Manifold Identification for Ultimately Communication-Efficient Distributed Optimization

Yu-Sheng Li

國立臺灣大學
National Taiwan University

Joint work with Wei-Lin Chiang (NTU) and Ching-pei Lee (NUS)

Outline

Overview

Empirical Risk Minimization

The Proposed Algorithm

Experiments

Distributed Machine Learning

Read 1 MB sequentially from memory	3 μs
Read 1 MB sequentially from network	22 μs
Read 1 MB sequentially from disk (SSD)	49 μs
Round trip in the same datacenter	500 μs

(*Latency Numbers Every Programmer Should Know.*[1])

[1] Originally by Jeff Dean in 2010, updated by Colin Scott at
https://colin-scott.github.io/personal_website/research/interactive_latency.html

Distributed Machine Learning

Read 1 MB sequentially from memory	3 μs
Read 1 MB sequentially from network	22 μs
Read 1 MB sequentially from disk (SSD)	49 μs
Round trip in the same datacenter	500 μs

(*Latency Numbers Every Programmer Should Know.*[1])

- Inter-machine communication may be more time-consuming than local computations within a machine

$$\boxed{\text{Comm. cost} = (\#\text{ Comm. rounds}) \times (\text{Bytes communicated per round})}$$

[1]Originally by Jeff Dean in 2010, updated by Colin Scott at https://colin-scott.github.io/personal_website/research/interactive_latency.html

Sparsity-inducing Regularization

- To avoid overfitting and to force some desired structure of the solution, usually a sparsity-inducing regularizer is introduced

Sparsity-inducing Regularization

- To avoid overfitting and to force some desired structure of the solution, usually a sparsity-inducing regularizer is introduced
- Example: ℓ_2- vs. ℓ_1-regularized logistic regression on **news20**

Relative reg. strength	Sparsity of solution	Test accuracy
ℓ_2-regularized		
2^0	1,355,191 (100%)	99.7449%
2^{10}	1,355,191 (100%)	97.0044%

Sparsity-inducing Regularization

- To avoid overfitting and to force some desired structure of the solution, usually a sparsity-inducing regularizer is introduced
- Example: ℓ_2- vs. ℓ_1-regularized logistic regression on **news20**

Relative reg. strength	Sparsity of solution	Test accuracy
ℓ_2-regularized		
2^0	1,355,191 (100%)	99.7449%
2^{10}	1,355,191 (100%)	97.0044%
ℓ_1-regularized		
2^0	67,071 (4.95%)	99.7499%
2^2	42,020 (3.10%)	99.7499%
2^4	14,524 (1.07%)	99.7449%
2^6	5,432 (0.40%)	99.6749%
2^8	1,472 (0.11%)	97.3495%
2^{10}	546 (0.04%)	92.8936%

Our contributions

Recall:

$$\boxed{\text{Comm. cost} = (\#\ \text{Comm. rounds}) \times (\text{Bytes communicated per round})}$$

Our contributions

Recall:

$$\boxed{\text{Comm. cost} = (\#\ \text{Comm. rounds}) \times (\text{Bytes communicated per round})}$$

- Focusing on the small subproblem
  $\Rightarrow$ fewer bytes to communicate

Our contributions

Recall:

