# COCOA Communication-Efficient Coordinate Ascent

Virginia Smith

*Martin Jaggi, Martin Takáč, Jonathan Terhorst,*
*Sanjay Krishnan, Thomas Hofmann, & Michael I. Jordan*

# LARGE-SCALE OPTIMIZATION

# LARGE-SCALE OPTIMIZATION

## COCOA

LARGE-SCALE OPTIMIZATION

COCOA

RESULTS

**LARGE-SCALE OPTIMIZATION**

COCOA

RESULTS

# Machine Learning with Large Datasets

# Machine Learning with Large Datasets

# Machine Learning with Large Datasets

YouTube
IMAGENET
amazon.com
NETFLIX
illumina
facebook
twitter
Google
VISA

image/music/video tagging
document categorization
item recommendation
click-through rate prediction
sequence tagging
protein structure prediction
sensor data prediction
spam classification
fraud detection

# Machine Learning Workflow

# Machine Learning Workflow

DATA & PROBLEM

*classification, regression, collaborative filtering, …*

# Machine Learning Workflow

DATA & PROBLEM

*classification, regression, collaborative filtering, …*

MACHINE LEARNING MODEL

*logistic regression, lasso, support vector machines, …*

# Machine Learning Workflow

DATA & PROBLEM

*classification, regression, collaborative filtering, …*

MACHINE LEARNING MODEL

*logistic regression, lasso, support vector machines, …*

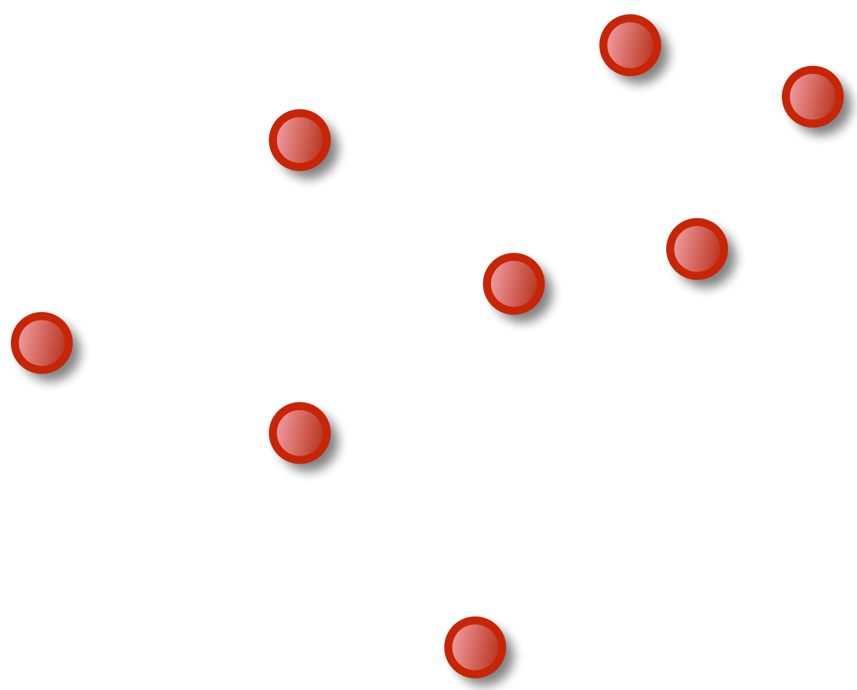
OPTIMIZATION ALGORITHM

*gradient descent, coordinate descent, Newton's method, …*

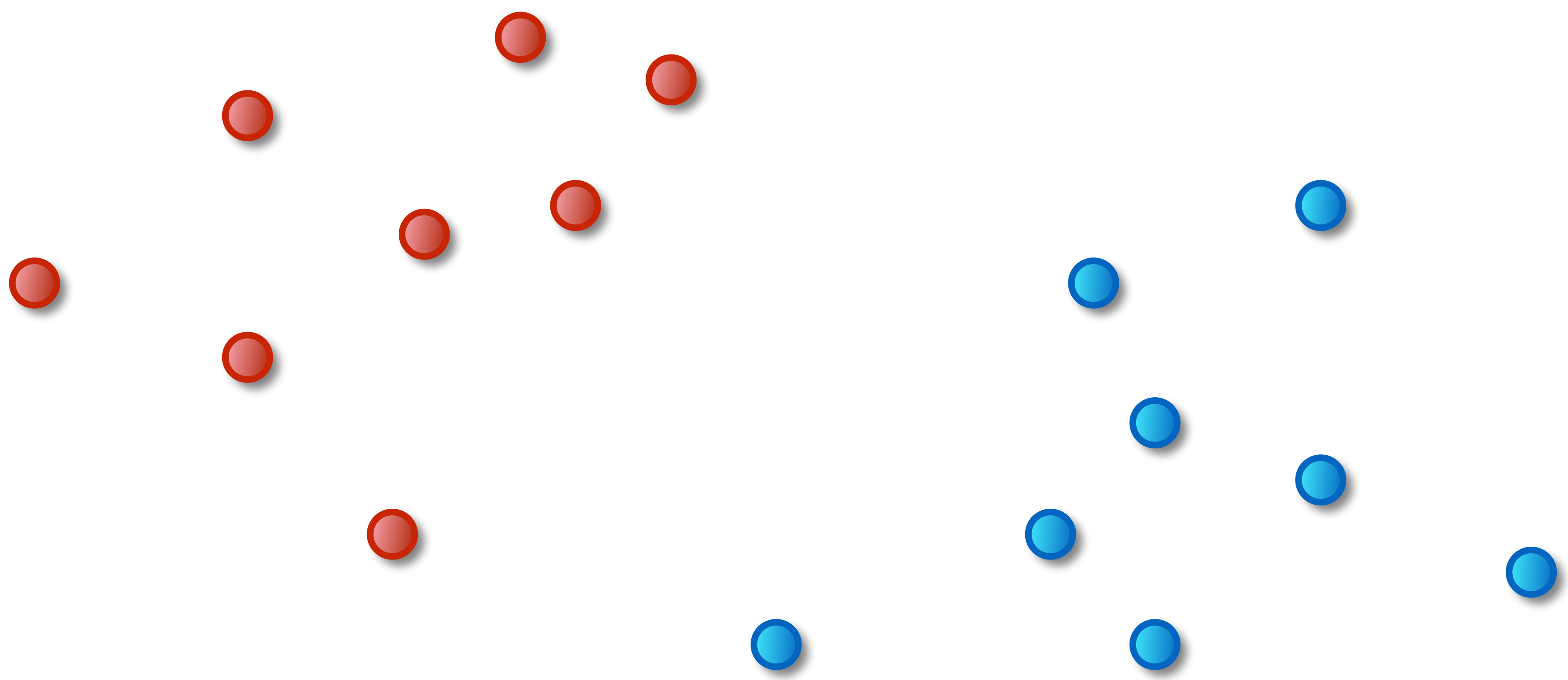
# Example: SVM Classification

# Example: SVM Classification

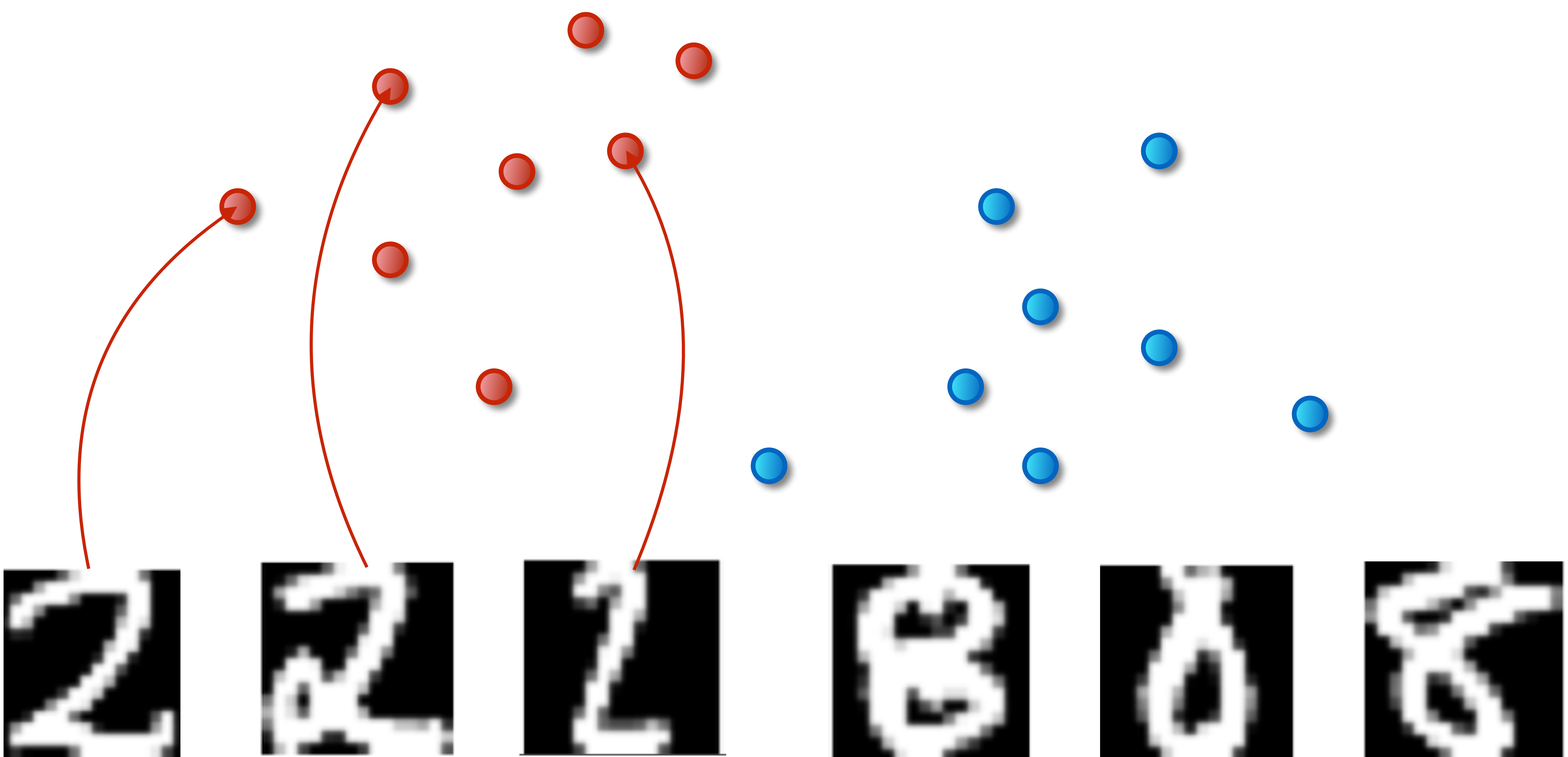
# Example: SVM Classification



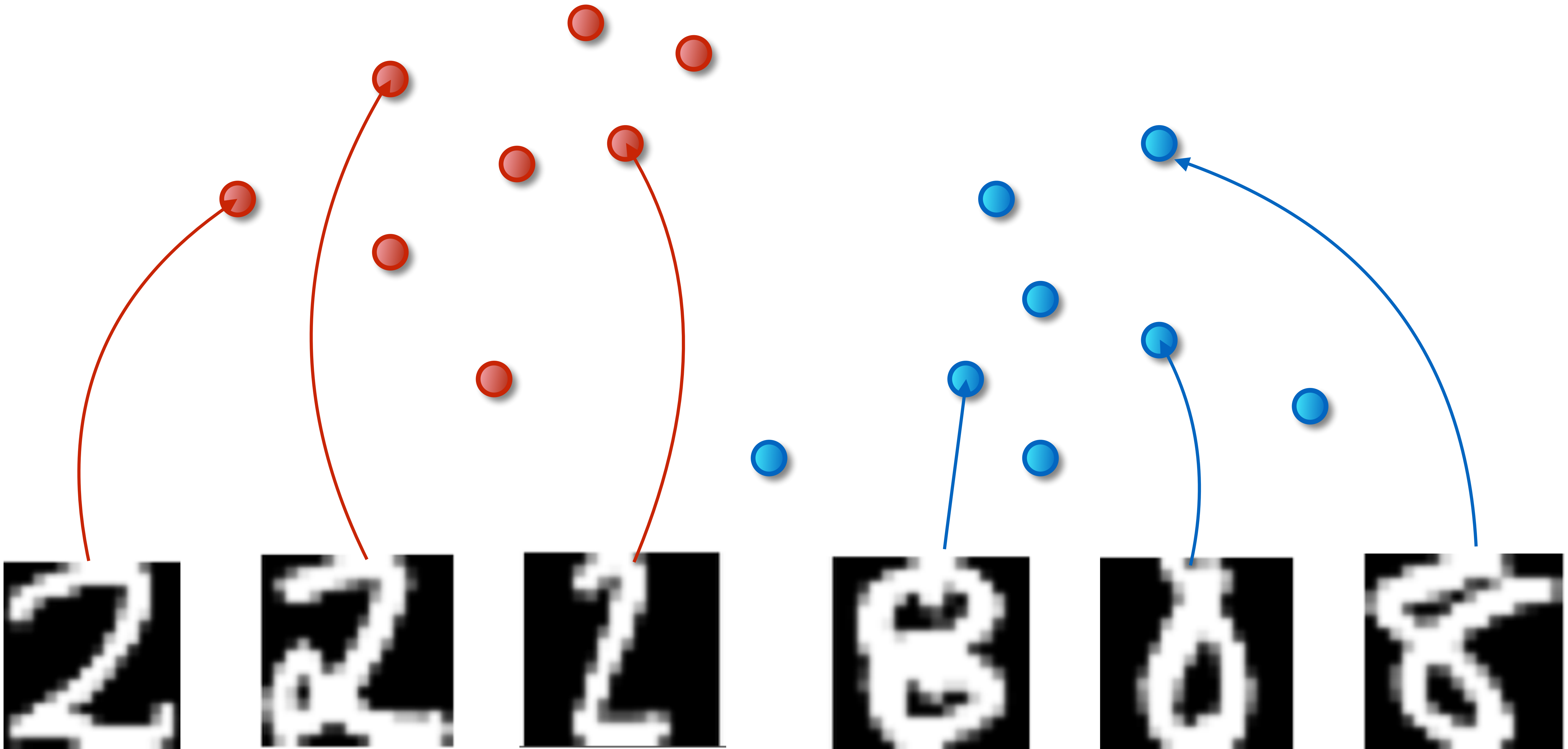
# Example: SVM Classification



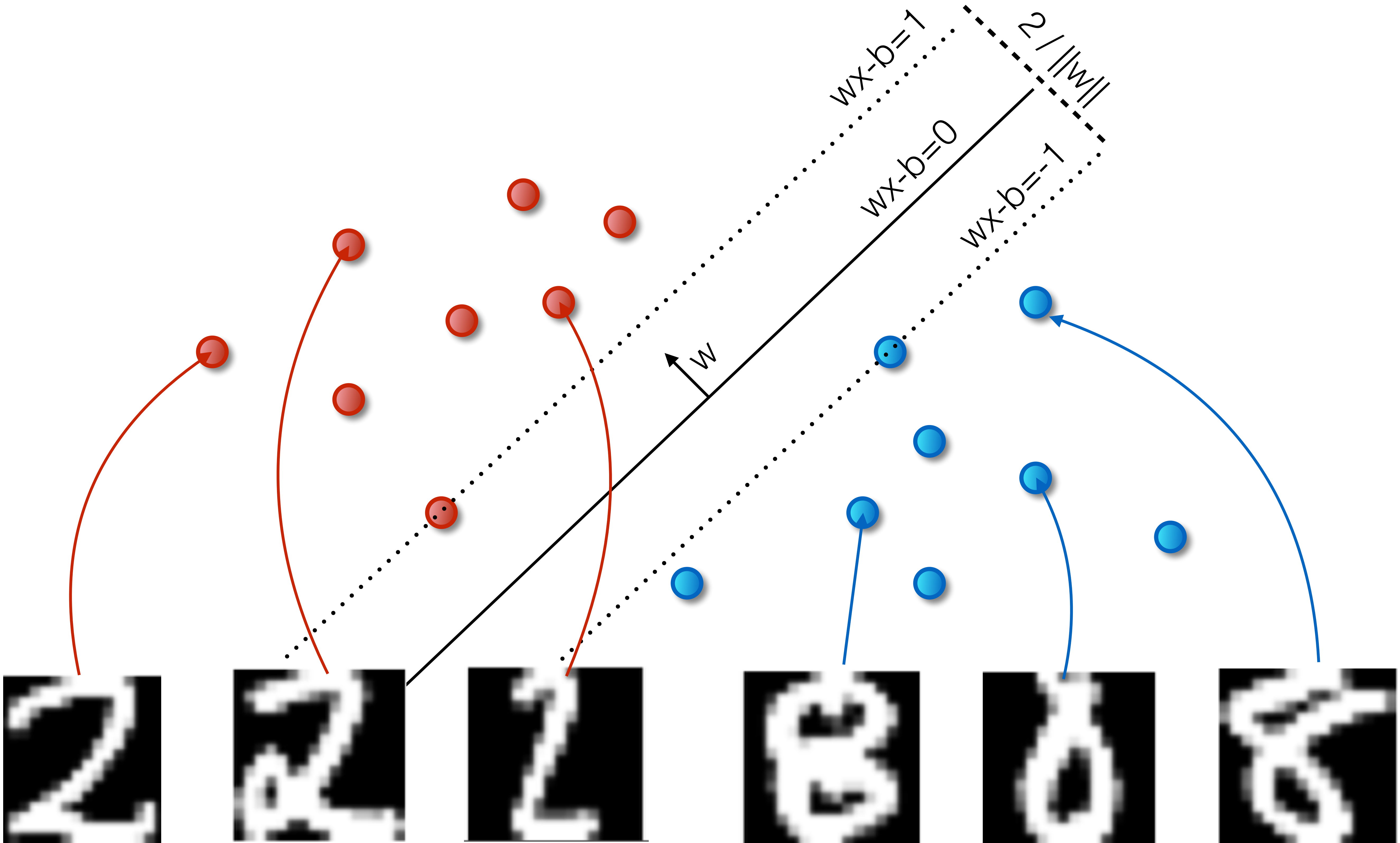
# Example: SVM Classification



# Example: SVM Classification

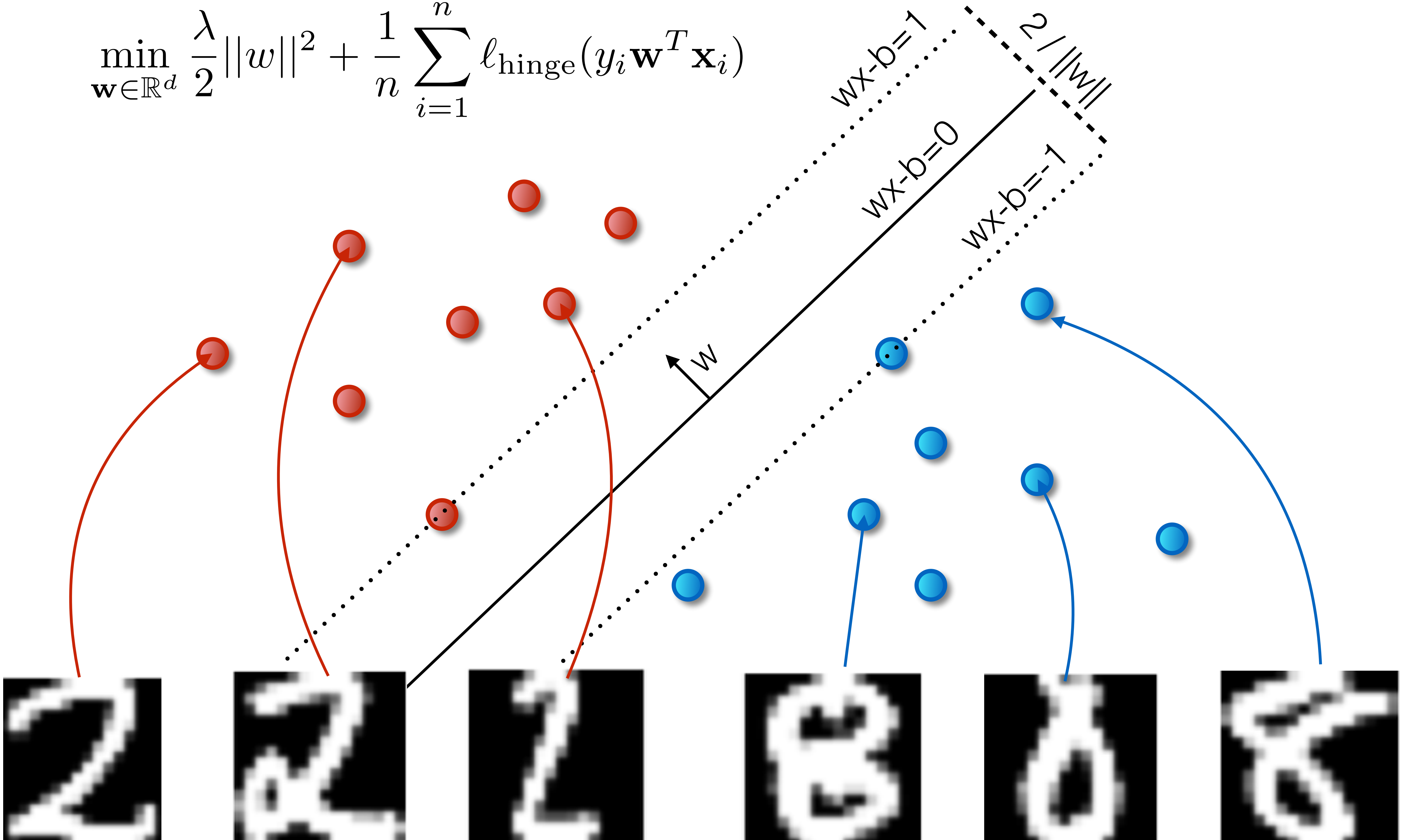


# Example: SVM Classification



# Example: SVM Classification

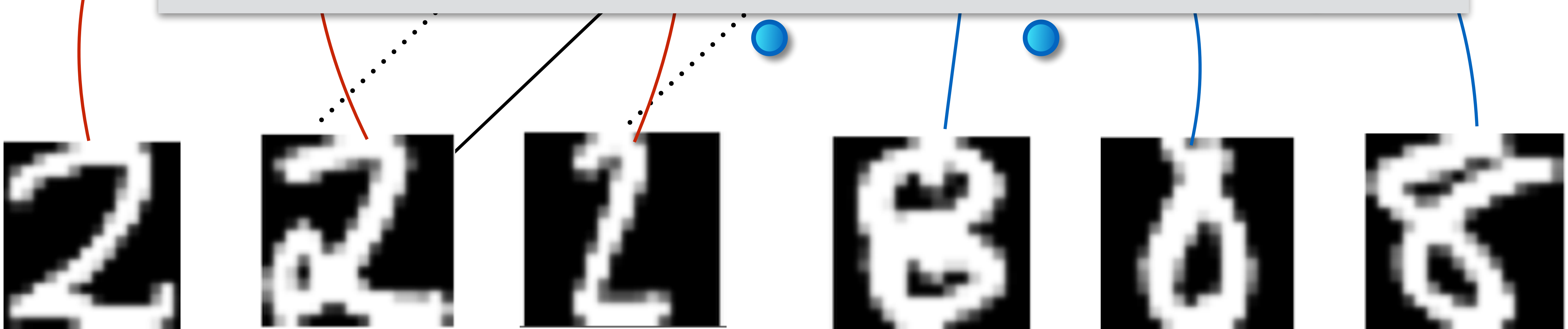
$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n} \ell_{\text{hinge}}(y_i \mathbf{w}^T \mathbf{x}_i)$$



# Example: SVM Classification

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum^{n} \ell_{\text{hinge}}(y_i \mathbf{w}^T \mathbf{x}_i)$$

Descent algorithms and line search methods
Acceleration, momentum, and conjugate gradients
Newton and Quasi-Newton methods
Coordinate descent
Stochastic and incremental gradient methods
*SMO*
*SVM$^{light}$*
*LIBLINEAR*



# Linear Regularized Loss Minimization

# Linear Regularized Loss Minimization

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n} \ell_i(\mathbf{w}^T\mathbf{x}_i)$$

