x\|^2 \right\}$

In synchronous case it will be the same

Each worker send its gradient and master calculate the weighted average

Back to Basics

Problems of the form

$$\min_{x \in \mathbb{R}^n} f(x) + r(x)$$

convex
L-smooth

convex
non-smooth

Call for

First-order proximal methods (for example Proximal Gradient Descent)

$$x^{k+1} = \mathbf{prox}_{\gamma g}(x^k - \gamma \nabla f(x^k))$$

where $\mathbf{prox}_{\gamma g}(x) := \operatorname{argmin}_u \left\{ g(u) + \frac{1}{2\gamma} \|u - x\|^2 \right\}$

In synchronous case it will be the same

Each worker send its gradient and master calculate the weighted average

$$F(x) = \sum_{i=1}^{M} \pi_i f_i(x)$$

Back to Basics

convex
L-smooth

Problems of the form

$$\min_{x \in \mathbb{R}^n} f(x) + r(x)$$

convex
non-smooth

Call for

First-order proximal methods (for example Proximal Gradient Descent)

$$x^{k+1} = \mathbf{prox}_{\gamma g}(x^k - \gamma \nabla f(x^k))$$

where $\mathbf{prox}_{\gamma g}(x) := \operatorname*{argmin}_u \left\{ g(u) + \frac{1}{2\gamma} \|u - x\|^2 \right\}$

In synchronous case it will be the same

Each worker send its gradient and master calculate the weighted average

$$F(x) = \sum_{i=1}^{M} \pi_i f_i(x) \longrightarrow \nabla F(x) = \sum_{i=1}^{M} \pi_i \nabla f_i(x)$$

Delays

Delays

Asynchronous updates bring some delays

Gradient that master receive from worker is computed in one of previous iterate points

Delays

Asynchronous updates bring some delays

Gradient that master receive from worker is computed in one of previous iterate points

Let's define a master's timeline

Delays

Asynchronous updates bring some delays

Gradient that master receive from worker is computed in one of previous iterate points

Let's define a master's timeline

k **- the number of updates master receive**

i^k**- an agent, that communicated with master at time**

Delays

Asynchronous updates bring some delays

Gradient that master receive from worker is computed in one of previous iterate points

Let’s define a master’s timeline

k **- the number of updates master receive**

i^k**- an agent, that communicated with master at time**

Let’s define a worker i timeline

Delays

Asynchronous updates bring some delays

Gradient that master receive from worker is computed in one of previous iterate points

Let’s define a master’s timeline

k **- the number of updates master receive**

i^k**- an agent, that communicated with master at time**

Let’s define a worker i timeline

d_i^k**- time elapsed from the last update**

D_i^k**- the time of penultimate update**

Delays

Asynchronous updates bring some delays

Gradient that master receive from worker is computed in one of previous iterate points

Let's define a master's timeline

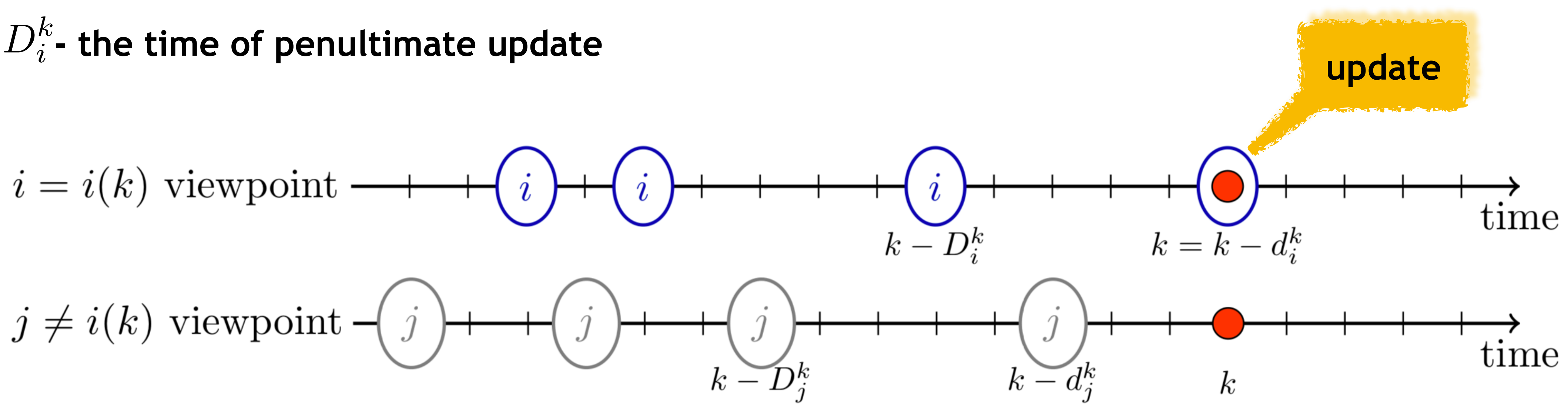
k - the number of updates master receive

i^k- an agent, that communicated with master at time

Let's define a worker i timeline

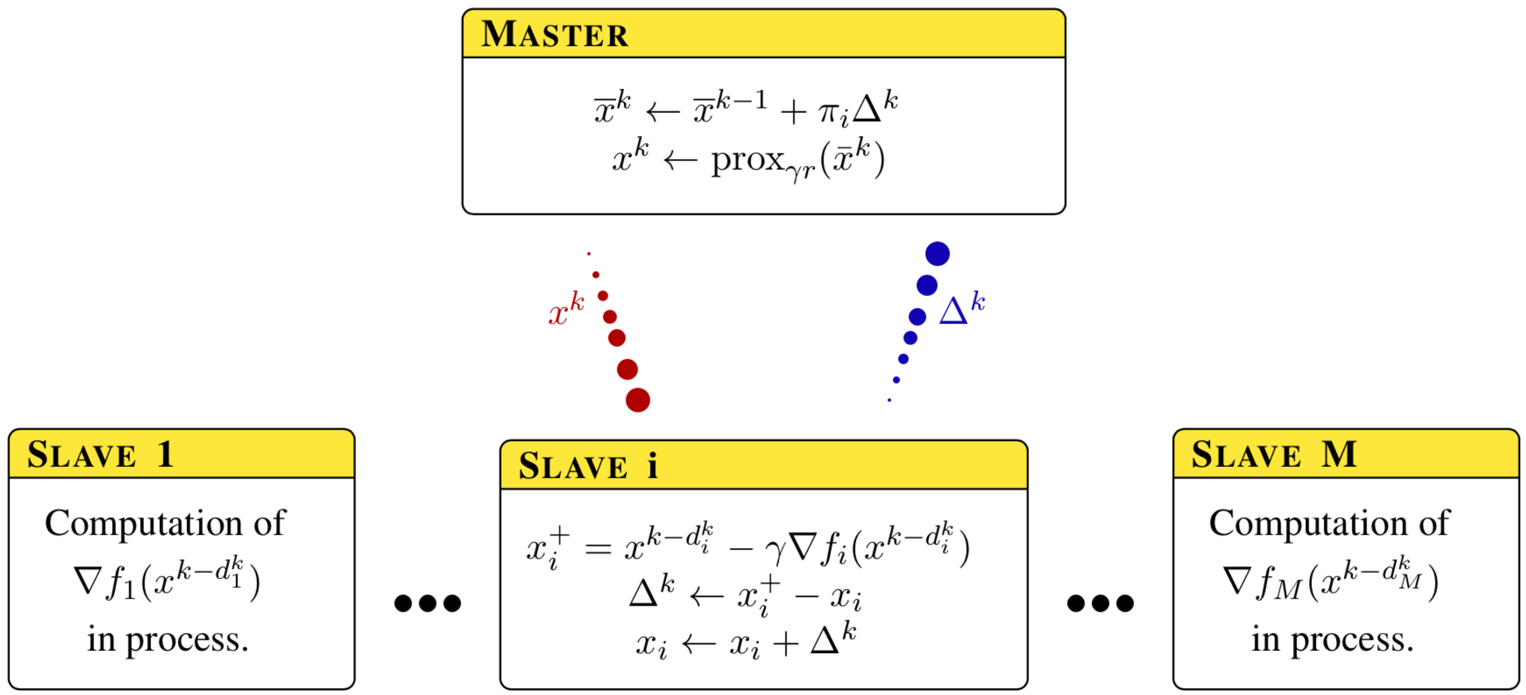
d_i^k- time elapsed from the last update