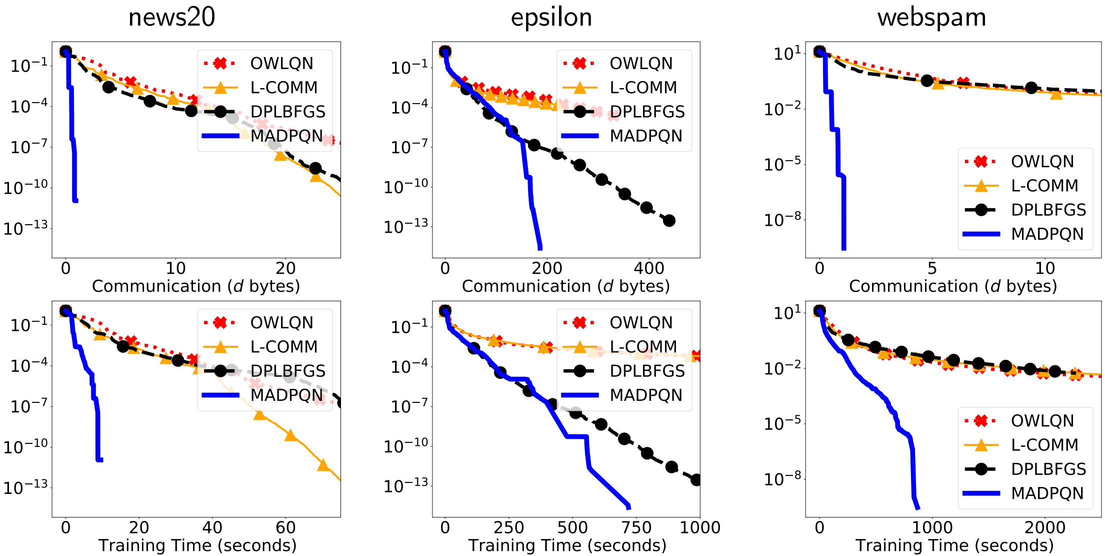
$$\boxed{\text{Comm. cost} = (\#\ \text{Comm. rounds}) \times (\text{Bytes communicated per round})}$$

- Focusing on the small subproblem
  $\Rightarrow$ fewer bytes to communicate
- Acceleration by smooth optimization in the correct manifold
  $\Rightarrow$ fewer rounds of communication

Results (ours: MADPQN)

y-axis: relative distance to the optimal value (log-scaled)

x-axis: communication costs (upper), training time (lower)



Outline

Outline

Distributed Empirical Risk Minimization (ERM)

- Train a model by minimizing a function that measures the performance on training data

$$\arg\min_{\boldsymbol{w}\in\mathbb{R}^d} \quad f(\boldsymbol{w}) := \sum_{k=1}^{K} f_k(\boldsymbol{w})$$

- There are K machines, and f_k is exclusively available on machine k

Distributed Empirical Risk Minimization (ERM)

- Train a model by minimizing a function that measures the performance on training data

$$\arg\min_{\boldsymbol{w}\in\mathbb{R}^d} \quad f(\boldsymbol{w}) := \sum_{k=1}^{K} f_k(\boldsymbol{w})$$

- There are K machines, and f_k is exclusively available on machine k
- Synchronize $\boldsymbol{w}$ or $\nabla f(\boldsymbol{w})$ by communication: communication cost per iteration is $O(d)$
- How to reduce the $O(d)$ cost?

Sparsity-inducing Regularizer

- If $\boldsymbol{w}$ is sparse throughout the training process, we only need to synchronize a shorter vector
- Regularized ERM:

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) + R(\boldsymbol{w})$$

Sparsity-inducing Regularizer

- If $\boldsymbol{w}$ is sparse throughout the training process, we only need to synchronize a shorter vector
- Regularized ERM:

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) + R(\boldsymbol{w})$$

- An ideal regularization term for forcing sparsity is the ℓ_0 norm:

$$\|\boldsymbol{w}\|_0 = \text{number of nonzeros in } \boldsymbol{w}$$

Sparsity-inducing Regularizer

- If $\boldsymbol{w}$ is sparse throughout the training process, we only need to synchronize a shorter vector
- Regularized ERM:

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) + R(\boldsymbol{w})$$

- An ideal regularization term for forcing sparsity is the ℓ_0 norm:

$$\|\boldsymbol{w}\|_0 = \text{number of nonzeros in } \boldsymbol{w}$$

- But this norm is not continuous and hence hard to optimize
- A good surrogate is the ℓ_1 norm $\|\boldsymbol{w}\|_1 = \sum_{i=1}^{d} |w_i|$
- Our algorithm works for other partly smooth R, e.g. group-LASSO