# Linear Regularized Loss Minimization

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n} \ell_i(\mathbf{w}^T\mathbf{x}_i)$$

- support vector machines

# Linear Regularized Loss Minimization

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n} \ell_i(\mathbf{w}^T\mathbf{x}_i)$$

- support vector machines
- logistic regression

# Linear Regularized Loss Minimization

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n} \ell_i(\mathbf{w}^T\mathbf{x}_i)$$

- support vector machines
- logistic regression
- lasso regression

# Linear Regularized Loss Minimization

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n} \ell_i(\mathbf{w}^T\mathbf{x}_i)$$

- support vector machines
- logistic regression
- lasso regression
- ridge regression

# Linear Regularized Loss Minimization

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n}\ell_i(\mathbf{w}^T\mathbf{x}_i)$$

- support vector machines
- logistic regression
- lasso regression
- ridge regression
- etc…

# Linear Regularized Loss Minimization

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n}\ell_i(\mathbf{w}^T\mathbf{x}_i)$$

- support vector machines
- logistic regression
- lasso regression
- ridge regression
- etc…

image/music/video tagging
document categorization
item recommendation
click-through rate prediction
sequence tagging
protein structure prediction
sensor data prediction
spam classification
fraud detection

# Machine Learning Workflow

DATA & PROBLEM

*classification, regression, collaborative filtering, …*

MACHINE LEARNING MODEL

*logistic regression, lasso, support vector machines, …*

OPTIMIZATION ALGORITHM

*gradient descent, coordinate descent, Newton's method, …*

# Machine Learning Workflow

| | |
|---|---|
| DATA & PROBLEM | *classification, regression, collaborative filtering, …* |
| MACHINE LEARNING MODEL | *logistic regression, lasso, support vector machines, …* |
| SYSTEMS SETTING | *multi-core, cluster, cloud, supercomputer, …* |
| OPTIMIZATION ALGORITHM | *gradient descent, coordinate descent, Newton’s method, …* |

# Machine Learning Workflow

classification, regression

*Open Problem:*

efficiently solving objective when data is distributed

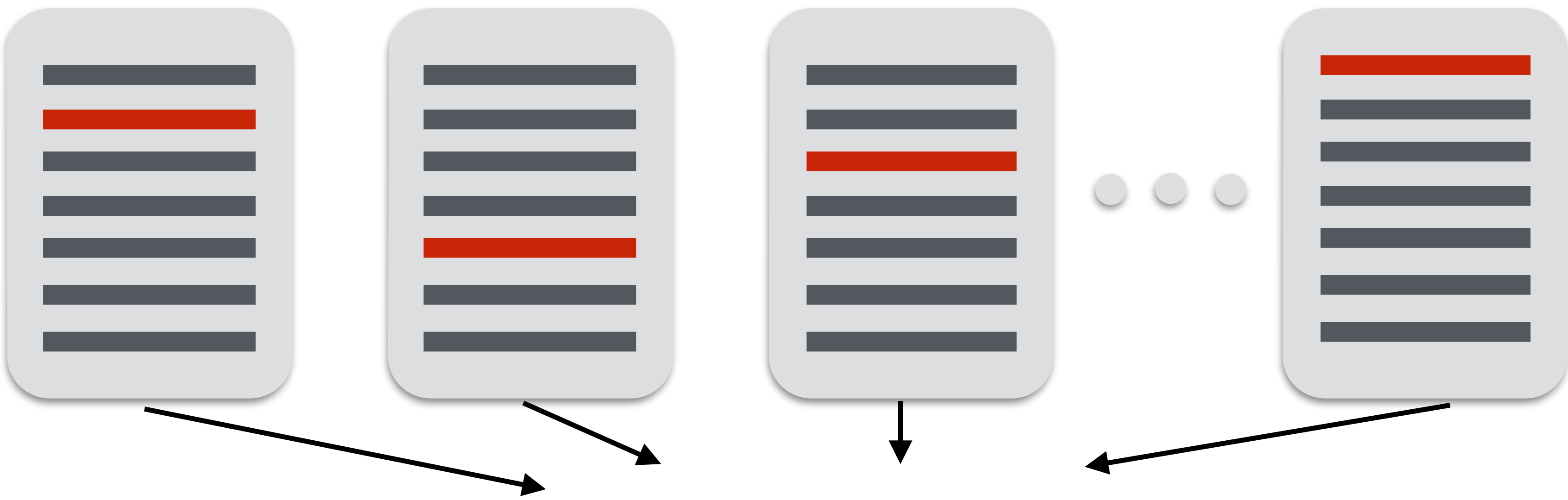
OPTIMIZATION ALGORITHM

*gradient descent, coordinate descent, Newton's method, …*

# Distributed Optimization

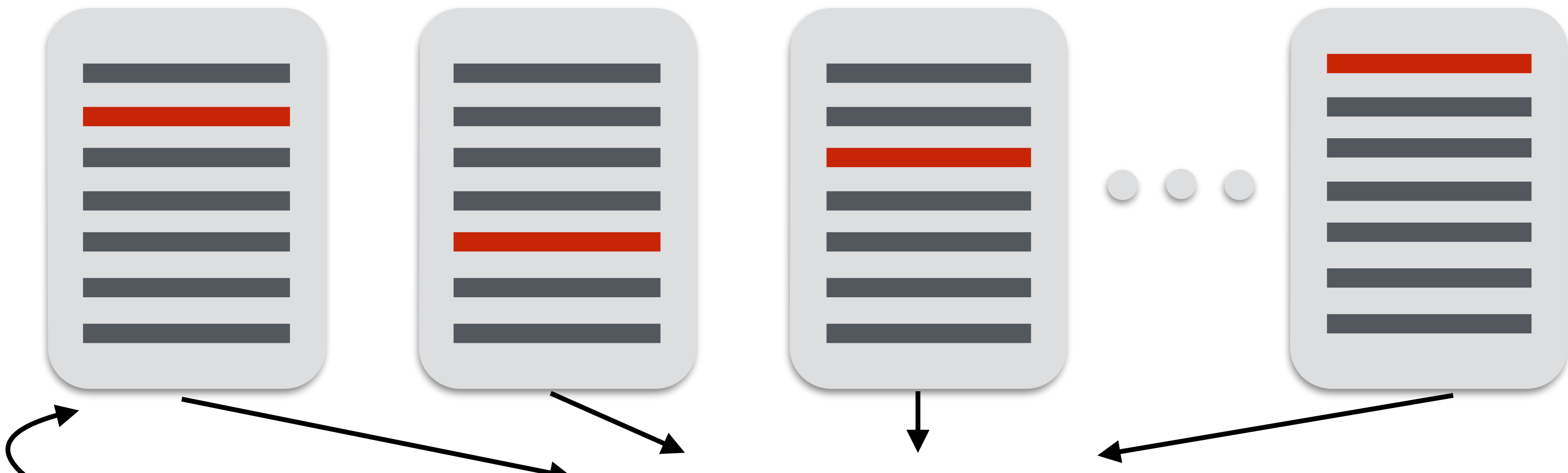
# Distributed Optimization

# Distributed Optimization



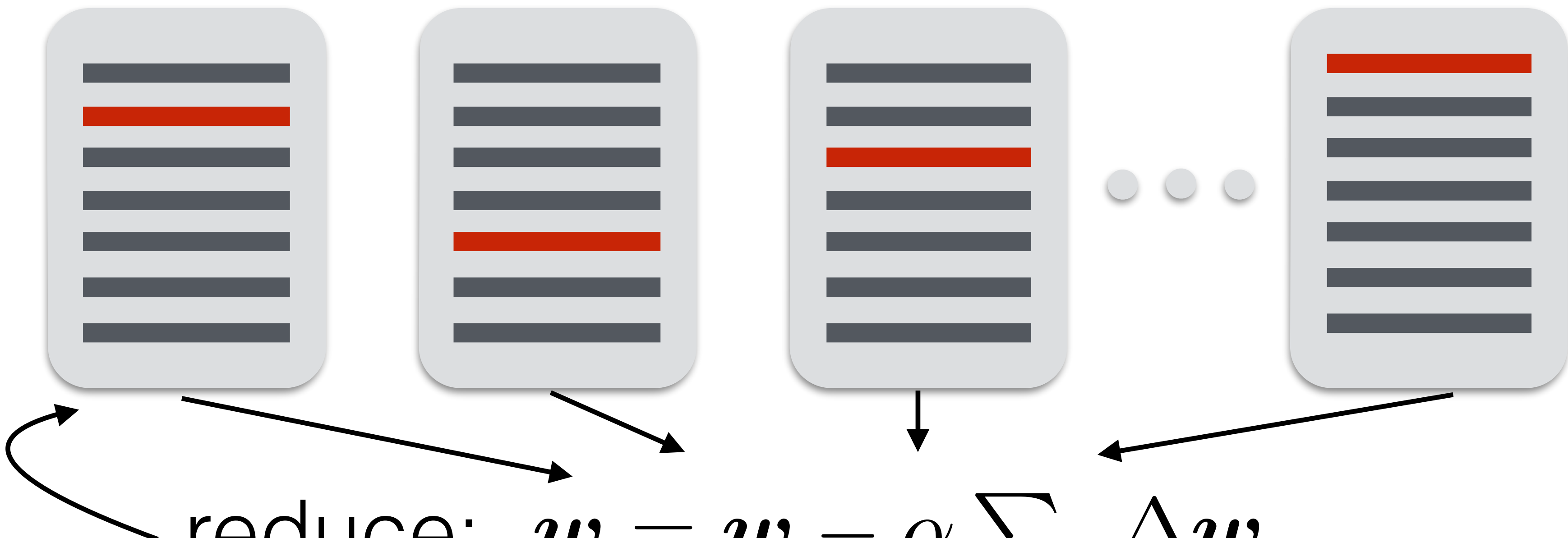
reduce: $\boldsymbol{w} = \boldsymbol{w} - \alpha \sum_k \Delta \boldsymbol{w}$