D_i^k- the time of penultimate update



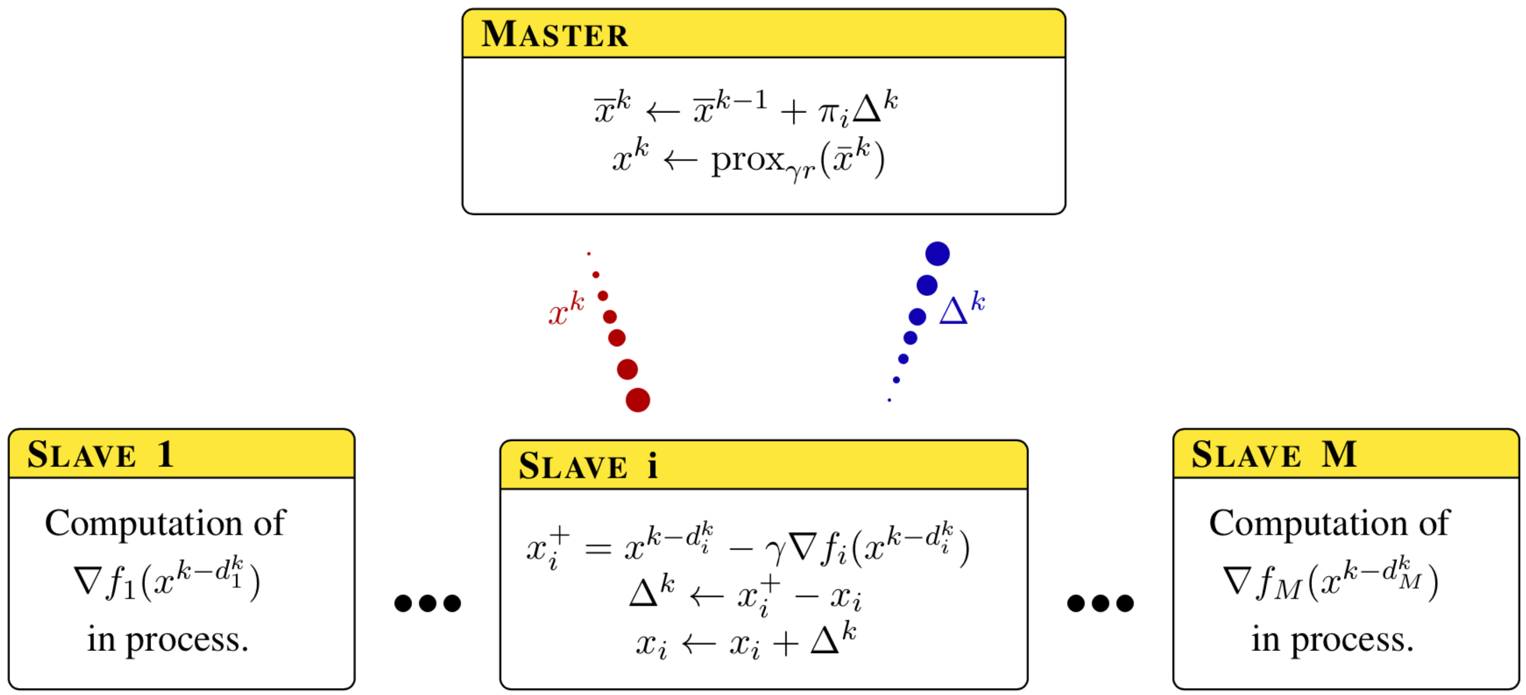
Non-sparsified Algorithm

Non-sparsified Algorithm



Mishchenko, K., Iutzeler, F., Malick, J., & Amini, M. R. (2018, July).
A delay-tolerant proximal-gradient algorithm for distributed learning.
In *International Conference on Machine Learning* (pp. 3584-3592).

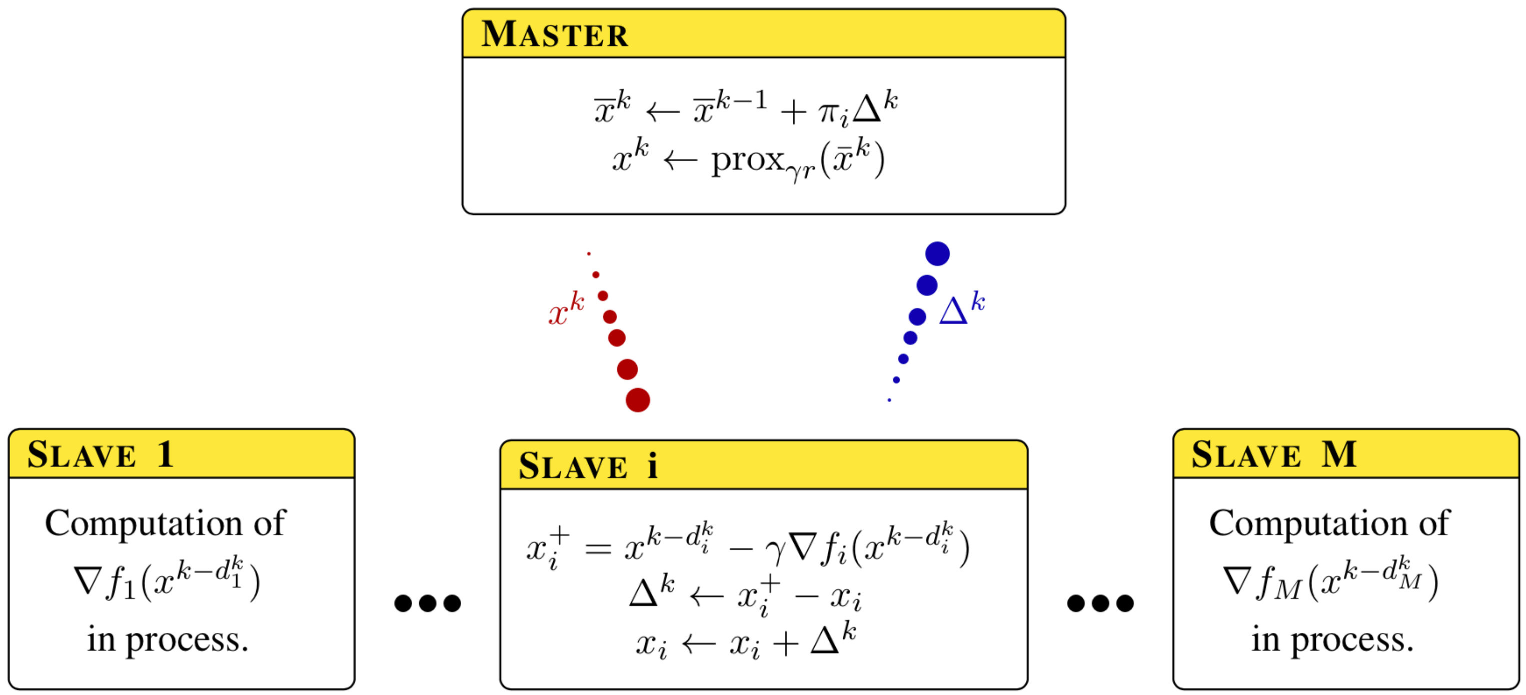
Non-sparsified Algorithm



If dimension is very big, it is very expansive to send full gradient

Mishchenko, K., Iutzeler, F., Malick, J., & Amini, M. R. (2018, July). A delay-tolerant proximal-gradient algorithm for distributed learning. In *International Conference on Machine Learning* (pp. 3584-3592).