The Regularized Problem

- Now the problem becomes
$$\min_{\boldsymbol{w}} f(\boldsymbol{w}) + \|\boldsymbol{w}\|_1,$$
which is harder to minimize than $f(\boldsymbol{w})$ alone since $\|\boldsymbol{w}\|_1$ is not differentiable
- As the gradient may not even exist, gradient descent or Newton method cannot be directly applied

Proximal Quasi-Newton

- Proximal gradient is a simple algorithm that solves

$$\min_{\boldsymbol{w}'} \nabla f(\boldsymbol{w})^\top (\boldsymbol{w}' - \boldsymbol{w}) + \frac{1}{2\alpha}\|\boldsymbol{w}' - \boldsymbol{w}\|_2^2 + \|\boldsymbol{w}'\|_1,$$

  where α is the step size for the current iteration
- Each calculation of ∇f requires one round of communication

Proximal Quasi-Newton

- Proximal gradient is a simple algorithm that solves

$$\min_{\boldsymbol{w}'} \nabla f(\boldsymbol{w})^\top (\boldsymbol{w}' - \boldsymbol{w}) + \frac{1}{2\alpha} \|\boldsymbol{w}' - \boldsymbol{w}\|_2^2 + \|\boldsymbol{w}'\|_1,$$

  where α is the step size for the current iteration
- Each calculation of ∇f requires one round of communication
- To reduce the amount of communication, we include some second-order information:

$$\text{reducing iterations} \Rightarrow \text{reducing rounds of communication}$$

- Replace the term $\|\boldsymbol{w}' - \boldsymbol{w}\|_2^2 / 2\alpha$ with $(\boldsymbol{w}' - \boldsymbol{w})^\top H (\boldsymbol{w}' - \boldsymbol{w})/2$ for some $H \approx \nabla^2 f(\boldsymbol{w})$

Outline

Utilizing Sparsity

- Even if we only update the nonzero entries of $\boldsymbol{w}$, if we still compute the whole gradient $\nabla f(\boldsymbol{w})$, then the communication cost remains $O(d)$

Utilizing Sparsity

- Even if we only update the nonzero entries of $\boldsymbol{w}$, if we still compute the whole gradient $\nabla f(\boldsymbol{w})$, then the communication cost remains $O(d)$
- Guess: if $w_i = 0$ at some iteration and it is likely to stay 0 at the next iteration, it remains 0 at the final solution
- Then we only solve the subproblem with respect to the coordinates that are likely to be nonzero

Utilizing Sparsity

- Even if we only update the nonzero entries of $\boldsymbol{w}$, if we still compute the whole gradient $\nabla f(\boldsymbol{w})$, then the communication cost remains $O(d)$
- Guess: if $w_i = 0$ at some iteration and it is likely to stay 0 at the next iteration, it remains 0 at the final solution
- Then we only solve the subproblem with respect to the coordinates that are likely to be nonzero
- A progressive shrinking approach: once we guess $w_i = 0$, we remove those coordinates from our problem in future iterations
- So the number of nonzeros in $\boldsymbol{w}$ (i.e. $\|\boldsymbol{w}\|_0$) gradually decreases

Convergence Issue

- What if our guess was wrong at some iteration?

Convergence Issue

- What if our guess was wrong at some iteration?
- Need to double-check: when some stopping criterion is met, we restart with all coordinates
- Training is terminated only when our model can hardly be improved using all coordinates

More Acceleration by Smooth Optimization

- $|w_i|$ becomes twice-differentiable when $w_i \neq 0$
- If the coordinates where $w_i \neq 0$ are fixed, the proximal approach is not needed anymore
- The problem can then be transformed into a smooth one for faster convergence