# Distributed Optimization



reduce: $\boldsymbol{w} = \boldsymbol{w} - \alpha \sum_k \Delta \boldsymbol{w}$

# Distributed Optimization



*"always communicate"*

reduce: $\boldsymbol{w} = \boldsymbol{w} - \alpha \sum_k \Delta \boldsymbol{w}$

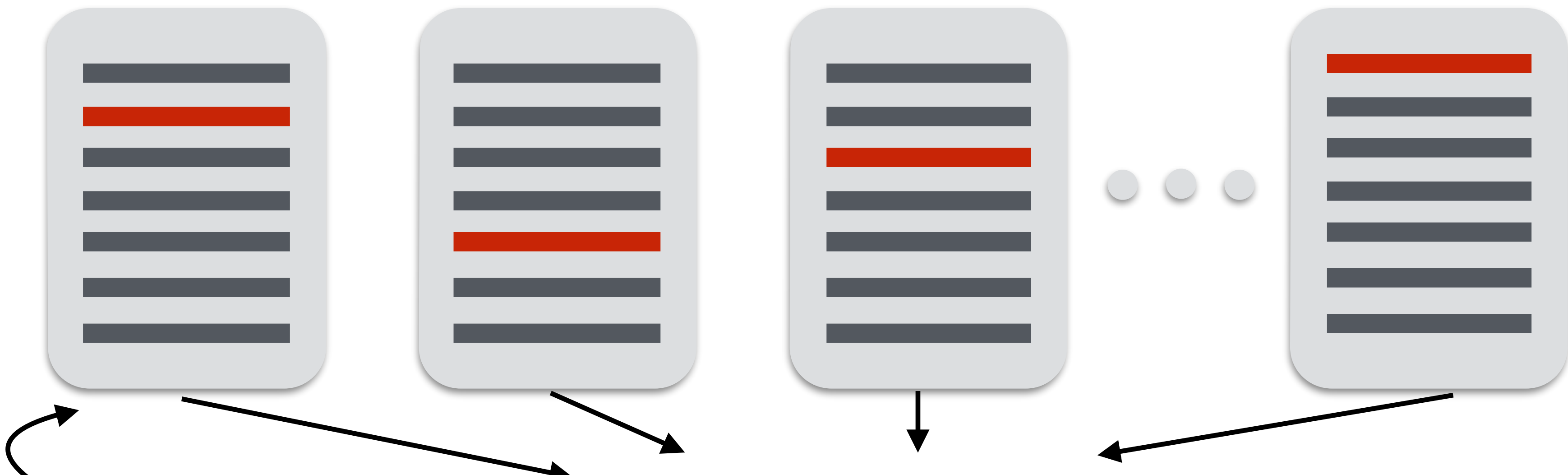
# Distributed Optimization

reduce: $\boldsymbol{w} = \boldsymbol{w} - \alpha \sum_k \Delta \boldsymbol{w}$

✔ convergence guarantees

*"always communicate"*

# Distributed Optimization

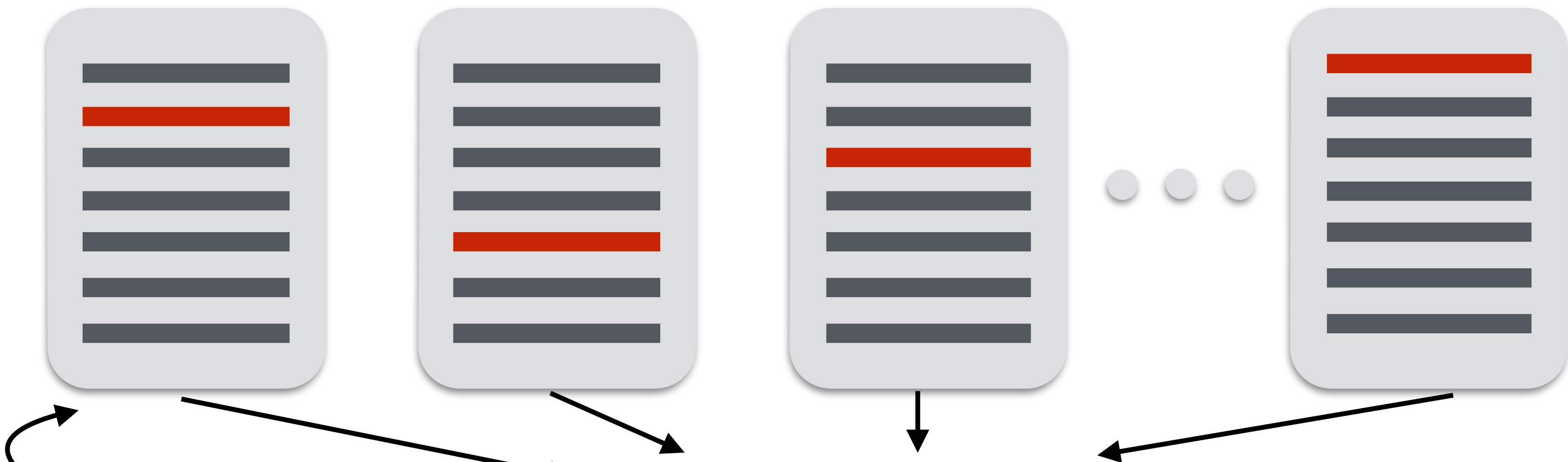


reduce: $\boldsymbol{w} = \boldsymbol{w} - \alpha \sum_k \Delta \boldsymbol{w}$

✔ convergence guarantees
✗ high communication

*"always communicate"*

# Distributed Optimization

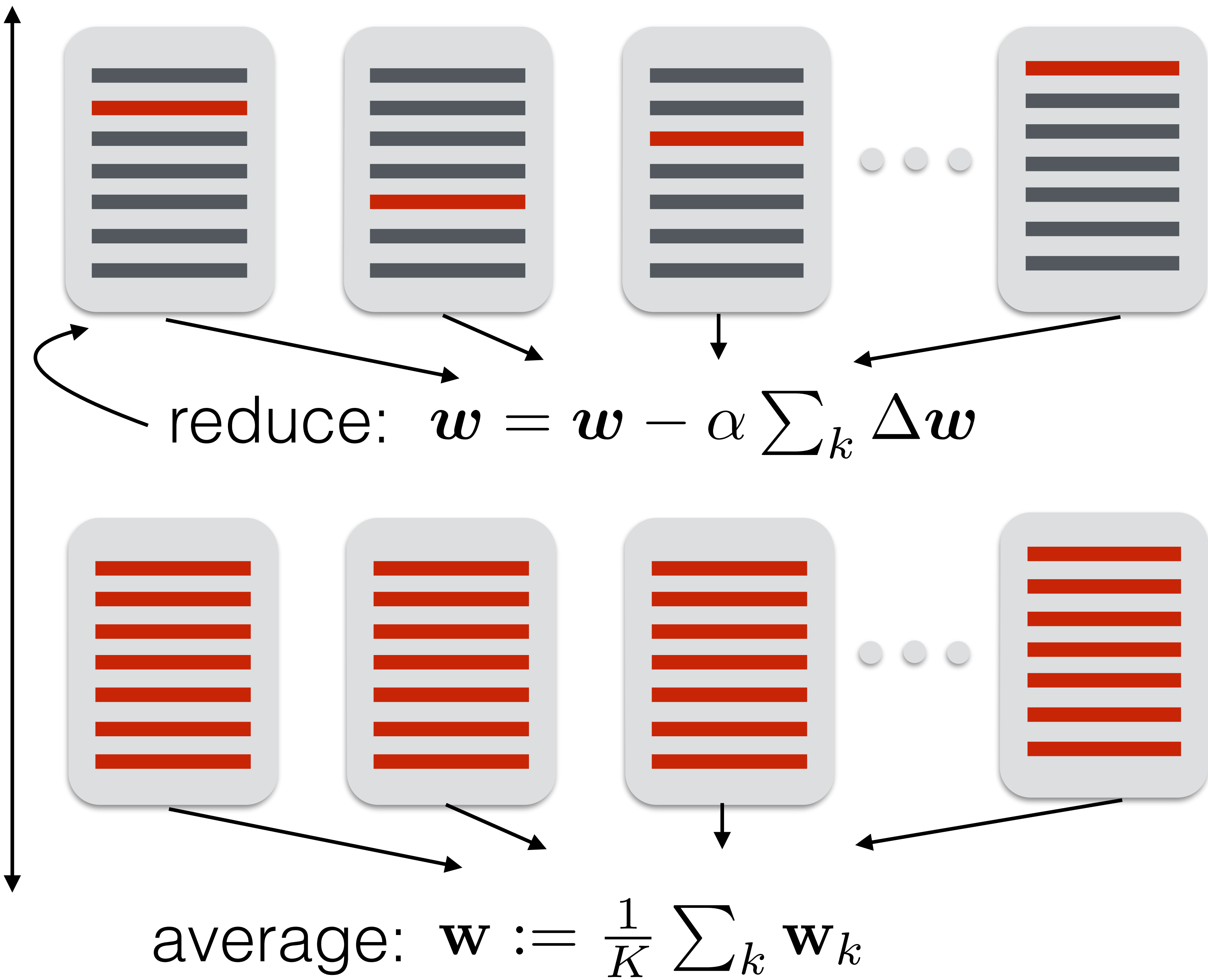


reduce: $\boldsymbol{w} = \boldsymbol{w} - \alpha \sum_k \Delta \boldsymbol{w}$

✔ convergence guarantees
✗ high communication

*“always communicate”*

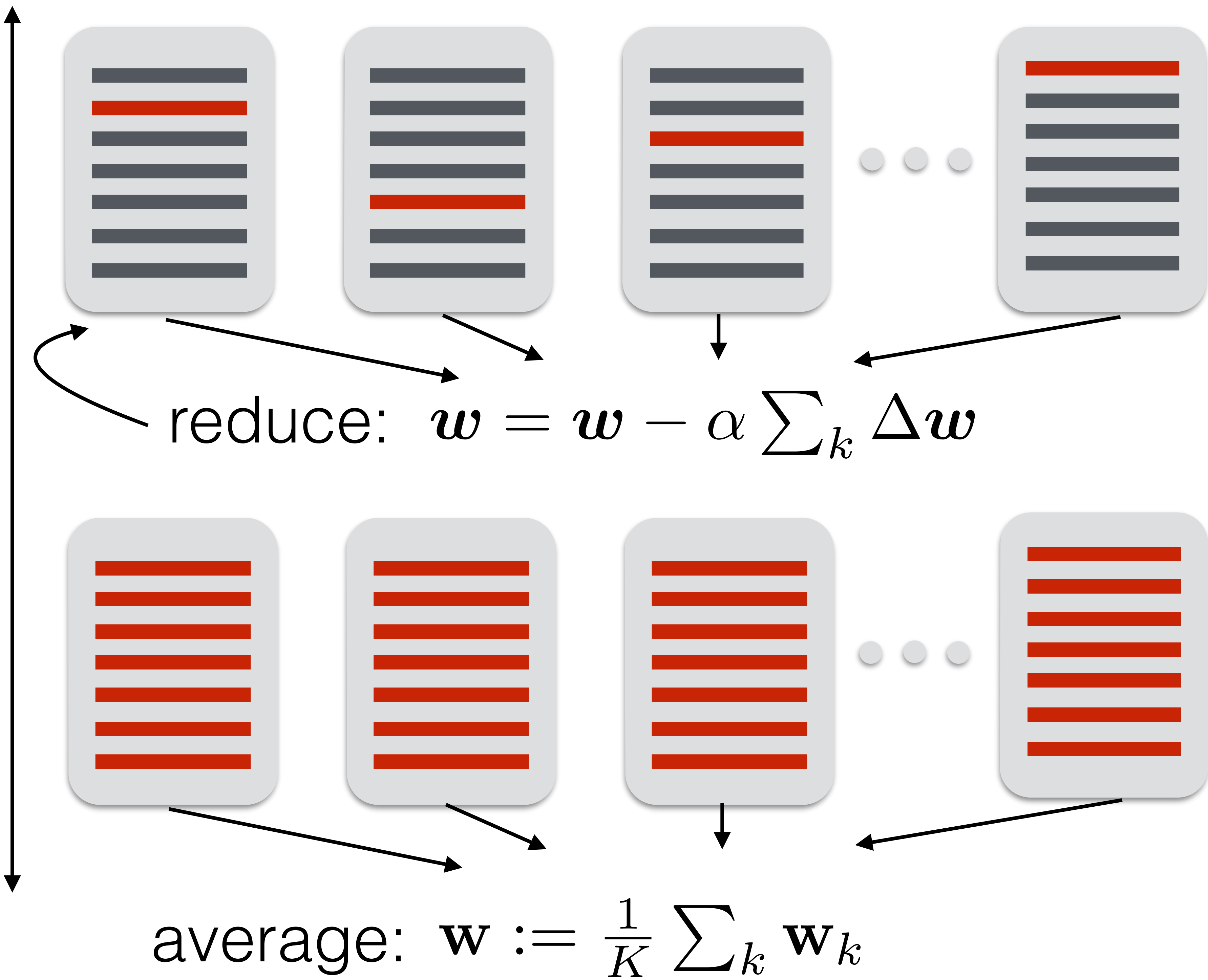