Non-sparsified Algorithm



If dimension is very big, it is very expansive to send full gradient

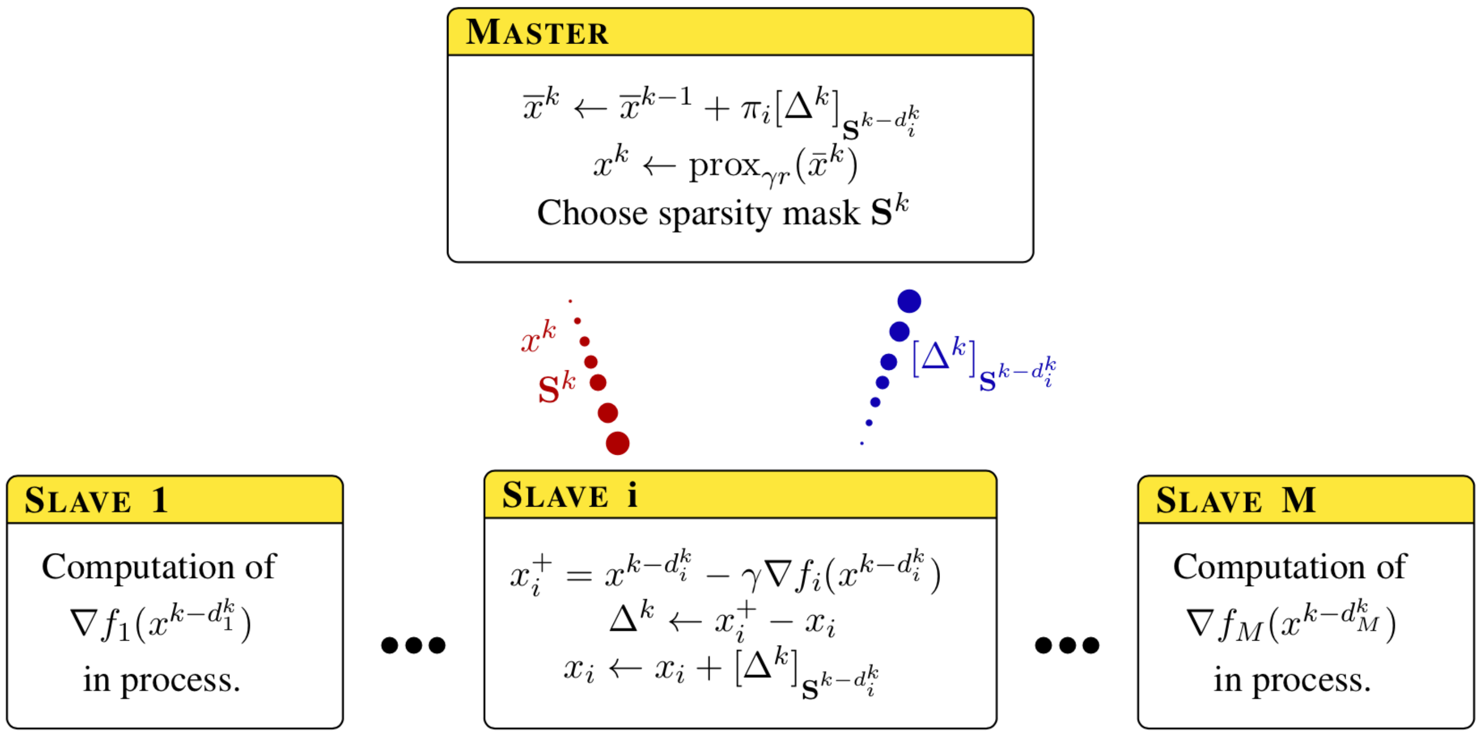
If regulariser is sparsity enforcing, no need to sparsify master → worker

Mishchenko, K., Iutzeler, F., Malick, J., & Amini, M. R. (2018, July). A delay-tolerant proximal-gradient algorithm for distributed learning. In *International Conference on Machine Learning* (pp. 3584-3592).

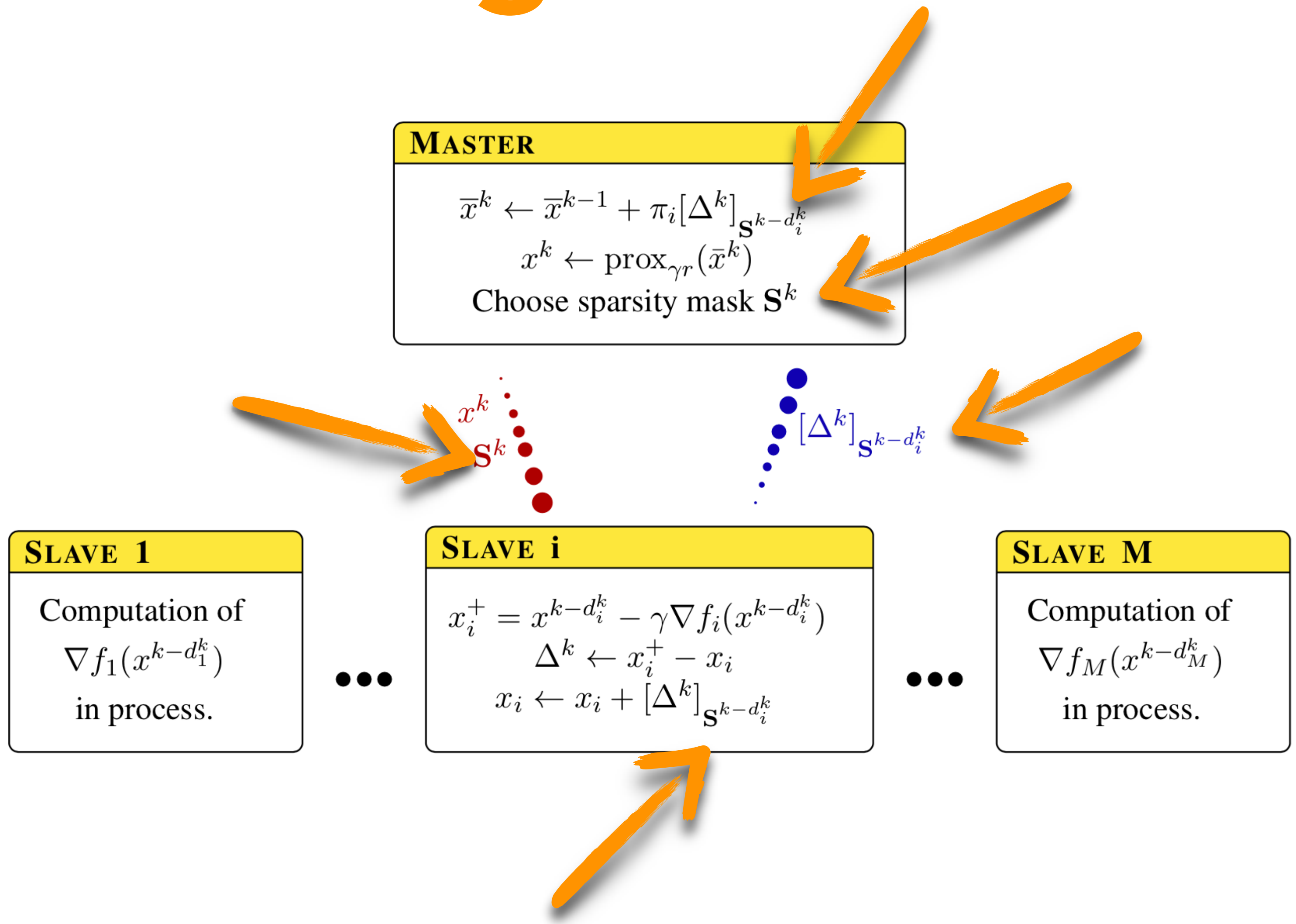
Fadili J., Malick J., Peyré G. Sensitivity analysis for mirror-stratifiable convex functions SIAM Journal on Optimization. – 2018. – T. 28. – №. 4. – C. 2975-3000.