More Acceleration by Smooth Optimization

- $|w_i|$ becomes twice-differentiable when $w_i \neq 0$
- If the coordinates where $w_i \neq 0$ are fixed, the proximal approach is not needed anymore
- The problem can then be transformed into a smooth one for faster convergence
- When the nonzero pattern (manifold) does not change for some iterations, it is likely to be the final pattern

More Acceleration by Smooth Optimization

- $|w_i|$ becomes twice-differentiable when $w_i \neq 0$
- If the coordinates where $w_i \neq 0$ are fixed, the proximal approach is not needed anymore
- The problem can then be transformed into a smooth one for faster convergence
- When the nonzero pattern (manifold) does not change for some iterations, it is likely to be the final pattern
- Example with $d = 5$:

$$\{1, 2, 3, 4, 5\}$$

More Acceleration by Smooth Optimization

- $|w_i|$ becomes twice-differentiable when $w_i \neq 0$
- If the coordinates where $w_i \neq 0$ are fixed, the proximal approach is not needed anymore
- The problem can then be transformed into a smooth one for faster convergence
- When the nonzero pattern (manifold) does not change for some iterations, it is likely to be the final pattern
- Example with $d = 5$:

$$\{1, 2, 3, 4, 5\} \rightarrow \{2, 3, 5\}$$

More Acceleration by Smooth Optimization

- $|w_i|$ becomes twice-differentiable when $w_i \neq 0$
- If the coordinates where $w_i \neq 0$ are fixed, the proximal approach is not needed anymore
- The problem can then be transformed into a smooth one for faster convergence
- When the nonzero pattern (manifold) does not change for some iterations, it is likely to be the final pattern
- Example with $d = 5$:

$$\{1, 2, 3, 4, 5\} \to \{2, 3, 5\} \to \{2, 5\}$$

More Acceleration by Smooth Optimization

- $|w_i|$ becomes twice-differentiable when $w_i \neq 0$
- If the coordinates where $w_i \neq 0$ are fixed, the proximal approach is not needed anymore
- The problem can then be transformed into a smooth one for faster convergence
- When the nonzero pattern (manifold) does not change for some iterations, it is likely to be the final pattern
- Example with $d = 5$:

$$\{1, 2, 3, 4, 5\} \rightarrow \{2, 3, 5\} \rightarrow \{2, 5\} \rightarrow \{2, 5\}$$

More Acceleration by Smooth Optimization

- $|w_i|$ becomes twice-differentiable when $w_i \neq 0$
- If the coordinates where $w_i \neq 0$ are fixed, the proximal approach is not needed anymore
- The problem can then be transformed into a smooth one for faster convergence
- When the nonzero pattern (manifold) does not change for some iterations, it is likely to be the final pattern
- Example with $d = 5$:

$$\{1, 2, 3, 4, 5\} \rightarrow \{2, 3, 5\} \rightarrow \{2, 5\} \rightarrow \{2, 5\} \rightarrow \{2, 5\}$$

More Acceleration by Smooth Optimization

- $|w_i|$ becomes twice-differentiable when $w_i \neq 0$
- If the coordinates where $w_i \neq 0$ are fixed, the proximal approach is not needed anymore
- The problem can then be transformed into a smooth one for faster convergence
- When the nonzero pattern (manifold) does not change for some iterations, it is likely to be the final pattern
- Example with $d = 5$:

$$\{1,2,3,4,5\} \rightarrow \{2,3,5\} \rightarrow \{2,5\} \rightarrow \{2,5\} \rightarrow \{2,5\} \xrightarrow{\text{accelerate}} \cdots$$

More Acceleration by Smooth Optimization

- $|w_i|$ becomes twice-differentiable when $w_i \neq 0$
- If the coordinates where $w_i \neq 0$ are fixed, the proximal approach is not needed anymore
- The problem can then be transformed into a smooth one for faster convergence
- When the nonzero pattern (manifold) does not change for some iterations, it is likely to be the final pattern
- Example with $d = 5$:

$$\{1,2,3,4,5\} \to \{2,3,5\} \to \{2,5\} \to \{2,5\} \to \{2,5\} \xrightarrow{\text{accelerate}} \cdots$$
$$\xrightarrow{\text{restart}} \{1,2,3,4,5\} \to \cdots$$

