



Carnegie
Mellon
University

RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

Google

Efficient Mini-batch Training for Stochastic Optimization

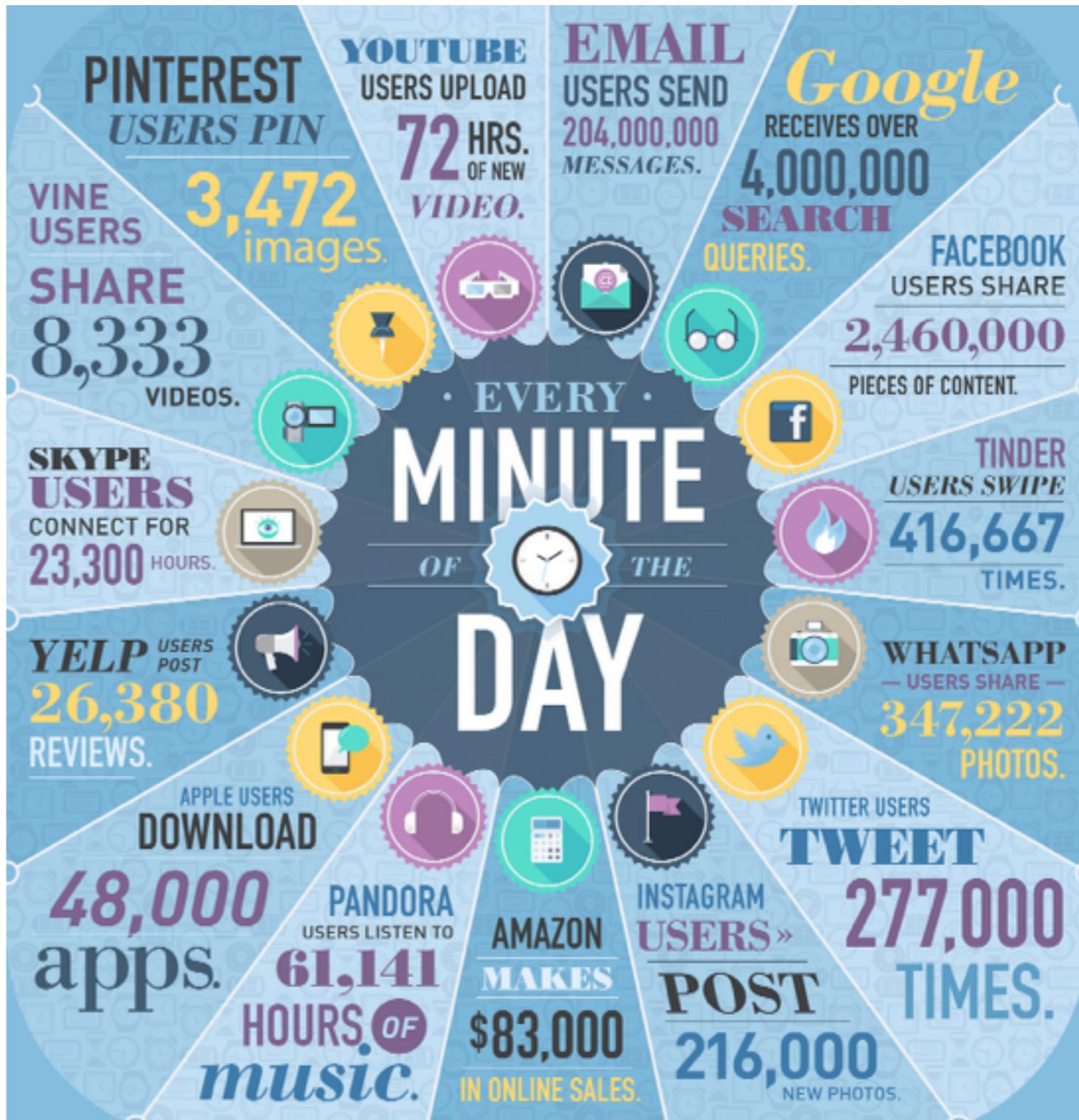
Mu Li

muli@cs.cmu.edu

Joint work with Tong Zhang, Yuqiang Chen, and Alex Smola

Big data

- ♦ A lot of “big” data
 - ✓ texts
 - ✓ images
 - ✓ voices
 - ✓ videos
- ♦ Most of them are user activities
 - ✓ can be modeled as supervised learning



Objective

- ♦ Convex optimization:

$$\min_{w \in \Omega} \frac{1}{n} \sum_{i=1}^n \phi_i(w)$$

- ♦ For example: Risk minimization

$$\phi_i(w) = \ell(x_i, y_i, w) + \lambda c(w)$$

- ✓ (x_i, y_i) are example pairs

Stochastic gradient descent (SGD)