Sparsified Algorithm

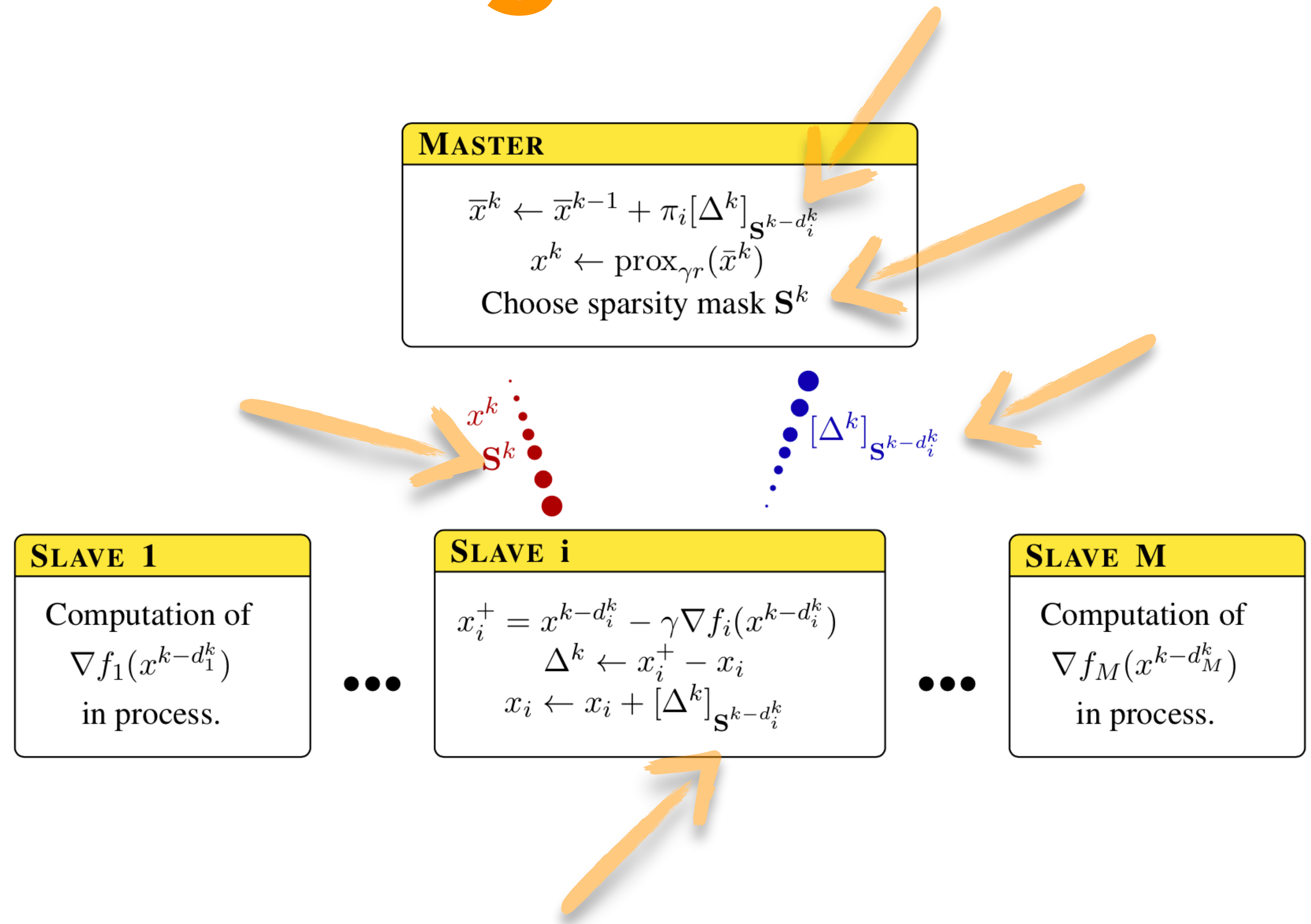
Sparsified Algorithm



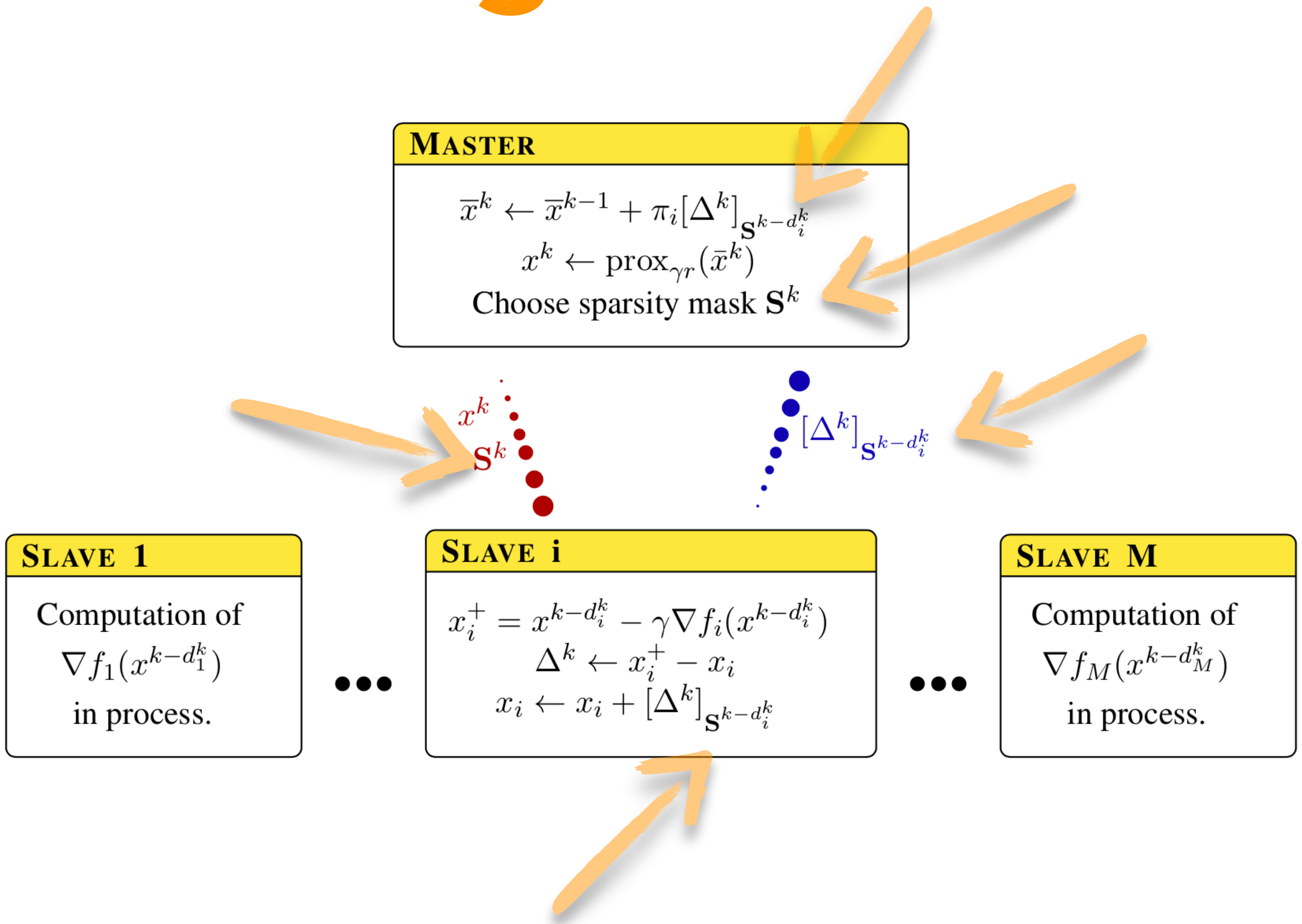
Sparsified Algorithm



Sparsified Algorithm

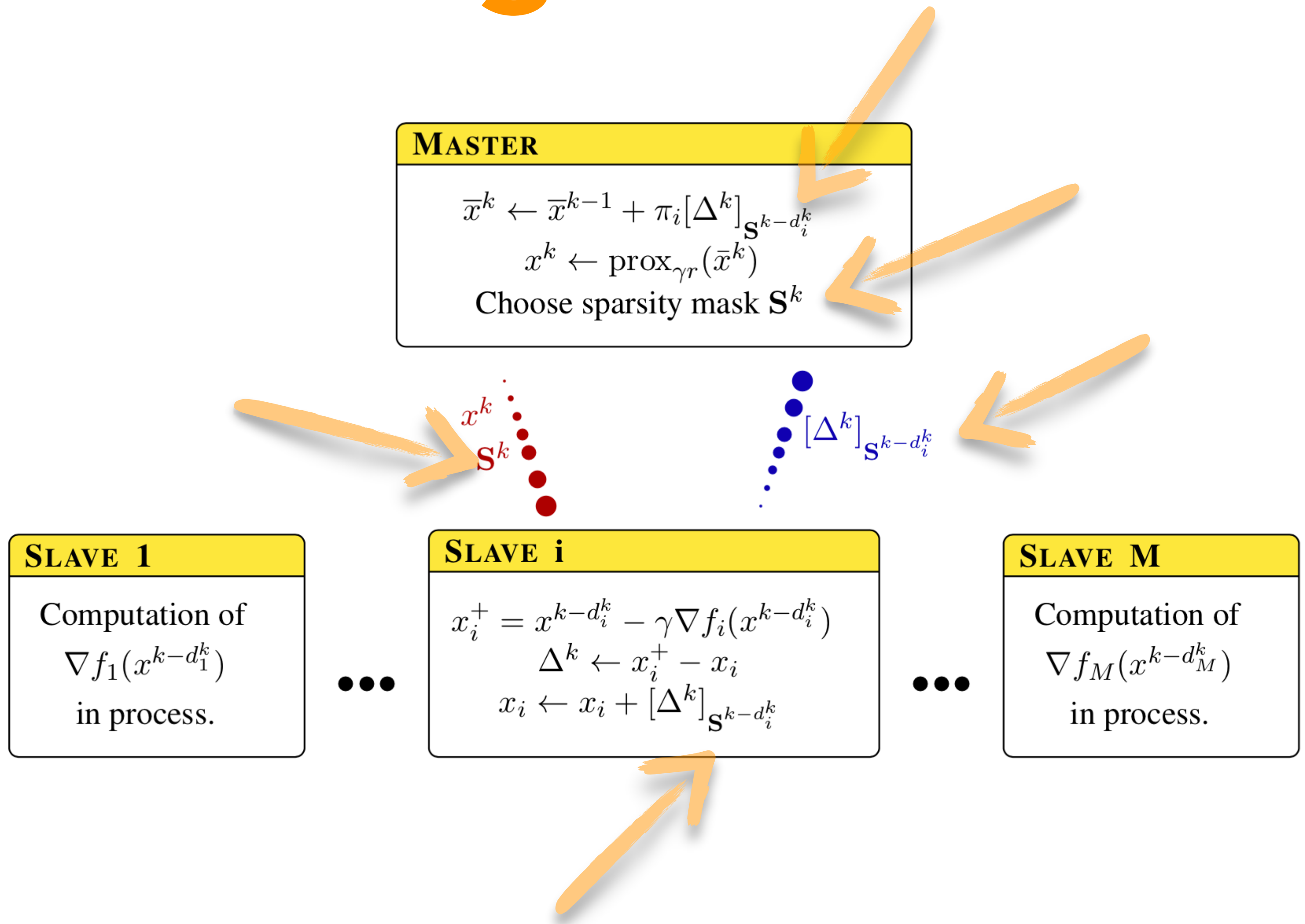


Sparsified Algorithm



How to chose the sparsity mask?

Sparsified Algorithm

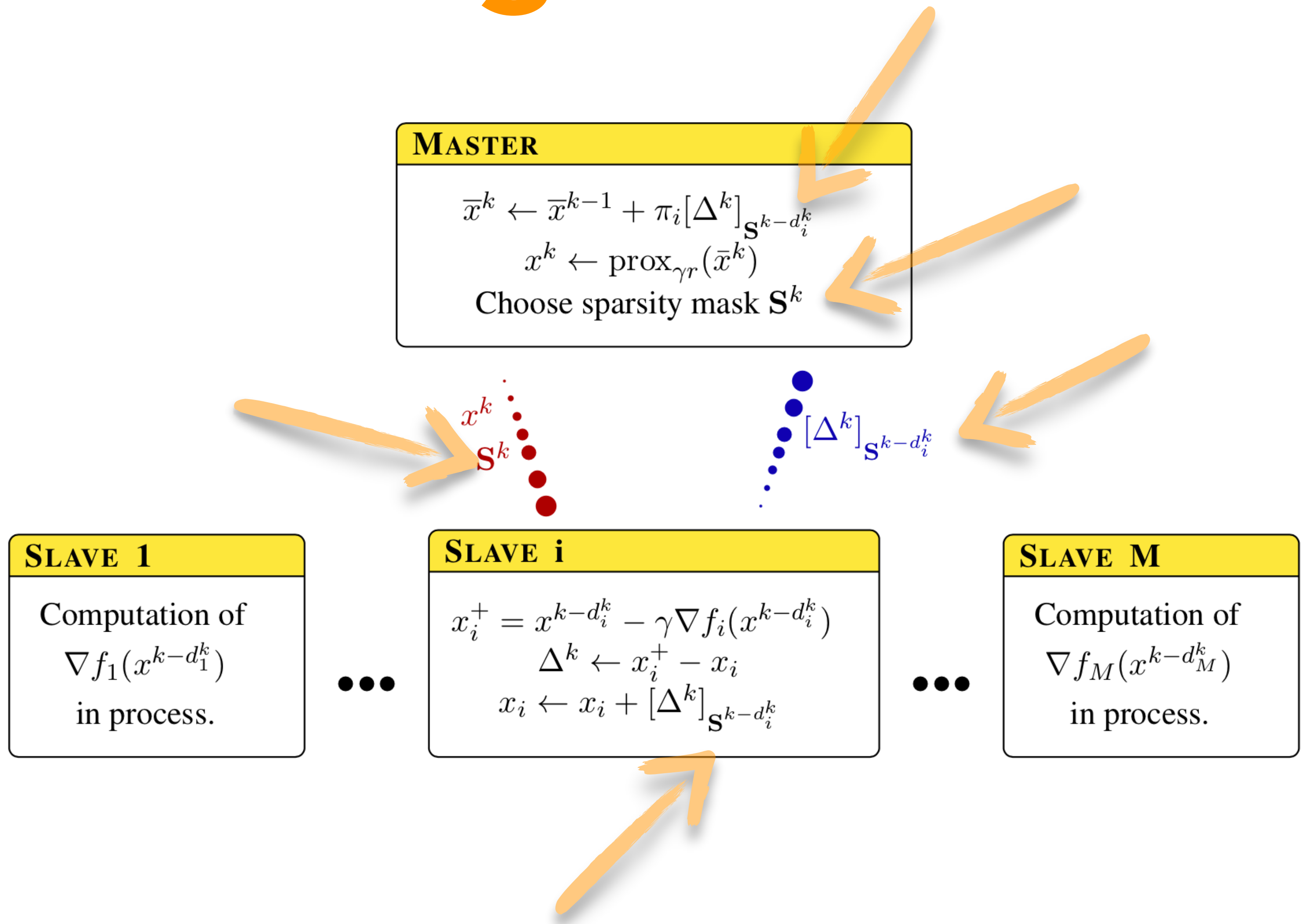


How to chose the sparsity mask?

Random uniform coordinate selection (Coordinate Descent with uniform probabilities)

Nesterov Y. Efficiency of coordinate descent methods on huge-scale optimization problems
SIAM Journal on Optimization. – 2012. – Т. 22. – №. 2. – С. 341-362.

Sparsified Algorithm



How to chose the sparsity mask?

Random uniform coordinate selection (Coordinate Descent with uniform probabilities)

OR…

Nesterov Y. Efficiency of coordinate descent methods on huge-scale optimization problems
SIAM Journal on Optimization. – 2012. – Т. 22. – №. 2. – С. 341-362.

Back to Ballot

It’s an example of identification from real world

Back to Ballot

1 wave - most loyal fans (the biggest amount of games visited)

It’s an example of identification from real world

Back to Ballot

1 wave - most loyal fans (the biggest amount of games visited)

2 wave (after cancellations) - uniformly random from all

It’s an example of identification from real world

Back to Ballot

1 wave - most loyal fans (the biggest amount of games visited)

2 wave (after cancellations) - uniformly random from all

This system gives a chance for everyone to watch a game

It's an example of identification from real world

Back to Ballot

1 wave - most loyal fans (the biggest amount of games visited)