More Acceleration by Smooth Optimization

- $|w_i|$ becomes twice-differentiable when $w_i \neq 0$
- If the coordinates where $w_i \neq 0$ are fixed, the proximal approach is not needed anymore
- The problem can then be transformed into a smooth one for faster convergence
- When the nonzero pattern (manifold) does not change for some iterations, it is likely to be the final pattern
- Example with $d = 5$:

$$
\begin{aligned}
&\{1,2,3,4,5\} \to \{2,3,5\} \to \{2,5\} \to \{2,5\} \to \{2,5\} \xrightarrow{\text{accelerate}} \cdots \\
\xrightarrow{\text{restart}} &\{1,2,3,4,5\} \to \cdots \\
\xrightarrow{\text{restart}} &\{1,2,3,4,5\} \to \cdots
\end{aligned}
$$

More Acceleration by Smooth Optimization

- $|w_i|$ becomes twice-differentiable when $w_i \neq 0$
- If the coordinates where $w_i \neq 0$ are fixed, the proximal approach is not needed anymore
- The problem can then be transformed into a smooth one for faster convergence
- When the nonzero pattern (manifold) does not change for some iterations, it is likely to be the final pattern
- Example with $d = 5$:

$$
\begin{aligned}
&\{1,2,3,4,5\} \to \{2,3,5\} \to \{2,5\} \to \{2,5\} \to \{2,5\} \xrightarrow{\text{accelerate}} \cdots \\
\xrightarrow{\text{restart}} &\{1,2,3,4,5\} \to \cdots \\
\xrightarrow{\text{restart}} &\{1,2,3,4,5\} \to \cdots \\
\xrightarrow{\text{restart}} &\{1,2,3,4,5\} \to \text{terminated}
\end{aligned}
$$

Theoretical Guarantees

Theorem

If a cluster point $\boldsymbol{w}^*$ *of* $\{\boldsymbol{w}$ after each restart$\}$ *satisfies*

$$\mathbf{0} \in \text{relint}\left(\nabla f(\boldsymbol{w}^*) + \partial R(\boldsymbol{w}^*)\right),$$

then the manifold of $\boldsymbol{w}^*$ *will be identified within finite restarts.*

Outline

Settings

- We show the effectiveness of the proposed approach by ℓ_1-regularized logistic regression

$$\min_{\boldsymbol{w}} \quad \sum_{i=1}^{n} \log(1 + \exp(-y_i \boldsymbol{x}_i^\top \boldsymbol{w})) + \|\boldsymbol{w}\|_1,$$

where there are n instances with features $\boldsymbol{x}_i \in \mathbb{R}^d$ and labels $y_i \in \{-1, 1\}$

Settings

- We show the effectiveness of the proposed approach by ℓ_1-regularized logistic regression

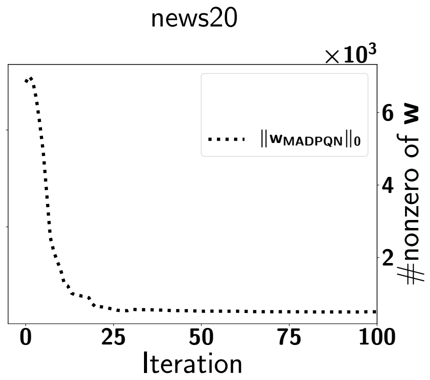
$$\min_{\boldsymbol{w}} \quad \sum_{i=1}^{n} \log(1 + \exp(-y_i \boldsymbol{x}_i^\top \boldsymbol{w})) + \|\boldsymbol{w}\|_1,$$

  where there are n instances with features $\boldsymbol{x}_i \in \mathbb{R}^d$ and labels $y_i \in \{-1, 1\}$
- The instances are evenly split across $K = 10$ machines, connected by Intel MPI in a 1Gbps network environment