- ♦ Process an example each time

one example

for $t = 1, 2, \dots, T$

draw a random example i_t

update $w_t = w_{t-1} - \eta_t \nabla \phi_{i_t}(w_{t-1})$

- ♦ Convergence rate $O(1/\sqrt{T})$
- ♦ Sequential, hard to parallelize

Minibatch SGD

- ♦ One example \rightarrow several examples

for $t = 1, 2, \dots, T$

draw a random minibatch

$$I_t \subset \{1, \dots, n\}$$

update

$$w_t = w_{t-1} - \eta_t \nabla \phi_{I_t}(w_{t-1})$$

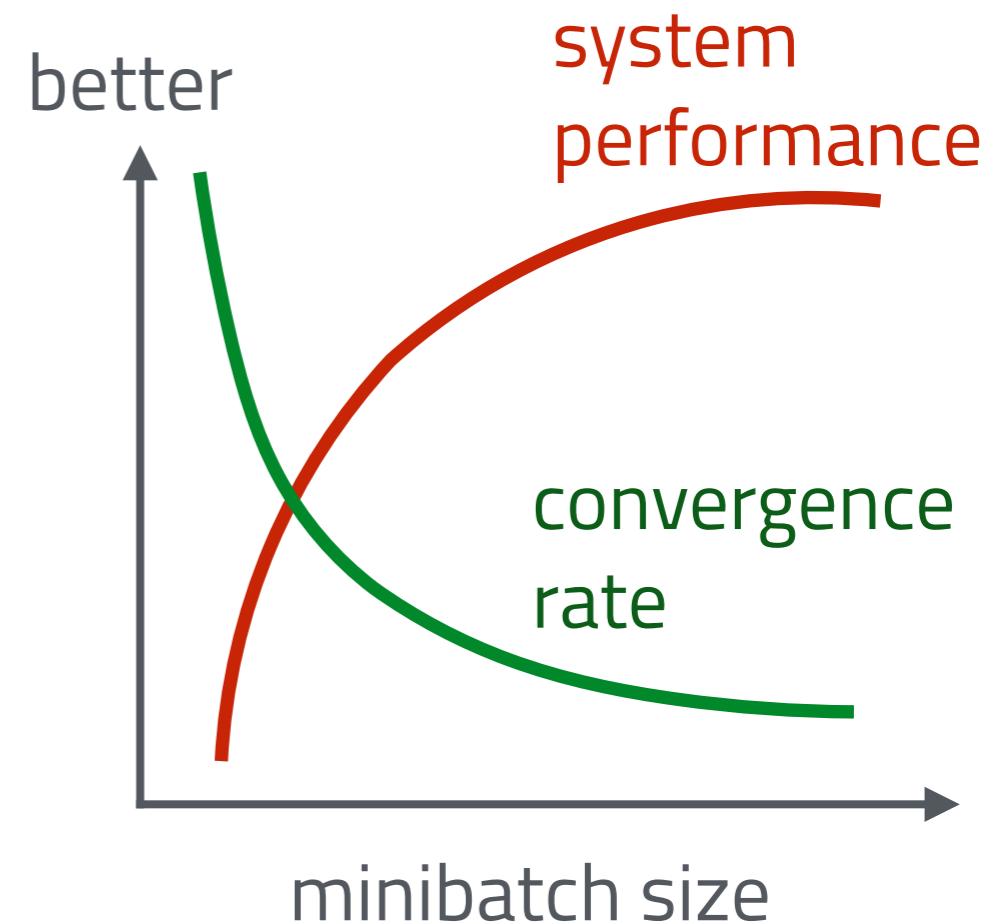
$$\phi_{I_t}(w) = \frac{1}{|I_t|} \sum_{i \in I_t} \phi_i(w)$$