2 wave (after cancellations) - uniformly random from all

This system gives a chance for everyone to watch a game

Most loyal have probability 1, all the others much smaller

It’s an example of identification from real world

Core Idea

, .

* - l_1-regularizer enforces coordinate sparsity

Core Idea

Let's introduce our adaptive way of mask selection for l_1-regularizer

,

.

*** - l_1-regularizer enforces coordinate sparsity**

Core Idea

Let's introduce our adaptive way of mask selection for l_1-regularizer

1 - select all the coordinates that are in master's current iterate

,

.

*** - l_1-regularizer enforces coordinate sparsity**

Core Idea

Let's introduce our adaptive way of mask selection for l_1-regularizer

1 - select all the coordinates that are in master's current iterate

2 - add some other coordinates with some probabilities

, .

*** - l_1-regularizer enforces coordinate sparsity**

Core Idea

Let’s introduce our adaptive way of mask selection for l_1-regularizer

1 - select all the coordinates that are in master’s current iterate

2 - add some other coordinates with some probabilities

As a result, all the coordinates from the support of the final solution would be selected

*** - l_1-regularizer enforces coordinate sparsity**

Core Idea

Let's introduce our adaptive way of mask selection for l_1-regularizer

1 - select all the coordinates that are in master's current iterate

2 - add some other coordinates with some probabilities

As a result, all the coordinates from the support of the final solution would be selected

Dimension reductions without loss of speed (but only from some finite moment of time)

,

.