# Distributed Optimization




✔ convergence guarantees
✗ high communication

*"always communicate"*

# Distributed Optimization

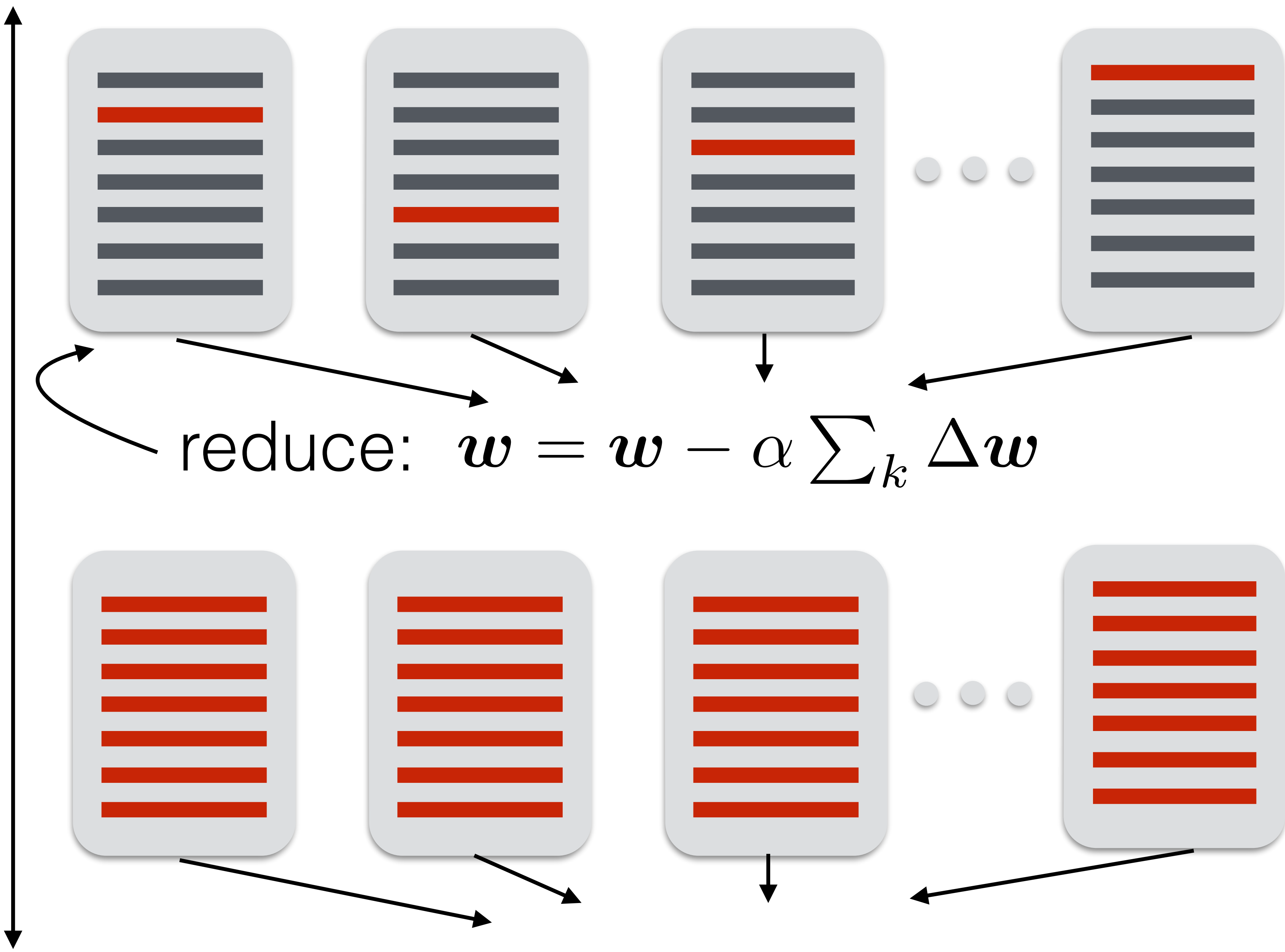



✔ convergence guarantees
✗ high communication

*"always communicate"*

*"never communicate"*

# Distributed Optimization




✔ convergence guarantees
✗ high communication

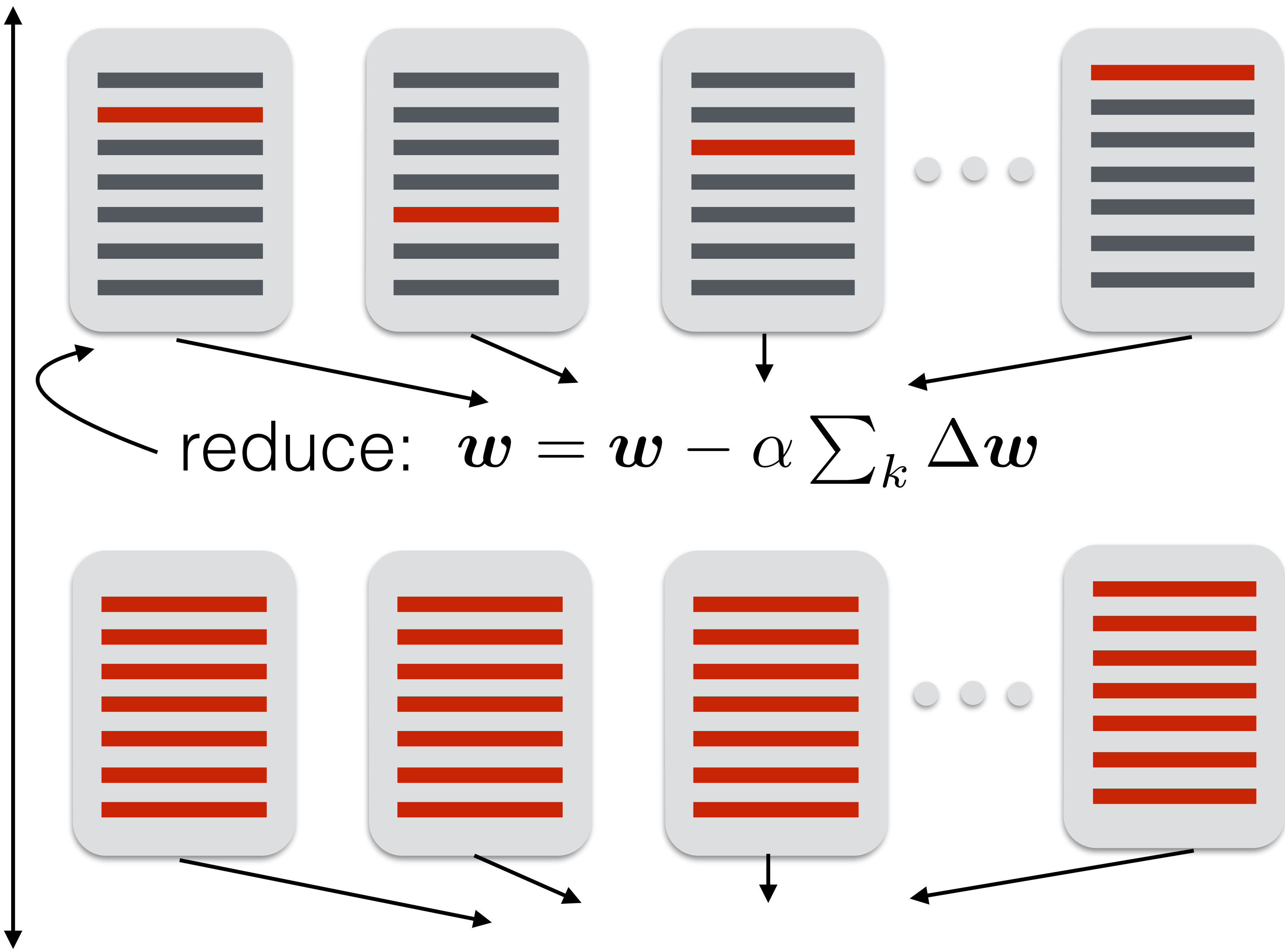
*"always communicate"*

✔ low communication

*"never communicate"*

average: $\mathbf{w} := \frac{1}{K} \sum_k \mathbf{w}_k$

# Distributed Optimization



reduce: $\boldsymbol{w} = \boldsymbol{w} - \alpha \sum_k \Delta \boldsymbol{w}$

average: $\mathbf{w} := \frac{1}{K} \sum_k \mathbf{w}_k$

✔ convergence guarantees
✗ high communication

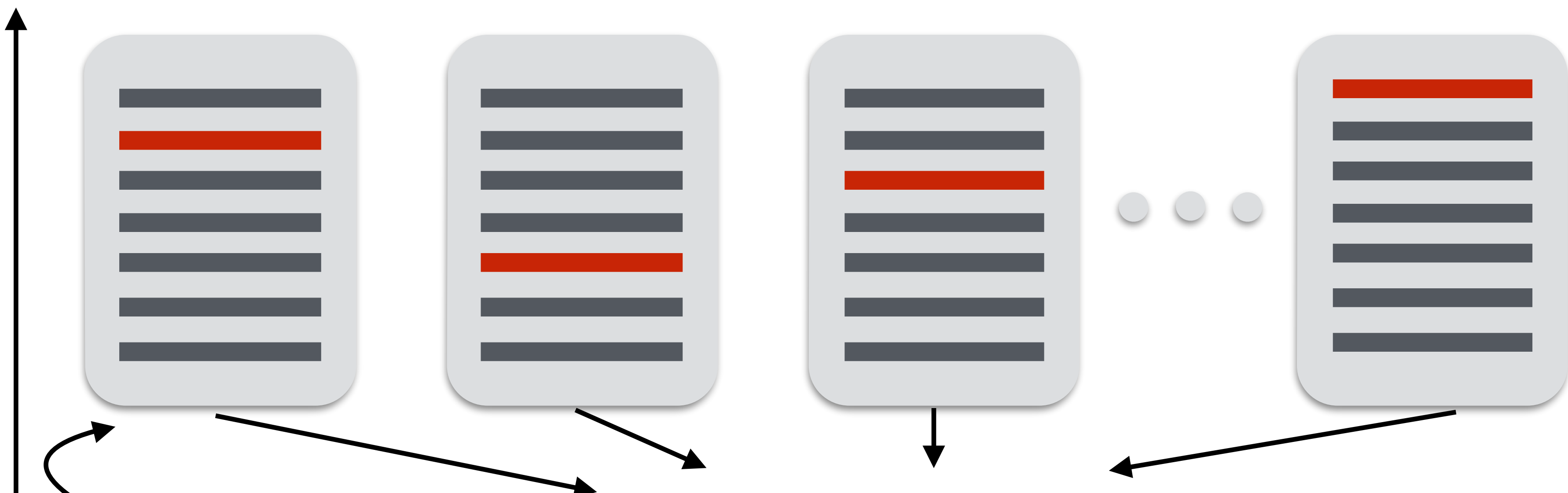
*"always communicate"*

✔ low communication

*"never communicate"*

ZDWJ, 2012

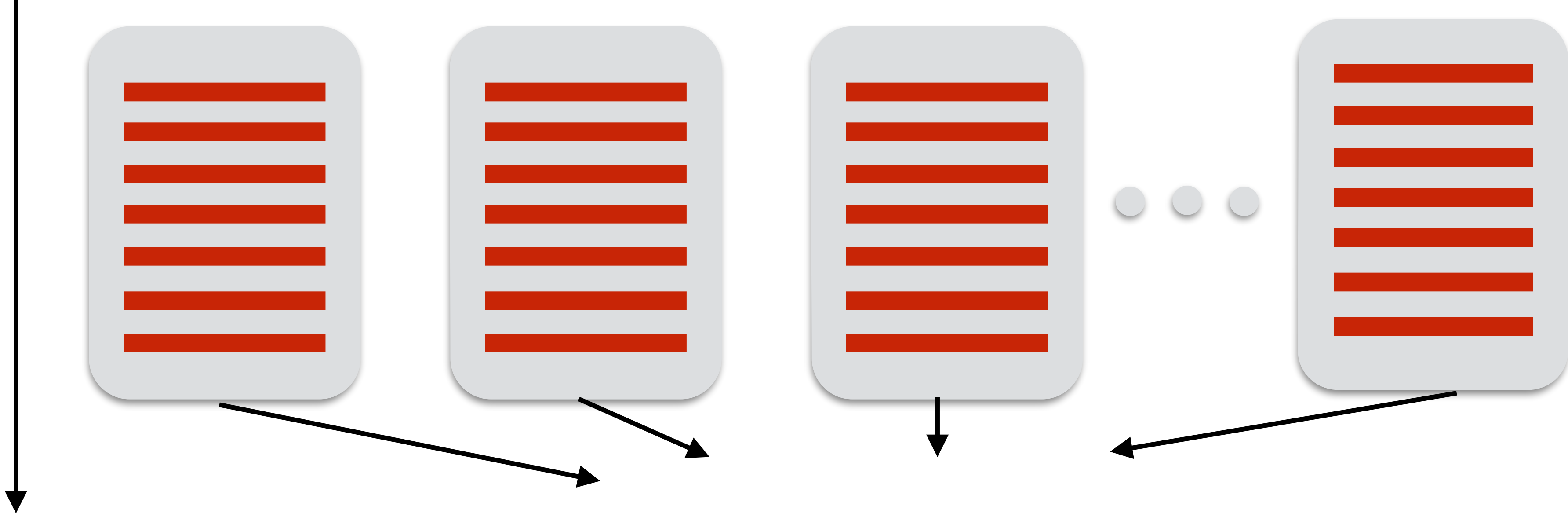
# Distributed Optimization



reduce: $\boldsymbol{w} = \boldsymbol{w} - \alpha \sum_k \Delta \boldsymbol{w}$