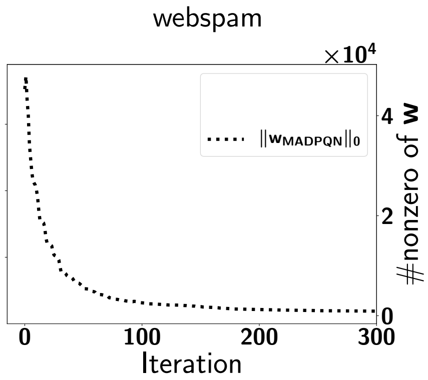
Data Statistics

Data set	Instances (n)	Features (d)	Nonzeros in optimal $\boldsymbol{w}^*$
news20	19,996	1,355,191	506
epsilon	400,000	2,000	1,463
webspam	350,000	16,609,143	793
url	2,396,130	3,231,961	25,399
avazu-site	25,832,830	999,962	11,858
KDD2010-b	19,264,097	29,890,096	2,005,632

Results

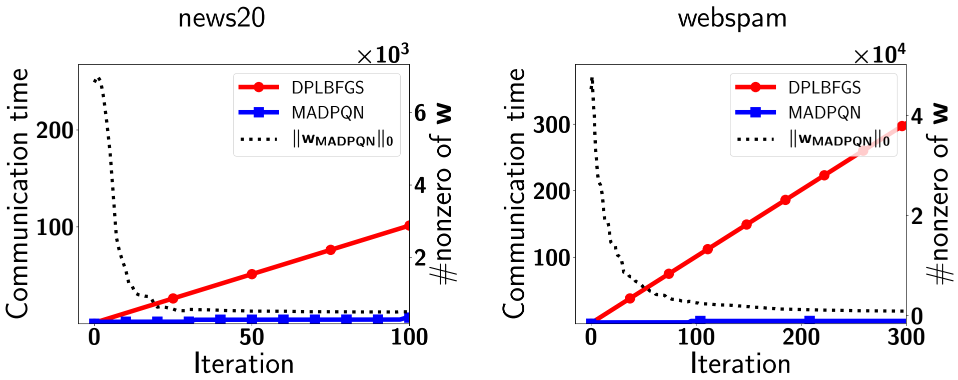


news20
×10^3
$\|\mathbf{w}_{\text{MADPQN}}\|_0$
6
4
2
#nonzero of w
0
25
50
75
100
Iteration



webspam
×10^4
$\|\mathbf{w}_{\text{MADPQN}}\|_0$
4
2
0
#nonzero of w
0
100
200
300
Iteration

Results



- DPLBFGS:
  a distributed proximal quasi-Newton method (Lee et al. 2019)
- Manifold-Aware Distributed Proximal Quasi-Newton (MADPQN):
  DPLBFGS + manifold selection + further acceleration