*** - l_1-regularizer enforces coordinate sparsity**

Core Idea

Let's introduce our adaptive way of mask selection for l_1-regularizer

1 - select all the coordinates that are in master's current iterate

2 - add some other coordinates with some probabilities

As a result, all the coordinates from the support of the final solution would be selected

Dimension reductions without loss of speed (but only from some finite moment of time)

Theoretical result (for s.c. objective)

$$\mathbb{E}\|x_i^k - x_i^\star\|^2 \le \left(2 - 2\frac{\gamma\mu L}{\mu + L} - \min_i p_i\right) \max_{j=1,..,M} \mathbb{E}\left\|x_j^{k-D_i^k} - x_j^\star\right\|^2$$

where $k \in [k_m, k_{m+1})$, $k_{m+1} = \min\left\{k : k - D_i^k \ge k_m \text{ for all } i\right\}$, **and stepsize** $\gamma \in (0, 2/(\mu + L)]$.

Grishchenko, D., Iutzeler, F., Malick, J., & Amini, M. R. (2018).
Asynchronous Distributed Learning with Sparse Communications and Identification.
arXiv preprint arXiv:1812.03871.

*** - l_1-regularizer enforces coordinate sparsity**

Core Idea v.2

$$\mathbb{E}\|x_i^k - x_i^\star\|^2 \leq \left(2 - 2\frac{\gamma\mu L}{\mu + L} - \min_i p_i\right) \max_{j=1,..,M} \mathbb{E}\left\|x_j^{k-D_i^k} - x_j^\star\right\|^2$$

Core Idea v.2

$$\mathbb{E}\|x_i^k - x_i^\star\|^2 \leq \left(2 - 2\frac{\gamma\mu L}{\mu + L} - \min_i p_i\right) \max_{j=1,\ldots,M} \mathbb{E}\left\|x_j^{k-D_i^k} - x_j^\star\right\|^2$$

should be less than 1!

.

Core Idea v.2

$$\mathbb{E}\|x_i^k - x_i^\star\|^2 \le \left(2 - 2\frac{\gamma\mu L}{\mu + L} - \min_i p_i\right) \max_{j=1,..,M} \mathbb{E}\left\|x_j^{k-D_i^k} - x_j^\star\right\|^2$$

should be less than 1!

Bound on the minimal probability that depends on problem conditioning

$$\min_i p_i \ge 1 - \frac{2\gamma\mu L}{\mu L}$$

.

Core Idea v.2

$$\mathbb{E}\|x_i^k - x_i^\star\|^2 \le \left(2 - 2\frac{\gamma\mu L}{\mu+L} - \min_i p_i\right) \max_{j=1,..,M} \mathbb{E}\left\|x_j^{k-D_i^k} - x_j^\star\right\|^2$$

should be less than 1!

Bound on the minimal probability that depends on problem conditioning

$$\min_i p_i \ge 1 - \frac{2\gamma\mu L}{\mu L}$$

For ill-conditioned problems there is NO sparsification

Core Idea v.2

$$\mathbb{E}\|x_i^k - x_i^\star\|^2 \le \left(2 - 2\frac{\gamma\mu L}{\mu + L} - \min_i p_i\right) \max_{j=1,..,M} \mathbb{E}\left\|x_j^{k-D_i^k} - x_j^\star\right\|^2$$

should be less than 1!

Bound on the minimal probability that depends on problem conditioning

$$\min_i p_i \ge 1 - \frac{2\gamma\mu L}{\mu L}$$

For ill-conditioned problems there is NO sparsification

Let us use l_2-regularizer to recondition the initial problem

Catalyst

Catalyst

Input: $\boldsymbol{x_0} \in \mathbb{R}^n$, smoothing parameter $\boldsymbol{\kappa}$, optimization method $\mathcal{M}$, $\boldsymbol{y_0} = \boldsymbol{x_0}$, $\boldsymbol{q} = \frac{\mu}{\mu + \boldsymbol{\kappa}}$

Output: $\boldsymbol{x}^\star \in \operatorname{argmin}_{x \in \mathbb{R}^n} f(x)$

while *desired stopping criterion is not satisfied* **do**

Find $\boldsymbol{x_k}$ using $\mathcal{M}$

$$\boldsymbol{x_k} \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}_{\boldsymbol{\varepsilon^k}}} \{ h_{\boldsymbol{k}}(x) \triangleq f(x) + \frac{\boldsymbol{\kappa}}{2} \| x - \boldsymbol{y_{k-1}} \|^2 \}$$

Compute $\boldsymbol{\alpha_k} \in (0;1)$ from $\boldsymbol{\alpha_k}^2 = (1 - \boldsymbol{\alpha_k}) \boldsymbol{\alpha_{k-1}}^2 + q \boldsymbol{\alpha_k}$

Compute $\boldsymbol{y_k}$ using $\boldsymbol{\beta_k}$ from $(0, 1)$

$$\boldsymbol{y_k} = \boldsymbol{x_k} + \boldsymbol{\beta_k} (\boldsymbol{x_k} - \boldsymbol{x_{k-1}}),$$

where

$$\boldsymbol{\beta_k} = \frac{\boldsymbol{\alpha_{k-1}} (1 - \boldsymbol{\alpha_{k-1}})}{\boldsymbol{\alpha_{k-1}}^2 + \boldsymbol{\alpha_k}}$$

end

Lin, H., Mairal, J., & Harchaoui, Z. (2015).A universal catalyst for first-order optimization. In *Advances in neural information processing systems* (pp. 3384-3392).

Catalyst

Input: $\boldsymbol{x_0} \in \mathbb{R}^n$, smoothing parameter $\boldsymbol{\kappa}$, optimization method $\mathcal{M}$, $\boldsymbol{y_0} = \boldsymbol{x_0}$, $\boldsymbol{q} = \frac{\mu}{\mu+\boldsymbol{\kappa}}$
Output: $\boldsymbol{x}^\star \in \operatorname{argmin}_{x\in\mathbb{R}^n} f(x)$
while *desired stopping criterion is not satisfied* **do**

Find $\boldsymbol{x_k}$ using $\mathcal{M}$

$$\boldsymbol{x_k} \in \underset{x\in\mathbb{R}^n}{\operatorname{argmin}_{\boldsymbol{\varepsilon^k}}} \{h_{\boldsymbol{k}}(x) \triangleq f(x) + \frac{\boldsymbol{\kappa}}{2}\|x - \boldsymbol{y_{k-1}}\|^2\}$$

Compute $\boldsymbol{\alpha_k} \in (0;1)$ from $\boldsymbol{\alpha_k}^2 = (1 - \boldsymbol{\alpha_k})\boldsymbol{\alpha_{k-1}}^2 + q\boldsymbol{\alpha_k}$
Compute $\boldsymbol{y_k}$ using $\boldsymbol{\beta_k}$ from $(0,1)$

$$\boldsymbol{y_k} = \boldsymbol{x_k} + \boldsymbol{\beta_k}(\boldsymbol{x_k} - \boldsymbol{x_{k-1}}),$$

where

$$\boldsymbol{\beta_k} = \frac{\boldsymbol{\alpha_{k-1}}(1 - \boldsymbol{\alpha_{k-1}})}{\boldsymbol{\alpha_{k-1}}^2 + \boldsymbol{\alpha_k}}$$

end

The only requirement: method $\mathcal{M}$ that solves $\min\limits_x h_k(x)$ with <u>linear</u> rate.

Lin, H., Mairal, J., & Harchaoui, Z. (2015).A universal catalyst for first-order optimization. In *Advances in neural information processing systems* (pp. 3384-3392).

Inner Method

Inner Method

WORKER i

Initialize $\boldsymbol{\kappa}$
Receive x, y from master
Initialize $x_i = x_i^+ = x$
Update objective function

$$h_i(\cdot) = f_i(\cdot) + \frac{\boldsymbol{\kappa}}{2}\|\cdot - y\|^2$$

while *not interrupted by master* **do**
- $[x_i^+]_{\mathbf{S}} \leftarrow [x - \gamma \nabla h_i(x)]_{\mathbf{S}}$
- $\Delta \leftarrow x_i^+ - x_i$
- Send $[\Delta]_{\mathbf{S}}$ to master
- $[x_i]_{\mathbf{S}} \leftarrow [x_i^+]_{\mathbf{S}}$
- Receive x and $\mathbf{S}$ from master

end

Inner Method

WORKER i

Initialize $\boldsymbol{\kappa}$

Receive x, y from master

Initialize $x_i = x_i^+ = x$

Update objective function

$$h_i(\cdot) = f_i(\cdot) + \frac{\boldsymbol{\kappa}}{2}\|\cdot - y\|^2$$

while *not interrupted by master* **do**

- $[x_i^+]_{\mathbf{S}} \leftarrow [x - \gamma \nabla h_i(x)]_{\mathbf{S}}$
- $\Delta \leftarrow x_i^+ - x_i$
- Send $[\Delta]_{\mathbf{S}}$ to master
- $[x_i]_{\mathbf{S}} \leftarrow [x_i^+]_{\mathbf{S}}$
- Receive x and $\mathbf{S}$ from master