✔ convergence guarantees
✗ high communication

*"always communicate"*



average: $\mathbf{w} := \frac{1}{K} \sum_k \mathbf{w}_k$

✔ low communication
✗ convergence not guaranteed

*"never communicate"*

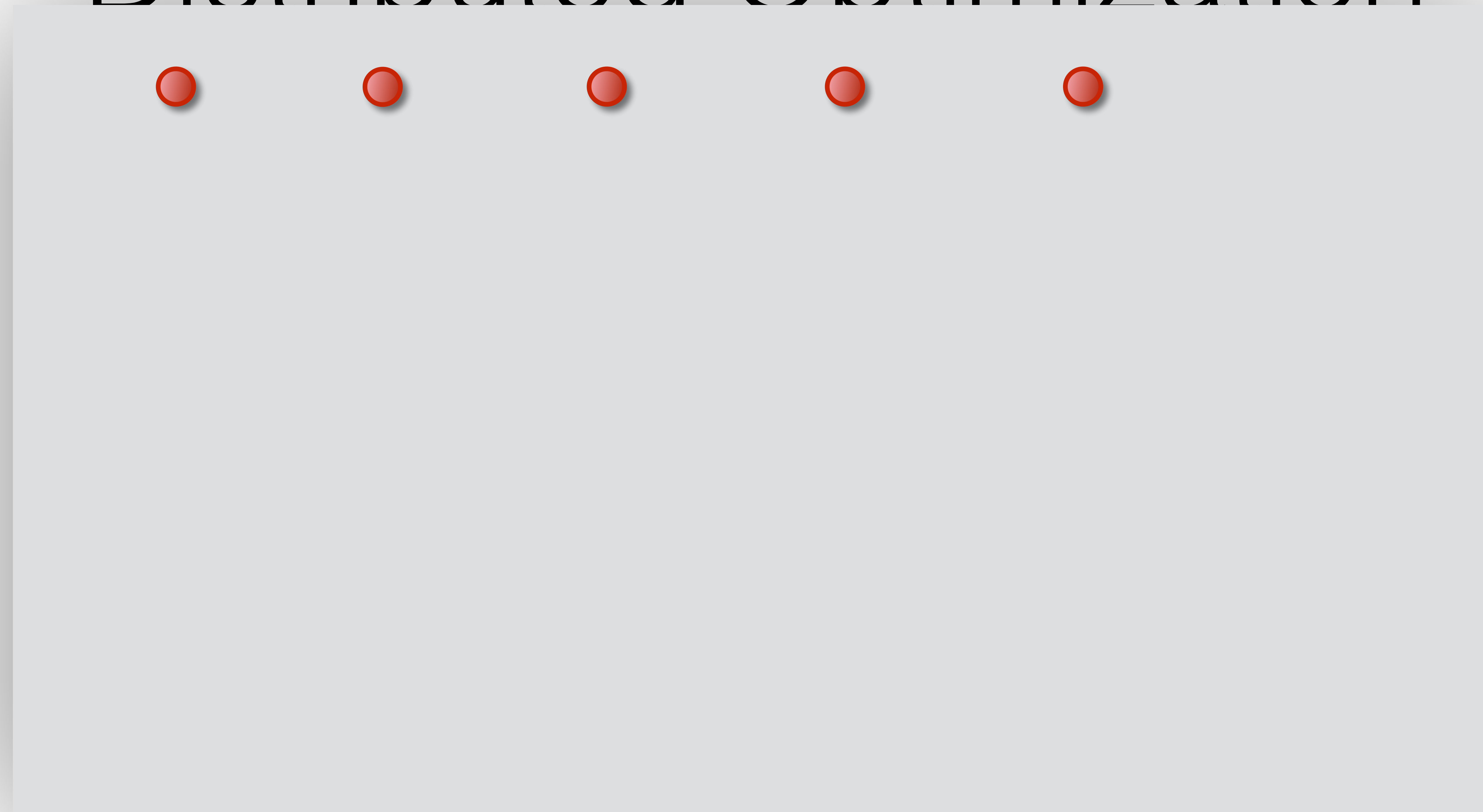
ZDWJ, 2012

# Distributed Optimization

average: $\mathbf{w} := \frac{1}{K} \sum_k \mathbf{w}_k$

ZDWJ, 2012

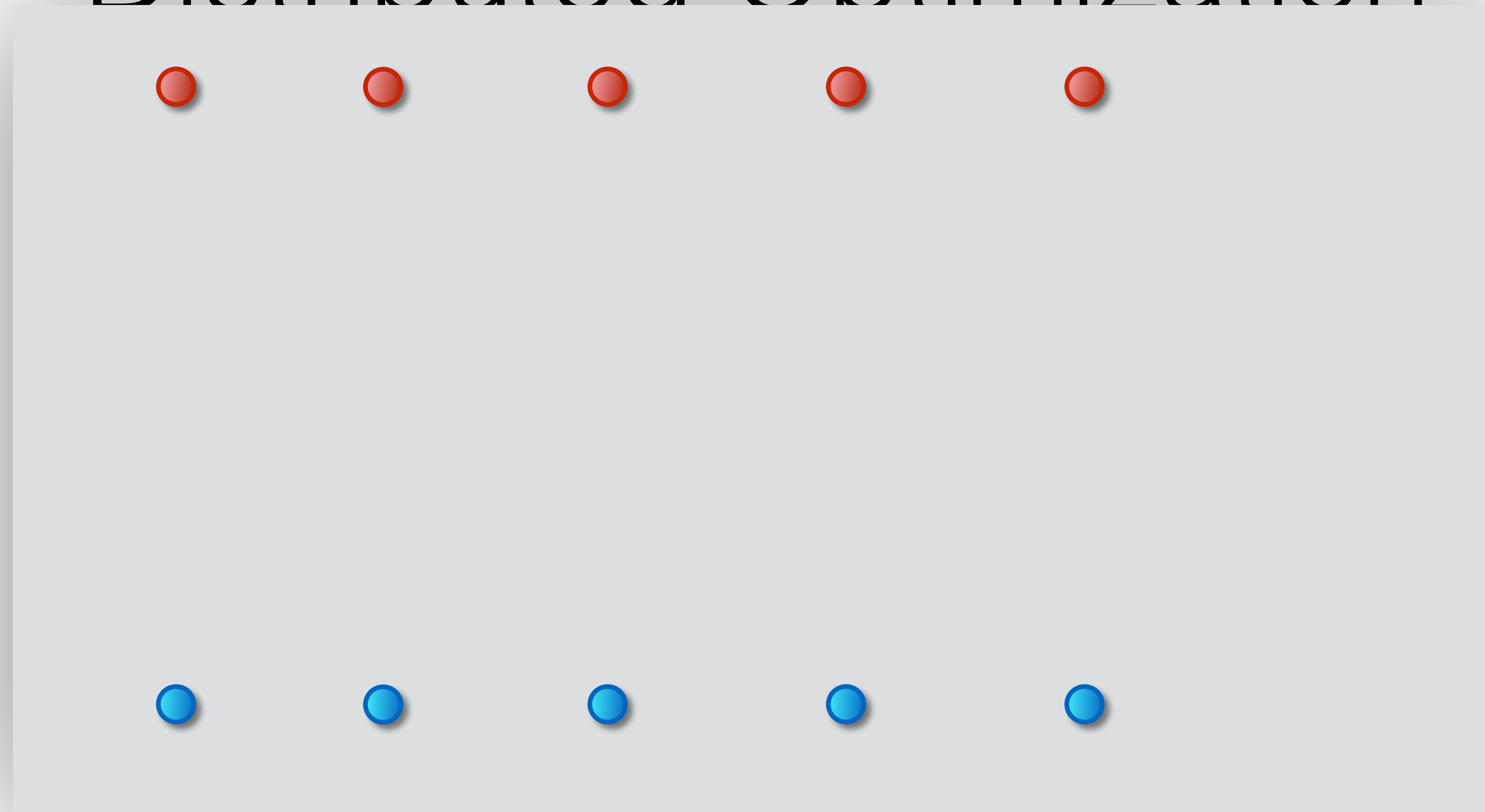
# Distributed Optimization



average: $\mathbf{w} := \frac{1}{K}\sum_k \mathbf{w}_k$

ZDWJ, 2012

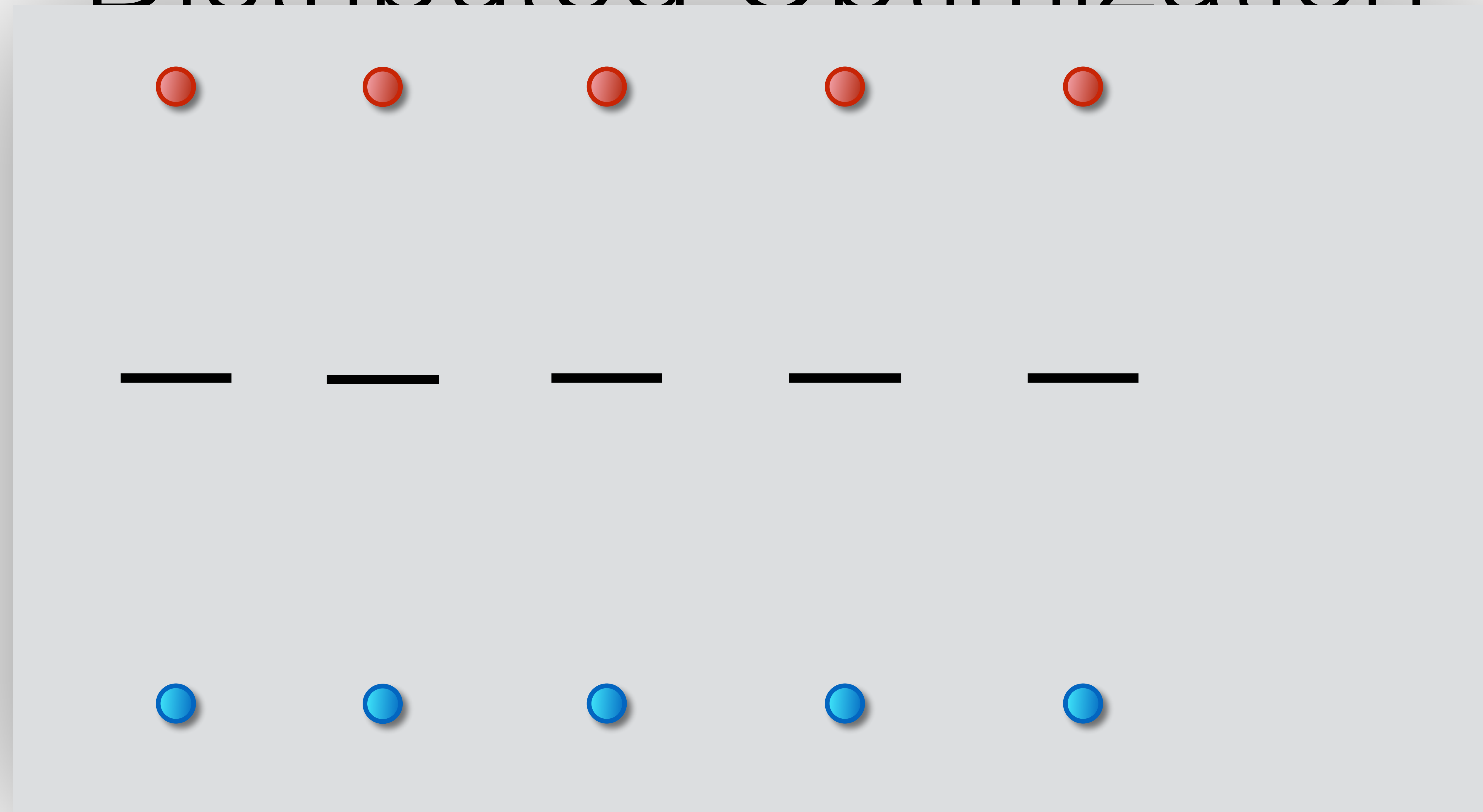
# Distributed Optimization



average: $\mathbf{w} := \frac{1}{K} \sum_k \mathbf{w}_k$

ZDWJ, 2012

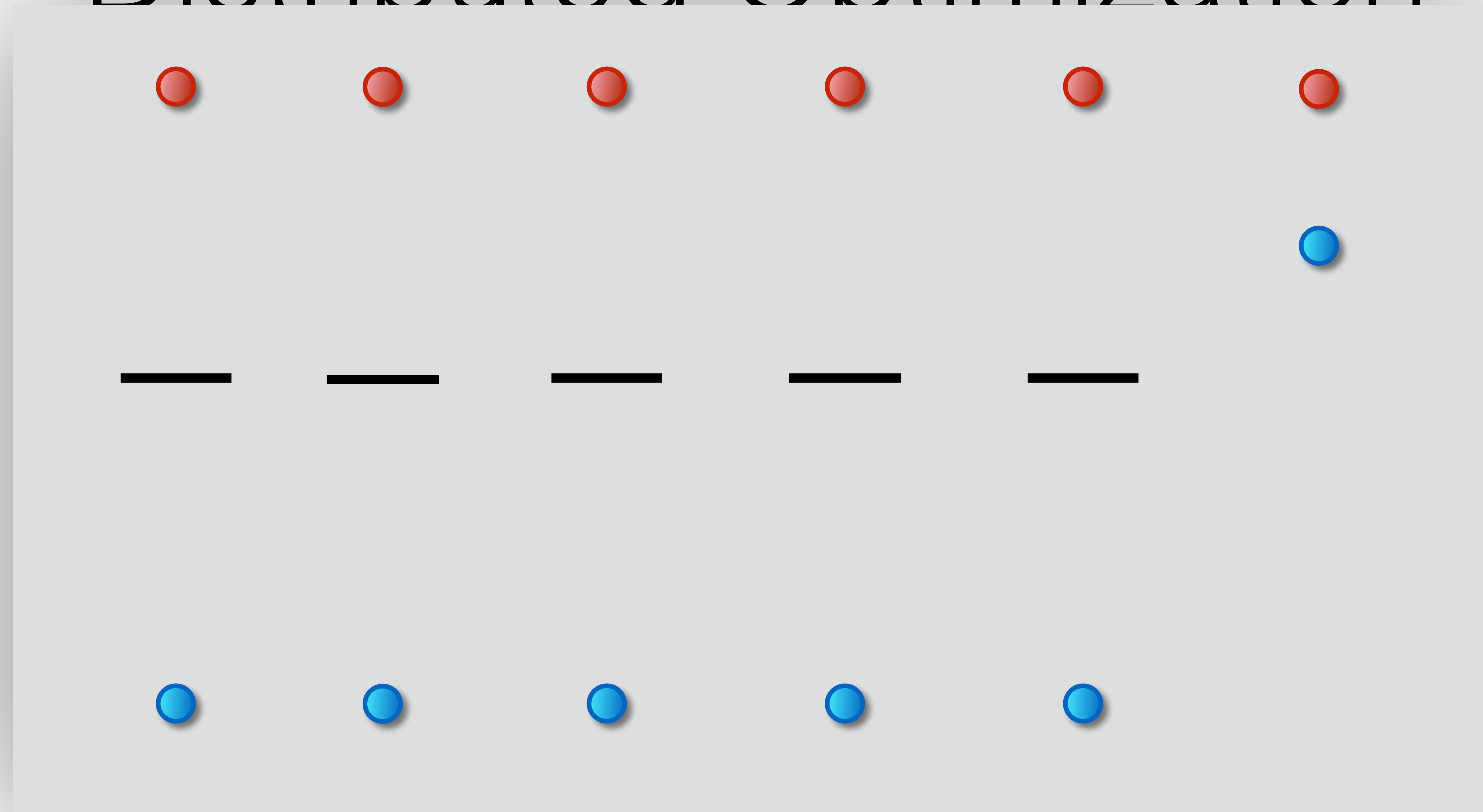
# Distributed Optimization



average: $\mathbf{w} := \frac{1}{K} \sum_k \mathbf{w}_k$

ZDWJ, 2012

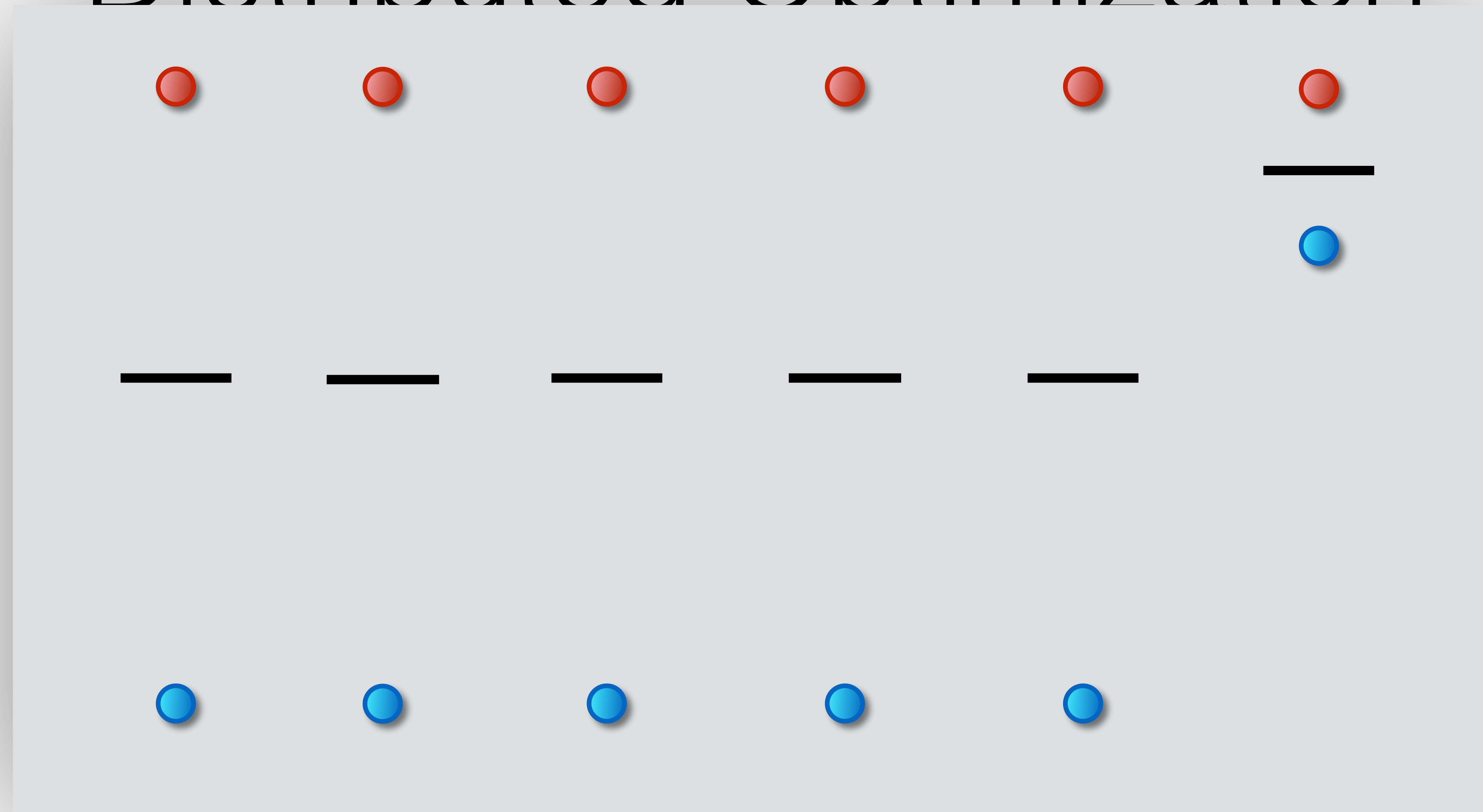
# Distributed Optimization



average: $\mathbf{w} := \frac{1}{K} \sum_k \mathbf{w}_k$

ZDWJ, 2012