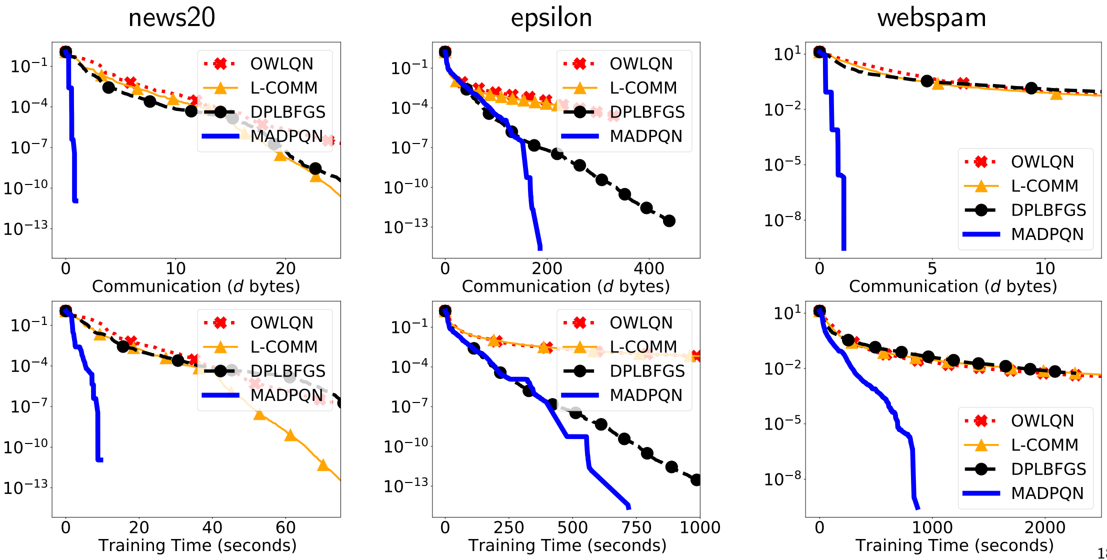
Comparison with state of the art

- OWLQN (Andrew and Gao 2007):
  an extension of a quasi-Newton method, LBFGS, which is the most commonly used distributed method
- L-COMM (Chiang et al. 2018):
  an extension of the common directions method (Wang et al. 2016)
- DPLBFGS (Lee et al. 2019):
  a distributed proximal LBFGS method
- MADPQN:
  Our proposed Manifold-Aware Distributed Proximal Quasi-Newton method

Results

y-axis: relative distance to the optimal value (log-scaled)

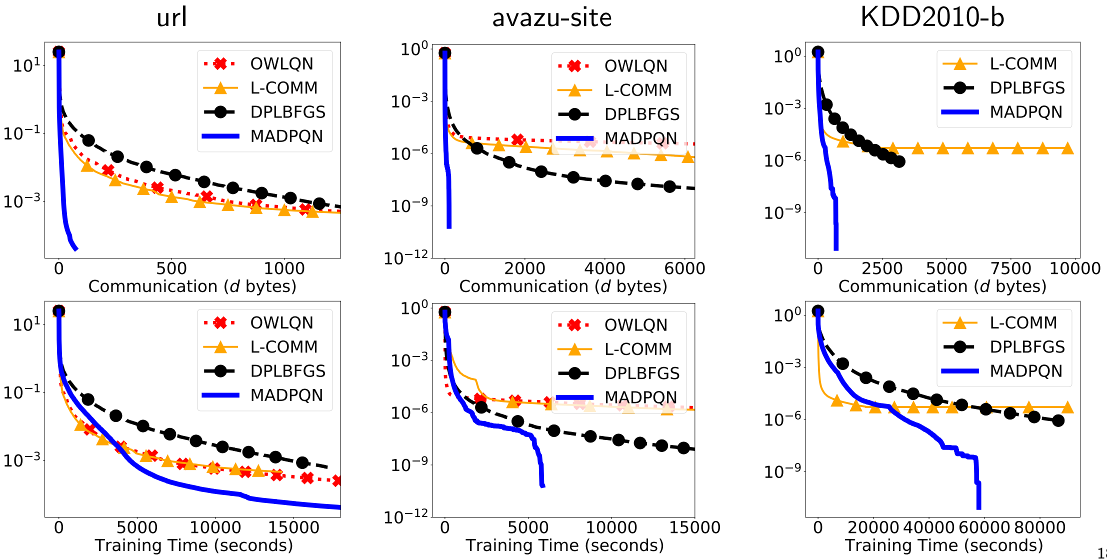
x-axis: communication costs (upper), training time (lower)



Results

y-axis: relative distance to the optimal value (log-scaled)

x-axis: communication costs (upper), training time (lower)



Conclusions

- Communication may be the bottleneck in distributed machine learning
- Communication cost can be reduced by utilizing the sparsity pattern throughout training
- Second-order information further improves convergence in later stage
- Theoretical support on manifold identification and superlinear convergence
- Source code to be released soon