end

Inner Method

MASTER

Reinitialize $k = 1, \bar{x}_l^0 = \boldsymbol{y_{l-1}}$
Broadcast $x_l^0 = \mathbf{prox}_{\gamma g}\left(\bar{x}_l^0\right), \boldsymbol{y_{l-1}}$
while *stopping criterion is not satisfied* **do**
  Receive $[\Delta^k]_{\mathbf{S}^{k-D^k_{i^k}}}$ from agent i^k
  $\bar{x}_l^k \leftarrow \bar{x}_l^{k-1} + \pi_i [\Delta^k]_{\mathbf{S}^{k-D^k_{i^k}}}$
  $x_l^k \leftarrow \mathbf{prox}_{\gamma r}(\bar{x}_l^k)$
  Draw sparsity mask $\mathbf{S}^k$
  Send $x_l^k, \mathbf{S}^k$ to agent i^k
  $k \leftarrow k+1$
end
while *some workers compute* **do**
  Receive $[\Delta^k]_{\mathbf{S}^{k-D^k_{i^k}}}$ from agent i^k
  $\bar{x}_l^k \leftarrow \bar{x}_l^{k-1} + \pi_i [\Delta^k]_{\mathbf{S}^{k-D^k_{i^k}}}$
  Stop worker
end
$\boldsymbol{x_l} \leftarrow \mathbf{prox}_{\gamma g}(\bar{x}_l^k)$

WORKER i

Initialize $\boldsymbol{\kappa}$
Receive x, y from master
Initialize $x_i = x_i^+ = x$
Update objective function

$$h_i(\cdot) = f_i(\cdot) + \frac{\boldsymbol{\kappa}}{2}\| \cdot - y\|^2$$

while *not interrupted by master* **do**
  $[x_i^+]_{\mathbf{S}} \leftarrow [x - \gamma \nabla h_i(x)]_{\mathbf{S}}$
  $\Delta \leftarrow x_i^+ - x_i$
  Send $[\Delta]_{\mathbf{S}}$ to master
  $[x_i]_{\mathbf{S}} \leftarrow [x_i^+]_{\mathbf{S}}$
  Receive x and $\mathbf{S}$ from master
end

Fly in the Ointment

Fly in the Ointment

Required size of update*

*** - in the beginning (when the support is big enough) coordinates from the support could be also selected with some probability**

Fly in the Ointment

Required size of update*

Bound on probability to guarantee such size

*** - in the beginning (when the support is big enough) coordinates from the support could be also selected with some probability**

Fly in the Ointment

Required size of update*

Bound on probability to guarantee such size

Bound on $\boldsymbol{\kappa} \geq \dfrac{(1 - \min_i p_i)L - \mu}{\min_i p_i}$

* - in the beginning (when the support is big enough) coordinates from the support could be also selected with some probability

Fly in the Ointment

Required size of update*

Bound on probability to guarantee such size

Bound on $\boldsymbol{\kappa} \geq \dfrac{(1 - \min_i p_i)L - \mu}{\min_i p_i}$

Requires more restarts than Catalyst with optimal parameter $\boldsymbol{\kappa}^\star$

if minimal probability is small enough and it's fair to tell about sparsification

* - in the beginning (when the support is big enough) coordinates from the support could be also selected with some probability

Communication Metric

after identification happened

Communication Metric

after identification happened

ρ **- the sparsity of the optimal solution**

Communication Metric

after identification happened

ρ **- the sparsity of the optimal solution**

ρn **- the size of communication from master**

Communication Metric

after identification happened

ρ **- the sparsity of the optimal solution**

ρn **- the size of communication from master**

$|\boldsymbol{S}^k| \geq \rho n(1 + (1-\rho)\min_i p_i)$ **- … from worker**

Communication Metric

after identification happened

ρ - the sparsity of the optimal solution

ρn - the size of communication from master

$|\boldsymbol{S}^k| \geq \rho n(1 + (1-\rho)\min_i p_i)$ - … from worker

Results:

No reason to select coordinates with different probabilities

Communication Metric

after identification happened

ρ - the sparsity of the optimal solution

ρn - the size of communication from master

$|\boldsymbol{S}^k| \geq \rho n(1 + (1-\rho)\min_i p_i)$ - … from worker

Results:

No reason to select coordinates with different probabilities

An optimal uniform probability is the following

$$p^\star = \frac{2\rho}{1+3\rho}$$

Theoretical Result

Theoretical Result

Theorem (s.c. case)

Consider the sparsity of the optimal solution ρ.

Choose $p = \dfrac{2\rho}{1+3\rho}$**, and corresponding** $\boldsymbol{\kappa} = \dfrac{(1-p)L-\mu}{p}$**.**

Then for any $\gamma \in \left(0, \dfrac{2}{\mu + L + 2\kappa}\right]$ **ASPY-DR algorithm**

converges $\tilde{O}\left(\sqrt{\dfrac{1+\rho}{\rho}}\right)$ **times faster in communications**

metrics than nonsparsified one.