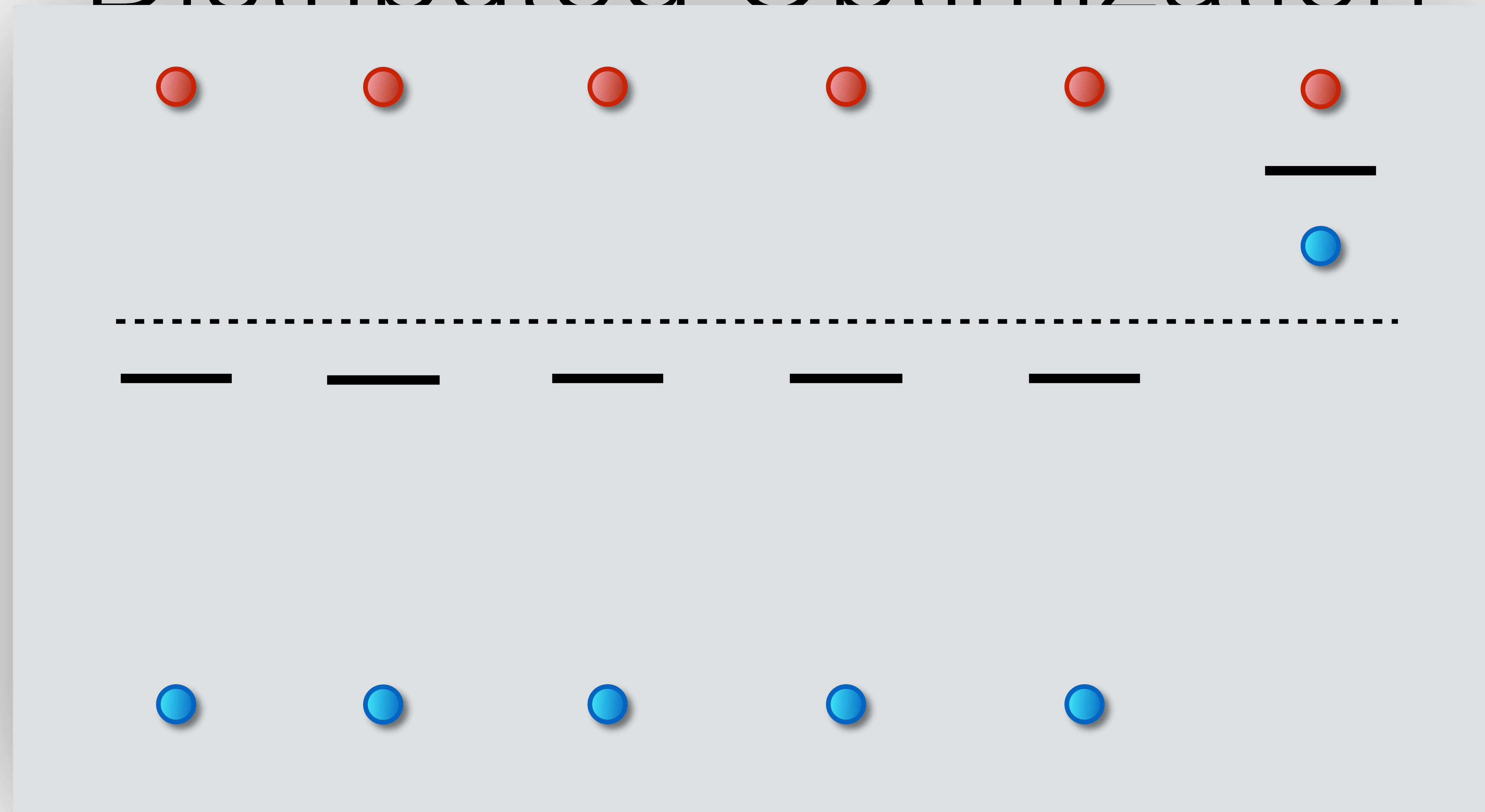
# Distributed Optimization



average: $\mathbf{w} := \frac{1}{K} \sum_k \mathbf{w}_k$

ZDWJ, 2012

# Distributed Optimization



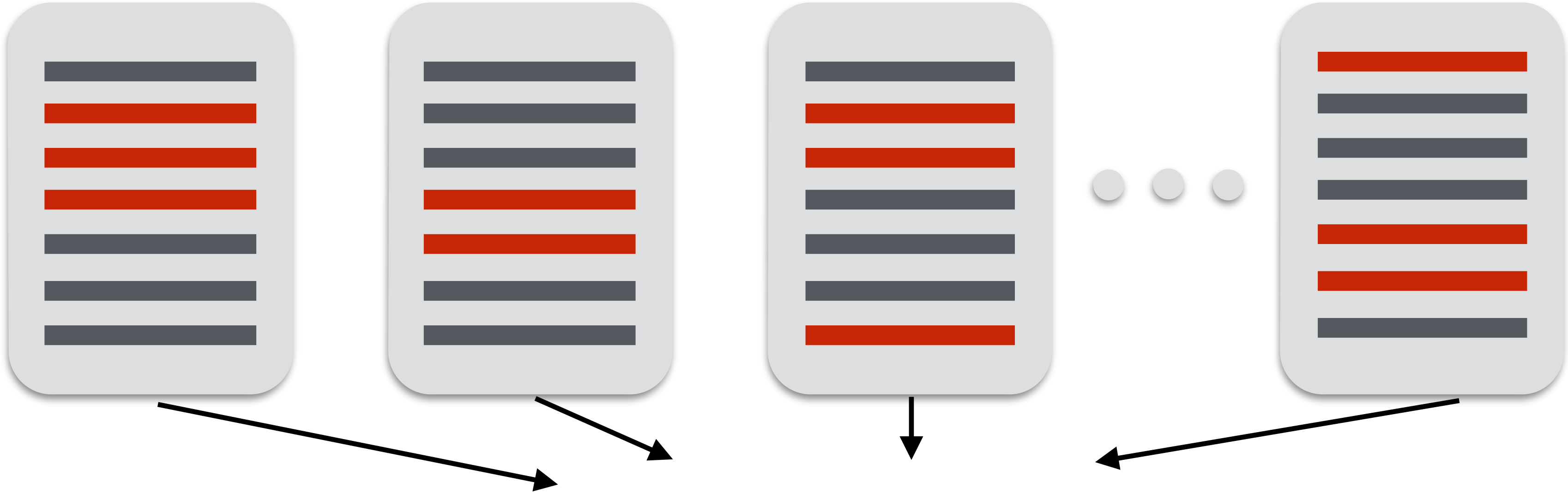
average: $\mathbf{w} := \frac{1}{K}\sum_k \mathbf{w}_k$

ZDWJ, 2012

# Mini-batch

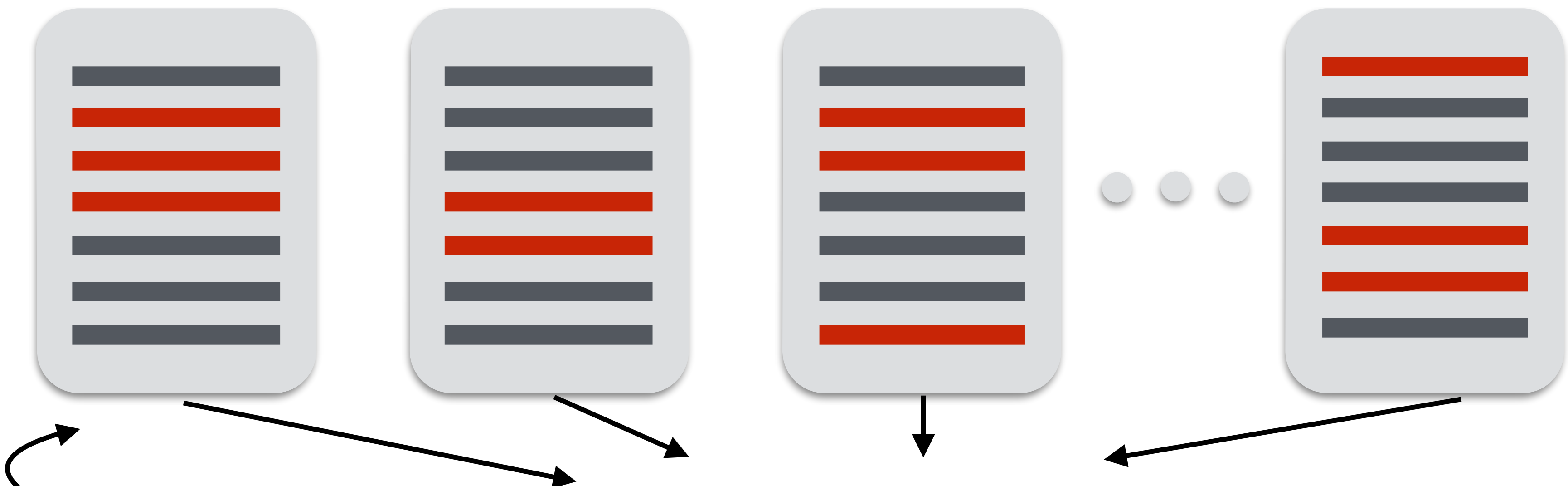
# Mini-batch

# Mini-batch



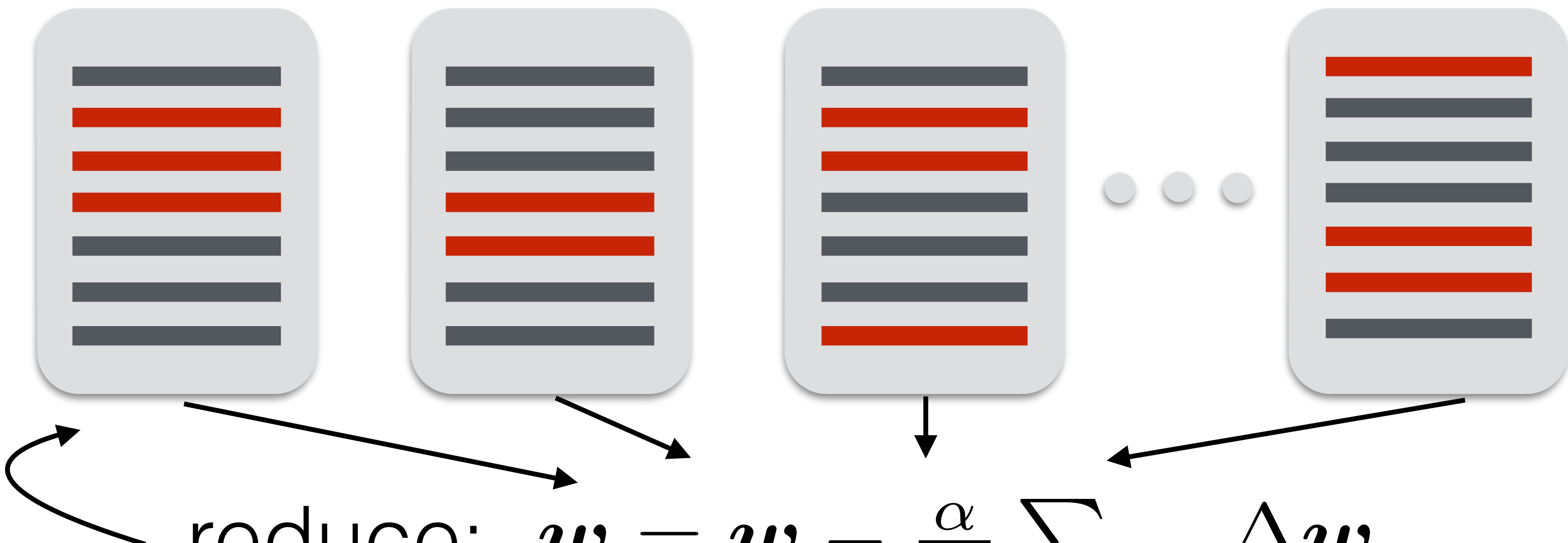
reduce: $\boldsymbol{w} = \boldsymbol{w} - \frac{\alpha}{|b|} \sum_{i \in b} \Delta \boldsymbol{w}$

# Mini-batch



reduce: $\boldsymbol{w} = \boldsymbol{w} - \frac{\alpha}{|b|} \sum_{i \in b} \Delta \boldsymbol{w}$

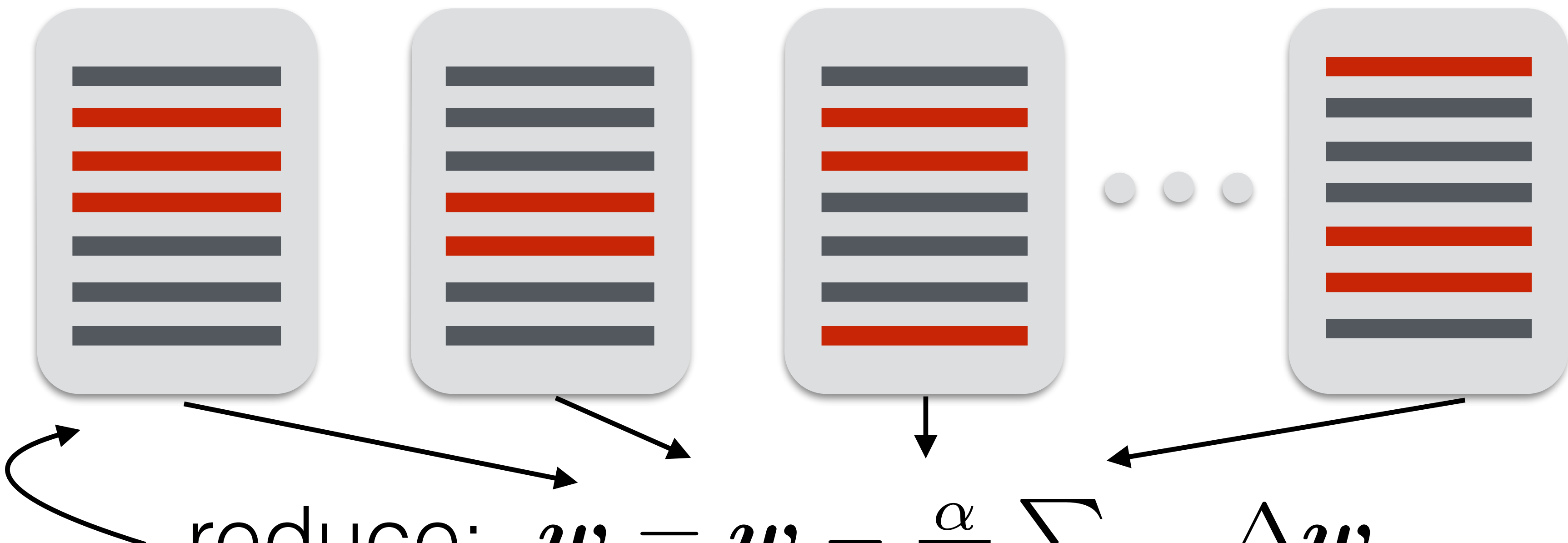
# Mini-batch



✔ convergence guarantees

reduce: $\boldsymbol{w} = \boldsymbol{w} - \frac{\alpha}{|b|} \sum_{i \in b} \Delta \boldsymbol{w}$

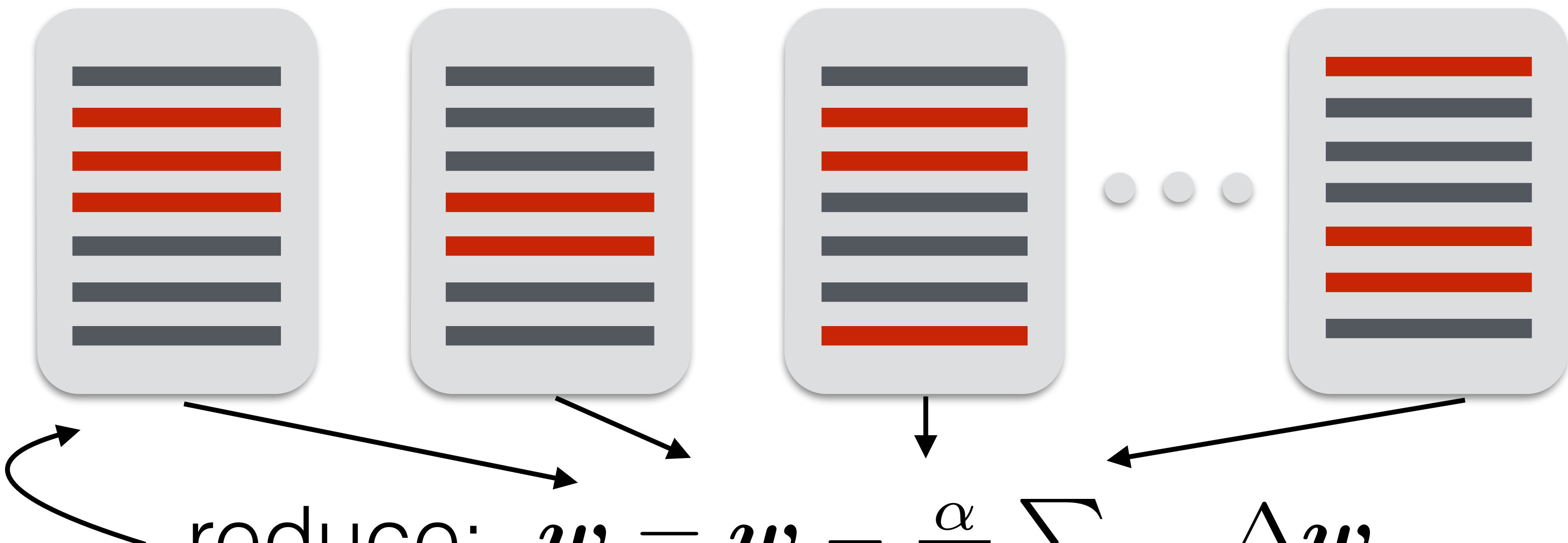
# Mini-batch



reduce: $\boldsymbol{w} = \boldsymbol{w} - \frac{\alpha}{|b|} \sum_{i \in b} \Delta \boldsymbol{w}$

✔ convergence guarantees
✔ tunable communication

# Mini-batch

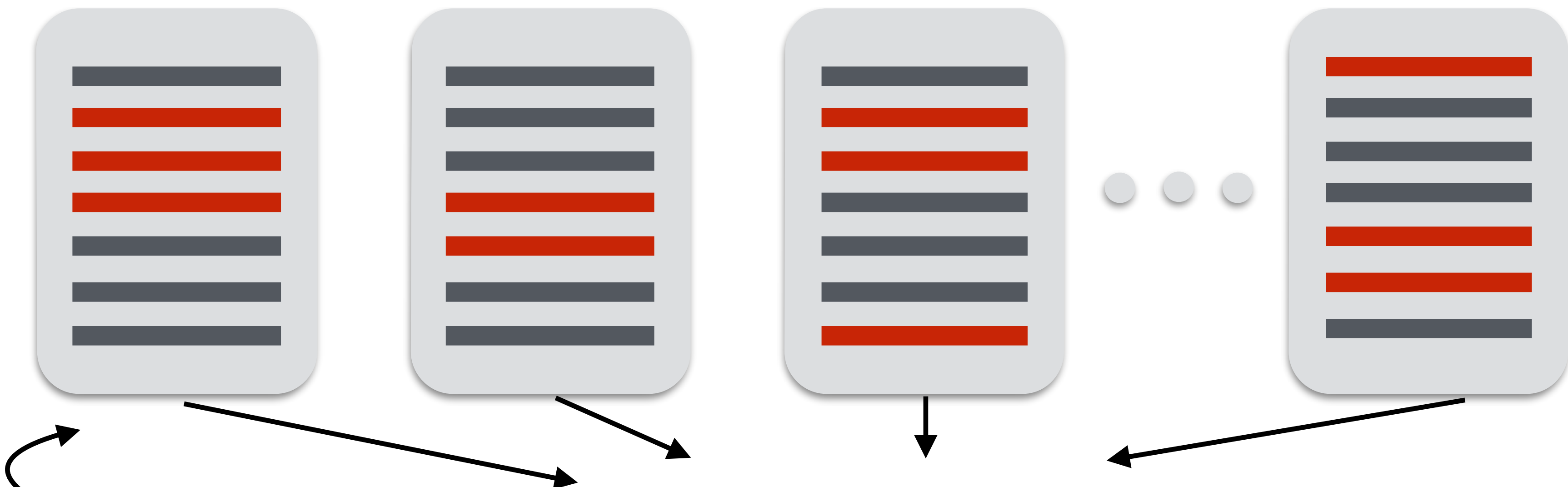


reduce: $\boldsymbol{w} = \boldsymbol{w} - \frac{\alpha}{|b|} \sum_{i \in b} \Delta \boldsymbol{w}$

✔ convergence guarantees
✔ tunable communication

- a natural middle-ground

# Mini-batch



✔ convergence guarantees
✔ tunable communication

reduce: $\boldsymbol{w} = \boldsymbol{w} - \frac{\alpha}{|b|} \sum_{i \in b} \Delta \boldsymbol{w}$

- a natural middle-ground

# Mini-batch Limitations

# Mini-batch Limitations

1. METHODS BEYOND SGD

# Mini-batch Limitations

1. METHODS BEYOND SGD
2. STALE UPDATES

# Mini-batch Limitations

1. METHODS BEYOND SGD
2. STALE UPDATES
3. AVERAGE OVER BATCH SIZE

LARGE-SCALE OPTIMIZATION

COCOA

RESULTS

LARGE-SCALE OPTIMIZATION

**COCOA**

RESULTS

# Mini-batch Limitations

1. METHODS BEYOND SGD
2. STALE UPDATES
3. AVERAGE OVER BATCH SIZE

# Mini-batch Limitations

1. ~~METHODS BEYOND SGD~~

   *Use Primal-Dual Framework*

2. ~~STALE UPDATES~~

   *Immediately apply local updates*

3. ~~AVERAGE OVER BATCH SIZE~~

   *Average over K << batch size*

# *Co*mmunication-Efficient Distributed Dual *Co*ordinate *A*scent (CoCoA)

1. ~~METHODS BEYOND SGD~~

   *Use Primal-Dual Framework*

2. ~~STALE UPDATES~~

   *Immediately apply local updates*

3. ~~AVERAGE OVER BATCH SIZE~~

   *Average over K << batch size*

# 1. Primal-Dual Framework

PRIMAL $\geq$ DUAL

# 1. Primal-Dual Framework

PRIMAL $\geq$ DUAL

$$\min_{\mathbf{w}\in\mathbb{R}^d} \left[ P(\mathbf{w}) := \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n} \ell_i(\mathbf{w}^T\mathbf{x}_i) \right]$$

# 1. Primal-Dual Framework

PRIMAL $\geq$ DUAL

$$\min_{\mathbf{w}\in\mathbb{R}^d}\left[P(\mathbf{w}) := \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n}\ell_i(\mathbf{w}^T\mathbf{x}_i)\right]$$

$$\max_{\alpha\in\mathbb{R}^n}\left[D(\alpha) := -||A\alpha||^2 - \frac{1}{n}\sum_{i=1}^{n}\ell_i^*(-\alpha_i)\right]$$

$$A_i = \frac{1}{\lambda n}x_i$$

# 1. Primal-Dual Framework

PRIMAL $\geq$ DUAL

$$\min_{\mathbf{w}\in\mathbb{R}^d}\left[P(\mathbf{w}) := \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n}\ell_i(\mathbf{w}^T\mathbf{x}_i)\right]$$

$$\max_{\alpha\in\mathbb{R}^n}\left[D(\alpha) := -||A\alpha||^2 - \frac{1}{n}\sum_{i=1}^{n}\ell_i^*(-\alpha_i)\right]$$

$$A_i = \frac{1}{\lambda n}x_i$$

- Stopping criteria given by duality gap

# 1. Primal-Dual Framework

PRIMAL $\geq$ DUAL

$$\min_{\mathbf{w}\in\mathbb{R}^d}\left[P(\mathbf{w}) := \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n}\ell_i(\mathbf{w}^T\mathbf{x}_i)\right]$$

$$\max_{\alpha\in\mathbb{R}^n}\left[D(\alpha) := -||A\alpha||^2 - \frac{1}{n}\sum_{i=1}^{n}\ell_i^*(-\alpha_i)\right]$$

$$A_i = \frac{1}{\lambda n}x_i$$

- Stopping criteria given by duality gap
- Good performance in practice

# 1. Primal-Dual Framework

PRIMAL $\geq$ DUAL

$$\min_{\mathbf{w}\in\mathbb{R}^d} \left[ P(\mathbf{w}) := \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n} \ell_i(\mathbf{w}^T\mathbf{x}_i) \right]$$

$$\max_{\alpha\in\mathbb{R}^n} \left[ D(\alpha) := -||A\alpha||^2 - \frac{1}{n}\sum_{i=1}^{n} \ell_i^*(-\alpha_i) \right]$$

$$A_i = \frac{1}{\lambda n} x_i$$

- Stopping criteria given by duality gap
- Good performance in practice
- Default in software packages e.g. liblinear

# 2. Immediately Apply Updates

# 2. Immediately Apply Updates

STALE

$$
\begin{aligned}
&\textbf{for } i \in b \\
&\quad | \quad \Delta \boldsymbol{w} \leftarrow \Delta \boldsymbol{w} - \alpha \nabla_i P(\boldsymbol{w}) \\
&\textbf{end} \\
&\boldsymbol{w} \leftarrow \boldsymbol{w} + \Delta \boldsymbol{w}
\end{aligned}
$$

# 2. Immediately Apply Updates

STALE

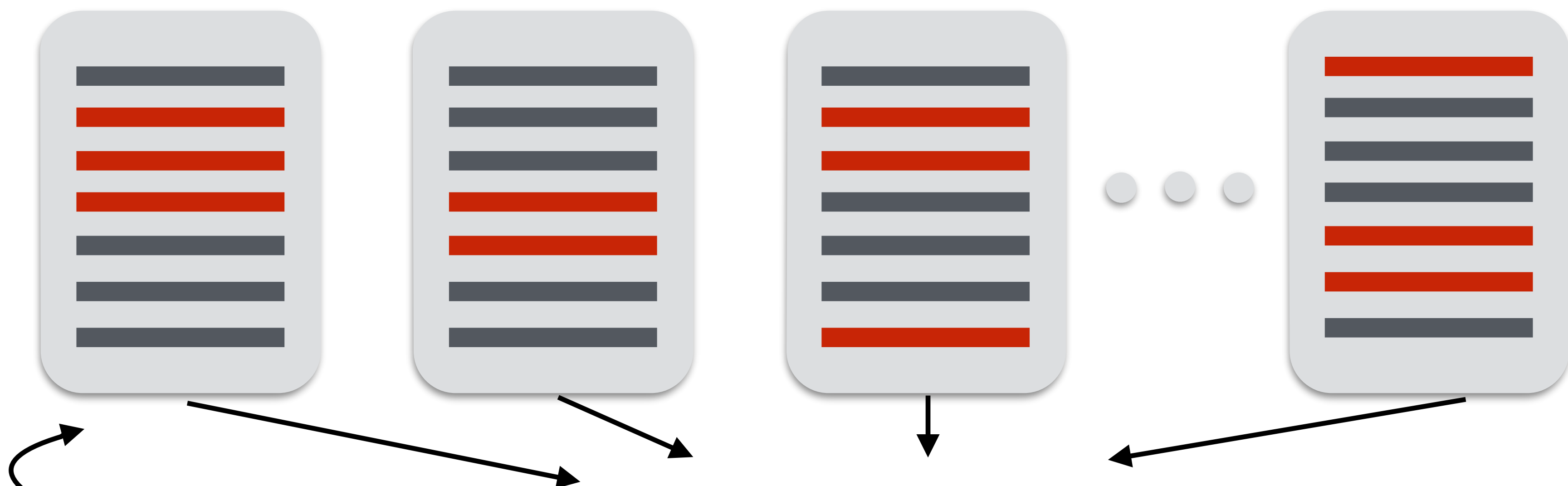
$$
\begin{aligned}
&\textbf{for } i \in b \\
&\quad \Delta\boldsymbol{w} \leftarrow \Delta\boldsymbol{w} - \alpha \nabla_i P(\boldsymbol{w}) \\
&\textbf{end} \\
&\boldsymbol{w} \leftarrow \boldsymbol{w} + \Delta\boldsymbol{w}
\end{aligned}
$$

FRESH

$$
\begin{aligned}
&\textbf{for } i \in b \\
&\quad \Delta\boldsymbol{w} \leftarrow \Delta\boldsymbol{w} - \alpha \nabla_i P(\boldsymbol{w}) \\
&\quad \boldsymbol{w} \leftarrow \boldsymbol{w} + \Delta\boldsymbol{w} \\
&\textbf{end}
\end{aligned}
$$

# 3. Average over K

# 3. Average over K



reduce: $\boldsymbol{w} = \boldsymbol{w} + \frac{1}{K}\sum_k \Delta\boldsymbol{w}_k$

# CoCoA

**Algorithm 1:** CoCoA

**Input**: $T \geq 1$, scaling parameter $1 \leq \beta_K \leq K$ (default: $\beta_K := 1$).
**Data**: $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ distributed over $K$ machines
**Initialize**: $\boldsymbol{\alpha}_{[k]}^{(0)} \leftarrow 0$ for all machines $k$, and $\boldsymbol{w}^{(0)} \leftarrow 0$
**for** $t = 1, 2, \dots, T$
  **for *all machines*** $k = 1, 2, \dots, K$ ***in parallel***
    $(\Delta\boldsymbol{\alpha}_{[k]}, \Delta\boldsymbol{w}_k) \leftarrow \textsc{LocalDualMethod}(\boldsymbol{\alpha}_{[k]}^{(t-1)}, \boldsymbol{w}^{(t-1)})$
    $\boldsymbol{\alpha}_{[k]}^{(t)} \leftarrow \boldsymbol{\alpha}_{[k]}^{(t-1)} + \frac{\beta_K}{K}\Delta\boldsymbol{\alpha}_{[k]}$
  **end**
  *reduce* $\boldsymbol{w}^{(t)} \leftarrow \boldsymbol{w}^{(t-1)} + \frac{\beta_K}{K}\sum_{k=1}^K \Delta\boldsymbol{w}_k$
**end**

**Procedure A:** LocalDualMethod: Dual algorithm on machine $k$

**Input**: Local $\boldsymbol{\alpha}_{[k]} \in \mathbb{R}^{n_k}$, and $\boldsymbol{w} \in \mathbb{R}^d$ consistent with other coordinate blocks of $\boldsymbol{\alpha}$ s.t. $\boldsymbol{w} = A\boldsymbol{\alpha}$
**Data**: Local $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_k}$
**Output**: $\Delta\boldsymbol{\alpha}_{[k]}$ and $\Delta\boldsymbol{w} := A_{[k]}\Delta\boldsymbol{\alpha}_{[k]}$

# CoCoA

✔ <10 lines of code in *Spark*

blocks of $\alpha$ s.t. $w = A\alpha$

**Data**: Local $\{(x_i, y_i)\}_{i=1}^{n_k}$

**Output**: $\Delta\alpha_{[k]}$ and $\Delta w := A_{[k]}\Delta\alpha_{[k]}$

# CoCoA

✔ <10 lines of code in *Spark*

✔ primal-dual framework allows for *any* internal optimization method

blocks of $\boldsymbol{\alpha}$ s.t. $\boldsymbol{w} = A\boldsymbol{\alpha}$

**Data**: Local $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_k}$

**Output**: $\Delta\boldsymbol{\alpha}_{[k]}$ and $\Delta\boldsymbol{w} := A_{[k]}\Delta\boldsymbol{\alpha}_{[k]}$

# CoCoA

✔ <10 lines of code in *Spark*

✔ primal-dual framework allows for *any* internal optimization method

✔ local updates applied *immediately*

blocks of $\boldsymbol{\alpha}$ s.t. $\boldsymbol{w} = A\boldsymbol{\alpha}$

**Data**: Local $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_k}$

**Output**: $\Delta\boldsymbol{\alpha}_{[k]}$ and $\Delta\boldsymbol{w} := A_{[k]}\Delta\boldsymbol{\alpha}_{[k]}$

# CoCoA

✔ <10 lines of code in *Spark*

✔ primal-dual framework allows for *any* internal optimization method

✔ local updates applied *immediately*

✔ average over K

blocks of $\boldsymbol{\alpha}$ s.t. $\boldsymbol{w} = A\boldsymbol{\alpha}$

**Data**: Local $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_k}$

**Output**: $\Delta\boldsymbol{\alpha}_{[k]}$ and $\Delta\boldsymbol{w} := A_{[k]}\Delta\boldsymbol{\alpha}_{[k]}$

# Convergence

# Convergence

Assumptions:

- $\ell_i$ are $1/\gamma$-smooth
- LocalDualMethod makes improvement $\Theta$ per step

  e.g. for SDCA $\Theta = \left(1 - \frac{\lambda n \gamma}{1 + \lambda n \gamma} \frac{1}{\tilde{n}}\right)^H$

# Convergence

Assumptions:

- $\ell_i$ are $1/\gamma$-smooth
- LocalDualMethod makes improvement $\Theta$ per step

  e.g. for SDCA $\Theta = \left(1 - \frac{\lambda n \gamma}{1 + \lambda n \gamma} \frac{1}{\tilde{n}}\right)^H$

$$E[D(\boldsymbol{\alpha}^*) - D(\boldsymbol{\alpha}^{(T)})] \leq \left(1 - (1 - \Theta)\frac{1}{K}\frac{\lambda n \gamma}{\sigma + \lambda n \gamma}\right)^T \left(D(\boldsymbol{\alpha}^*) - D(\boldsymbol{\alpha}^{(0)})\right)$$

# Convergence

Assumptions:

- $\ell_i$ are $1/\gamma$-smooth
- LocalDualMethod makes improvement $\Theta$ per step

  e.g. for SDCA $\Theta = \left(1 - \frac{\lambda n\gamma}{1+\lambda n\gamma}\frac{1}{\tilde{n}}\right)^H$

$$E[D(\boldsymbol{\alpha}^*) - D(\boldsymbol{\alpha}^{(T)})] \leq \left(1 - (1-\Theta)\frac{1}{K}\frac{\lambda n\gamma}{\sigma + \lambda n\gamma}\right)^T \left(D(\boldsymbol{\alpha}^*) - D(\boldsymbol{\alpha}^{(0)})\right)$$

- applies also to duality gap

  $0 \leq \sigma \leq n/K$
- measure of difficulty of data partition

# Convergence

Assumptions:

ep

$E[D(\boldsymbol{\alpha}^*$ … $(\boldsymbol{\alpha}^{(0)})\Big)$

✔ it converges!

measure of difficulty of data partition

# Convergence

Assumptions:

✔ it converges!