Numerical Experiments

Numerical Experiments

Software: Python + MPI4py

Amount of workers: 4

Problem: standard regularized logistic regression

Dataset: Madelon

Final sparsity: 0.05

Optimal probability: ≈ 0.9

Numerical Experiments

Software: Python + MPI4py

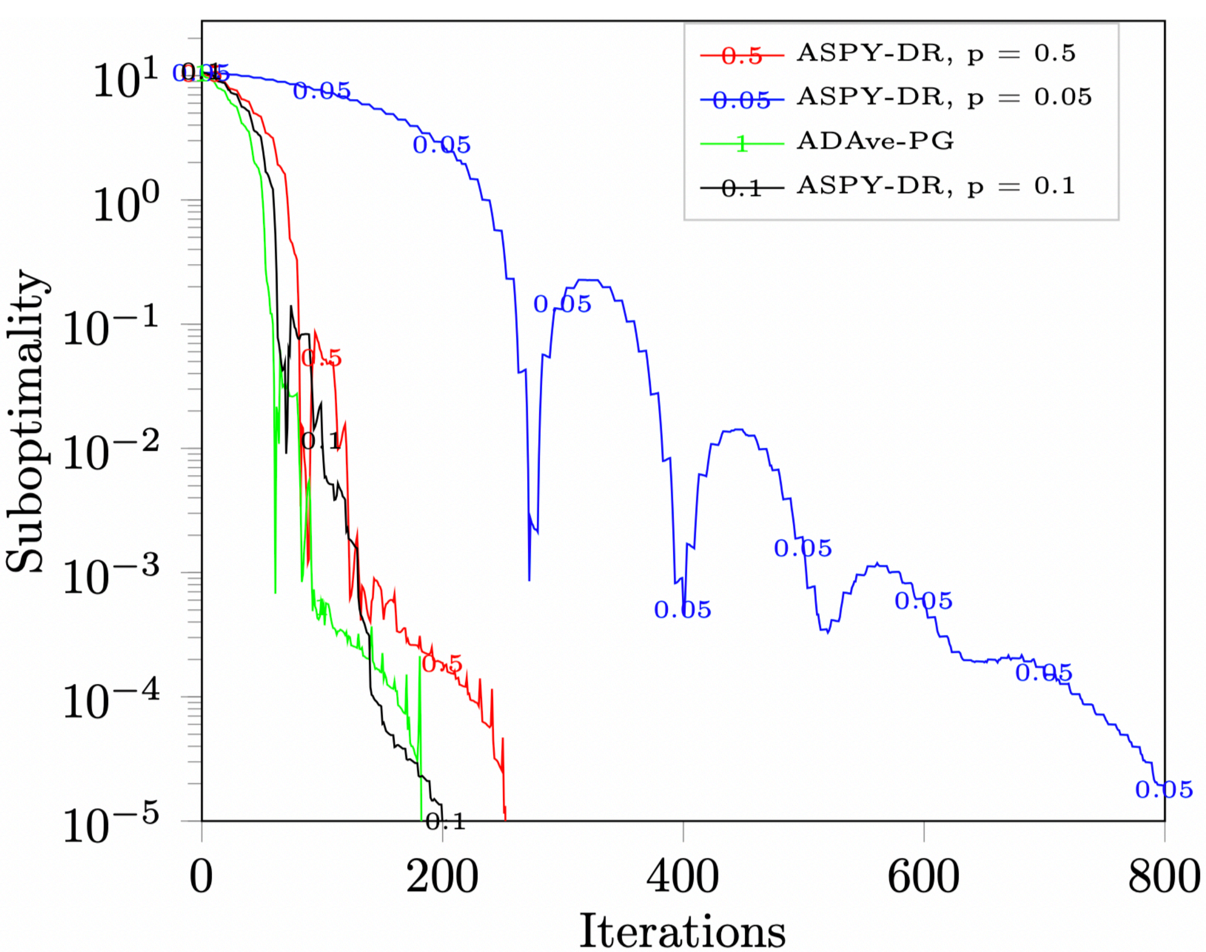
Amount of workers: 4

Problem: standard regularized logistic regression

Dataset: Madelon

Final sparsity: 0.05

Optimal probability: ≈ 0.9



Numerical Experiments

Software: Python + MPI4py

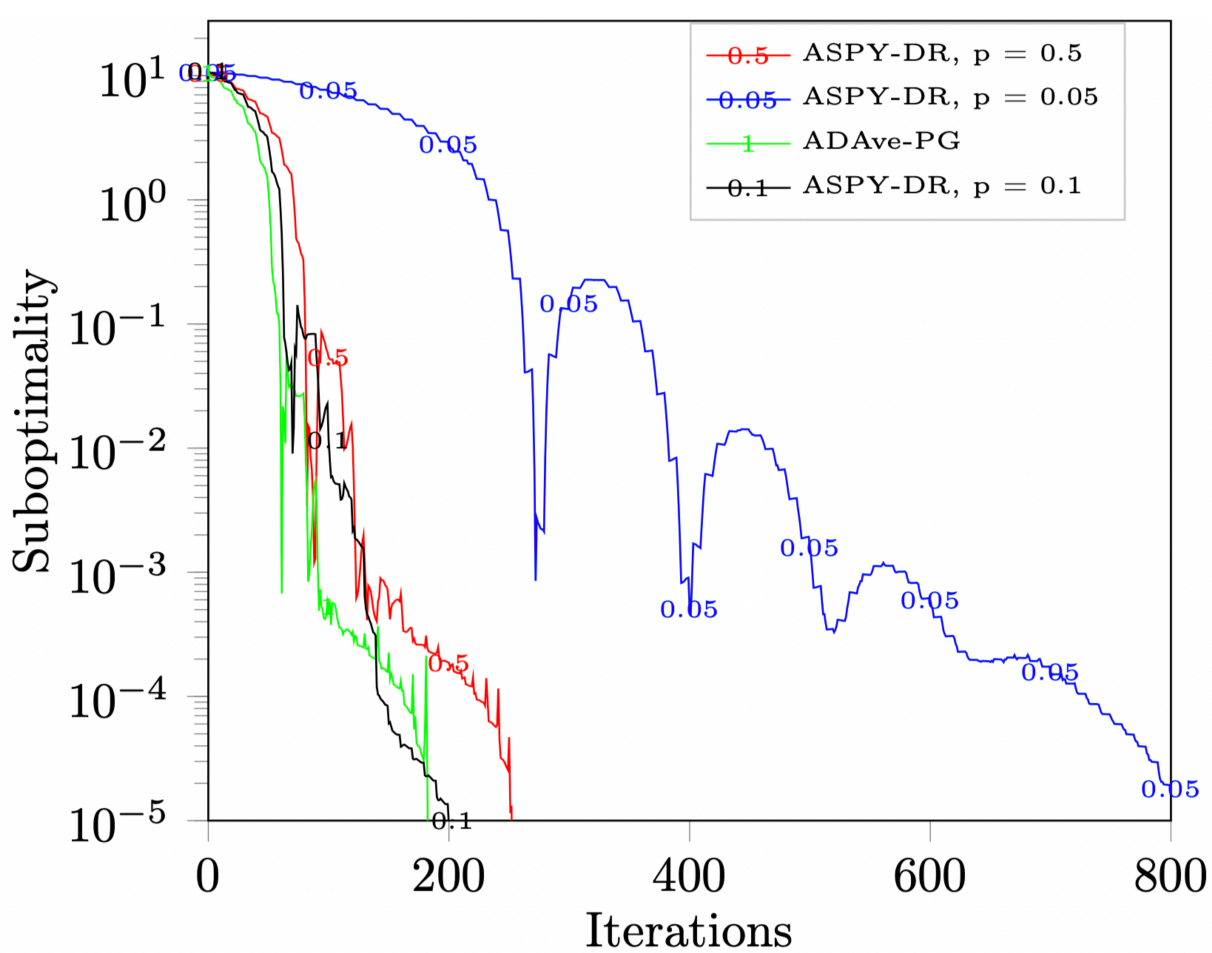
Amount of workers: 4

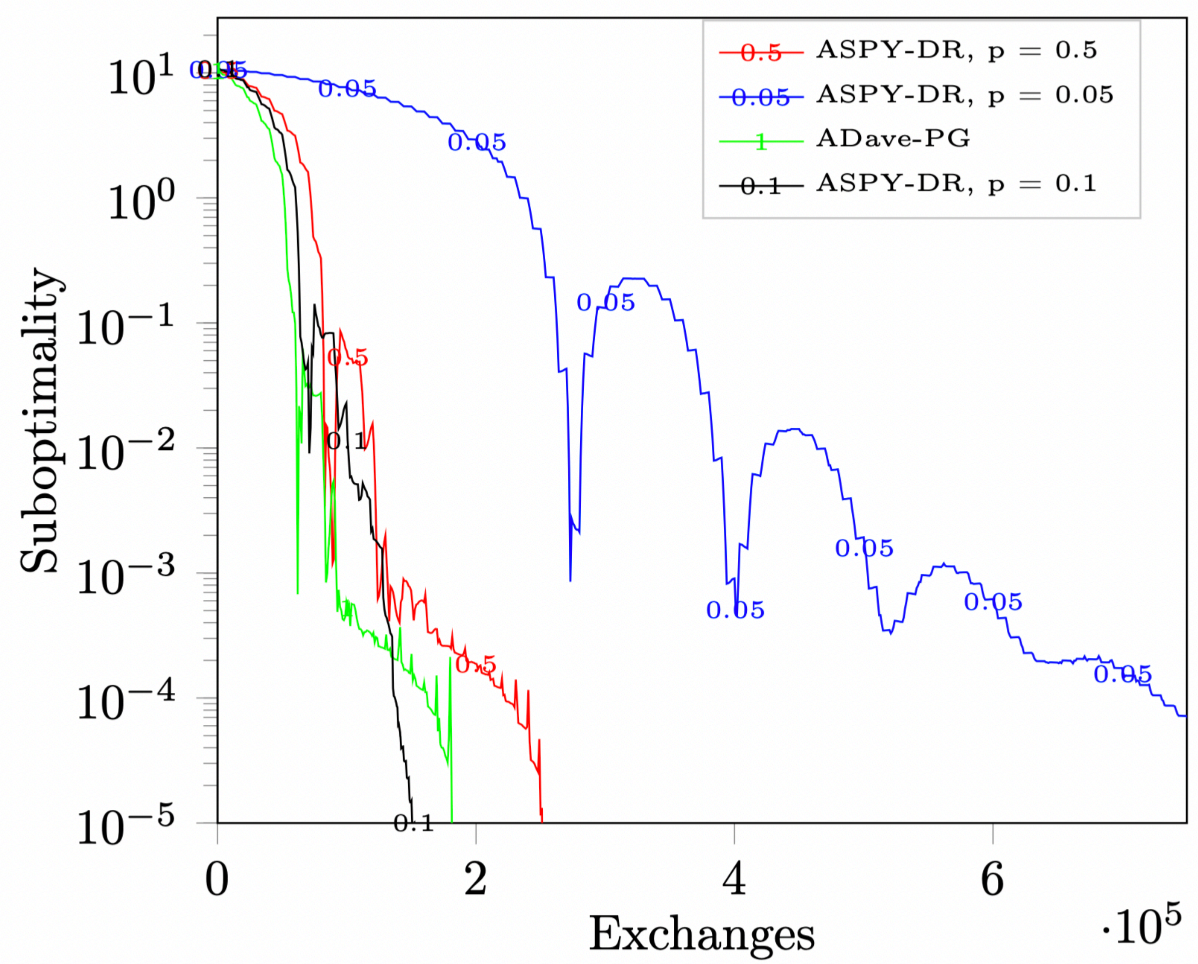
Problem: standard regularized logistic regression

Dataset: Madelon

Final sparsity: 0.05

Optimal probability: ≈ 0.9





Thank You For Your Attention