✔ inherits convergence rate of locally used method

- measure of difficulty of data partition

# Convergence

Assumptions:

ep

$E[D(\boldsymbol{\alpha}^*$ ... $(\boldsymbol{\alpha}^{(0)})\big)$

✔ it converges!

✔ inherits convergence rate of locally used method

✔ convergence rate is linear for smooth losses

measure of difficulty of data partition

LARGE-SCALE OPTIMIZATION

COCOA

RESULTS!

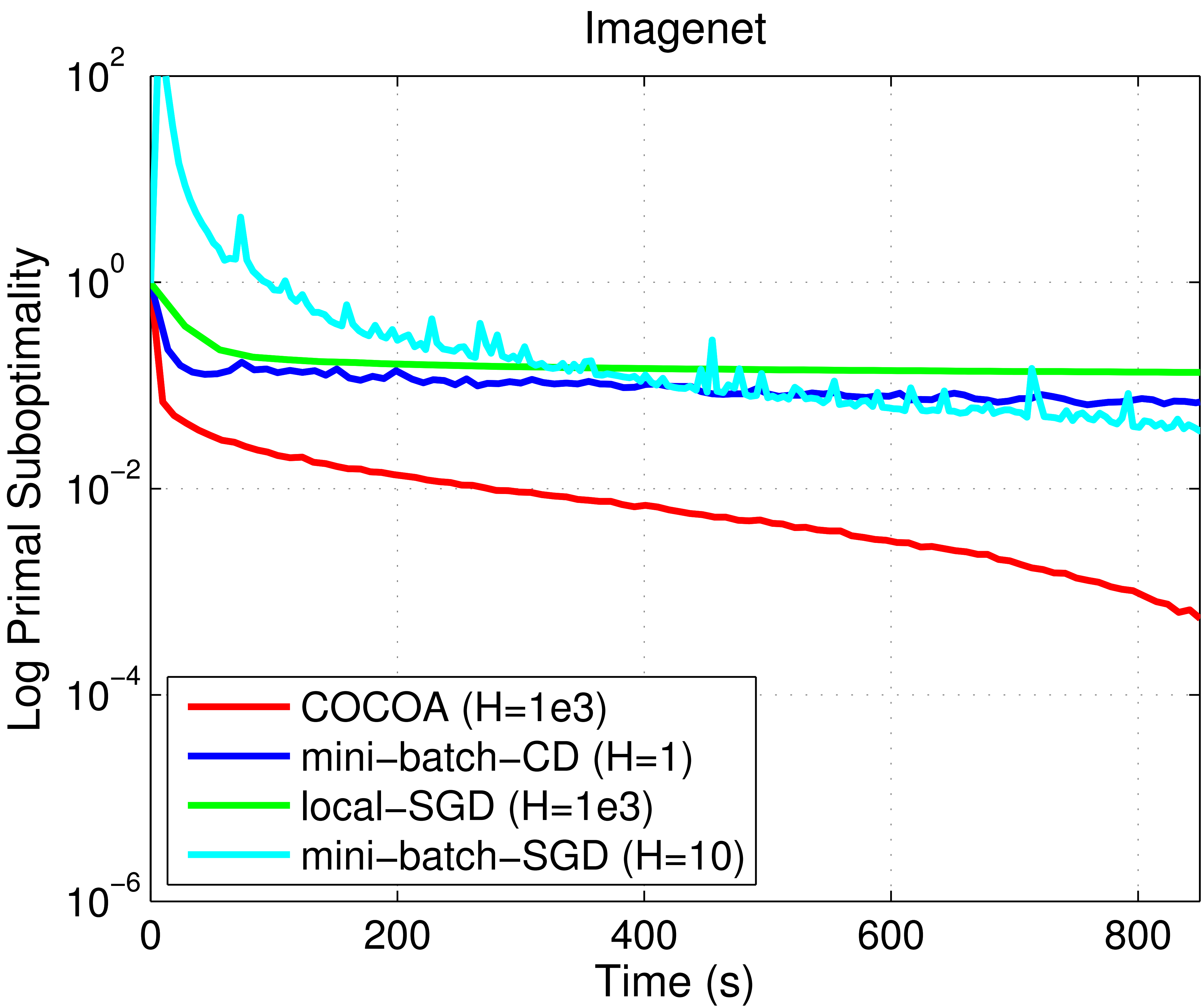
LARGE-SCALE OPTIMIZATION

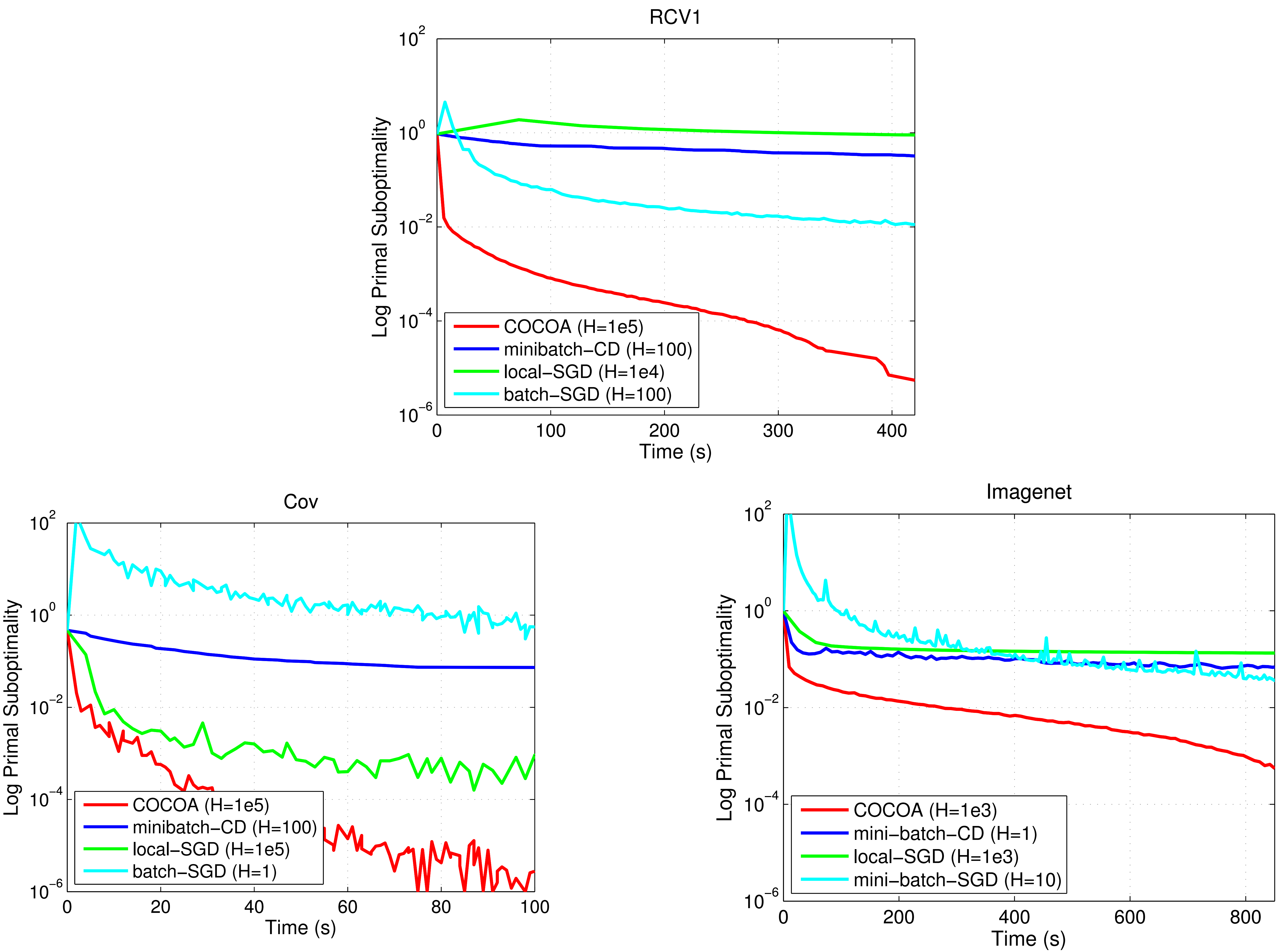
CoCoA

**RESULTS!**

# Empirical Results in Spark

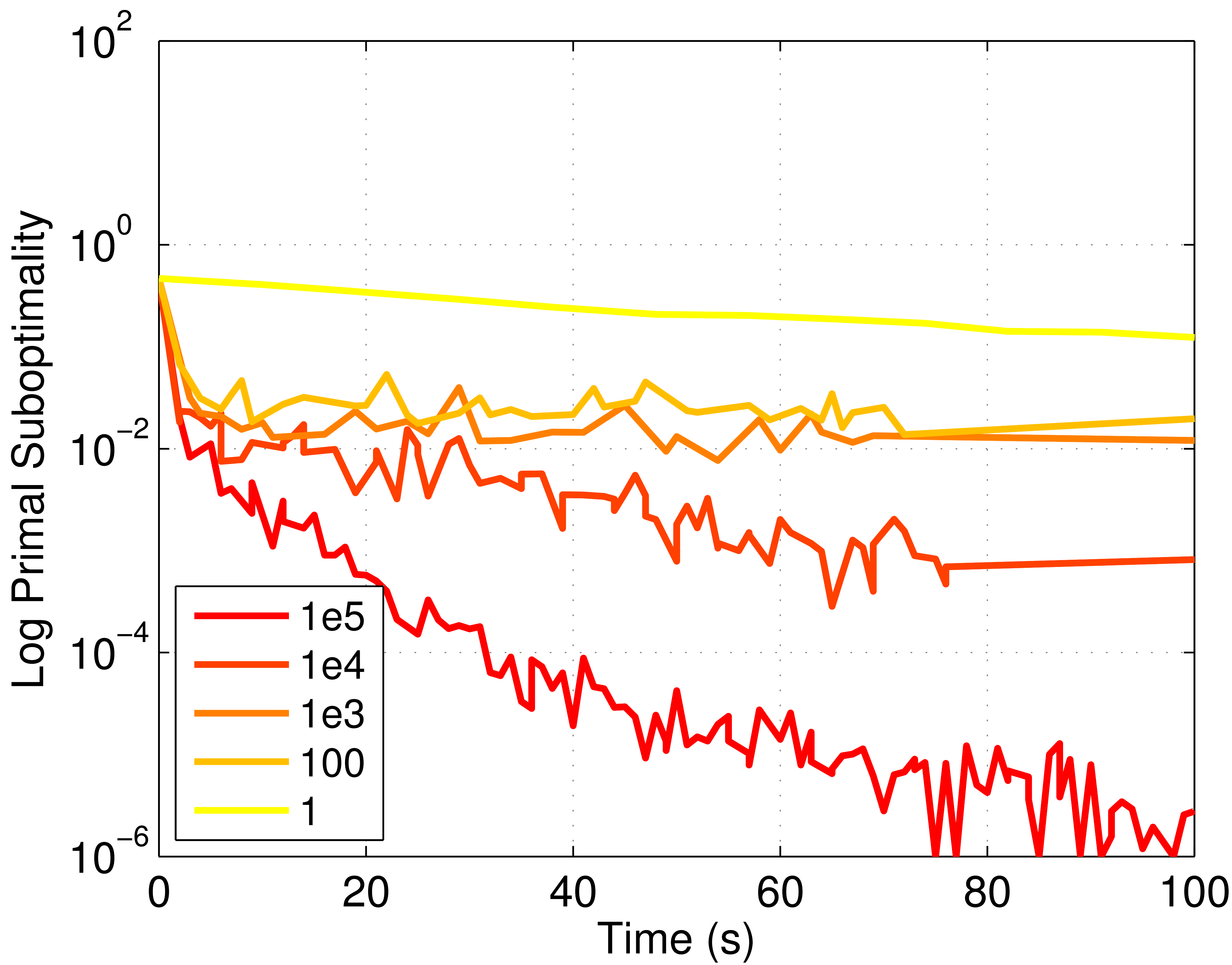
| Dataset | Training (n) | Features (d) | Sparsity | Workers (K) |
|---|---|---|---|---|
| Cov | 522,911 | 54 | 22.22% | 4 |
| Rcv1 | 677,399 | 47,236 | 0.16% | 8 |
| Imagenet | 32,751 | 160,000 | 100% | 32 |


Imagenet
Log Primal Suboptimality
$10^{2}$
$10^{0}$
$10^{-2}$
$10^{-4}$
$10^{-6}$
0
200
400
600
800
Time (s)
COCOA (H=1e3)
mini−batch−CD (H=1)
local−SGD (H=1e3)
mini−batch−SGD (H=10)



RCV1
Log Primal Suboptimality
Time (s)
COCOA (H=1e5)
minibatch−CD (H=100)
local−SGD (H=1e4)
batch−SGD (H=100)
Cov
COCOA (H=1e5)
minibatch−CD (H=100)
local−SGD (H=1e5)
batch−SGD (H=1)
Imagenet
COCOA (H=1e3)
mini−batch−CD (H=1)
local−SGD (H=1e3)
mini−batch−SGD (H=10)

# Effect of H on COCOA



# COCOA Take-Aways

# COCOA Take-Aways

- A framework for distributed optimization

# COCOA Take-Aways

- A framework for distributed optimization
- Uses state-of-the-art primal-dual methods

# COCOA Take-Aways

- A framework for distributed optimization
- Uses state-of-the-art primal-dual methods
- Reduces communication:
    - applies updates immediately
    - averages over # of machines

# COCOA Take-Aways

- A framework for distributed optimization
- Uses state-of-the-art primal-dual methods
- Reduces communication:
  - applies updates immediately
  - averages over # of machines
- Strong convergence guarantees & results in practice

# COCOA Take-Aways

- A framework for distributed optimization
- Uses state-of-the-art primal-dual methods
- Reduces communication:
    - applies updates immediately
    - averages over # of machines
- Strong convergence guarantees & results in practice

*NIPS* '14

# COCOA Take-Aways

- A framework for distributed optimization
- Uses state-of-the-art primal-dual methods
- Reduces communication:
  - applies updates immediately
  - averages over # of machines
- Strong convergence guarantees & results in practice

github.com/gingsmith/cocoa

*NIPS* '14

# Future Work

# Future Work

- optimal scaling between adding & averaging

# Future Work

- optimal scaling between adding & averaging
- similar rates for local-SGD?

# Future Work

- optimal scaling between adding & averaging
- similar rates for local-SGD?
- additional experiments: models/datasets

# Future Work

- optimal scaling between adding & averaging
- similar rates for local-SGD?
- additional experiments: models/datasets
- integration into *Spark MLlib*

# Thanks!

github.com/gingsmith/cocoa