example minibatch

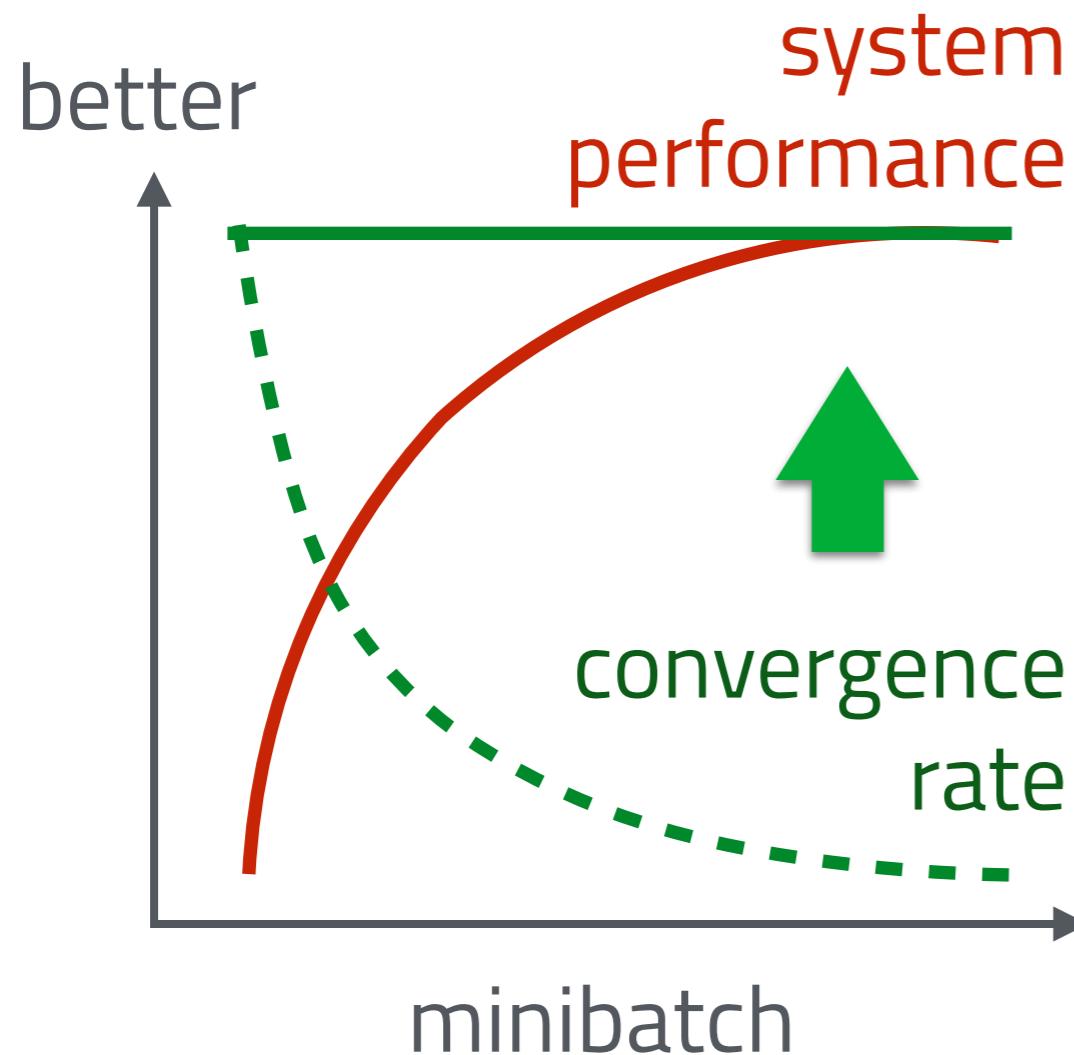
- ♦  Efficient parallel/distributed implementation within a minibatch

Effects of the minibatch size

- ♦ Fix #examples bT :
- ♦ minibatch size $b \uparrow$, then #iteration $T \downarrow$
- ♦ System performance \uparrow
 - ✓ #synchronization \downarrow
 - ✓ network communication \downarrow
- ♦ Convergence rate \downarrow
 - ✓ it is $\mathcal{O}(1/\sqrt{bT} + 1/T)$
 - ✓ $O(1/T) \uparrow$



Our goal



Key idea:

When minibatch size ↑,
sample variance ↓.
Solve a more “accurate”
optimization problem
over each minibatch.

Observation

- ♦ Rewrite the update rule of minibatch SGD:

$$w_t = \operatorname{argmin}_{w \in \Omega} \left[\phi_{I_t}(w_{t-1}) + \langle \nabla \phi_{I_t}(w_{t-1}), w - w_{t-1} \rangle + \frac{1}{2\eta_t} \|w - w_{t-1}\|_2^2 \right]$$

A coarse first-order
approximation

a conservative
penalty

- ♦ Fast to solve
- ♦ Data utilization is low
 - ✓ large switching cost to the next minibatch

The proposed solution: EMSO

- ♦ Solve a conservative subproblem:

$$w_t = \operatorname{argmin}_{w \in \Omega} \left[\phi_{I_t}(w) + \frac{\gamma_t}{2} \|w - w_{t-1}\|_2^2 \right].$$

exact objective
over minibatch

a conservative
penalty

- ♦ achieve a full utilization of the minibatch
- ♦ avoid over utilization

Convergence rate

- ♦ Minibatch SGD: $\mathcal{O}(1/\sqrt{bT} + 1/T)$
- ♦ EMSO: $\mathcal{O}(1/\sqrt{bT})$
 - ✓ only depends on the #examples
- ♦ Can be further improved when the objective is λ -strongly convex

$$\mathcal{O}(\log T/(\lambda bT) + \lambda/(\sqrt{bT}))$$

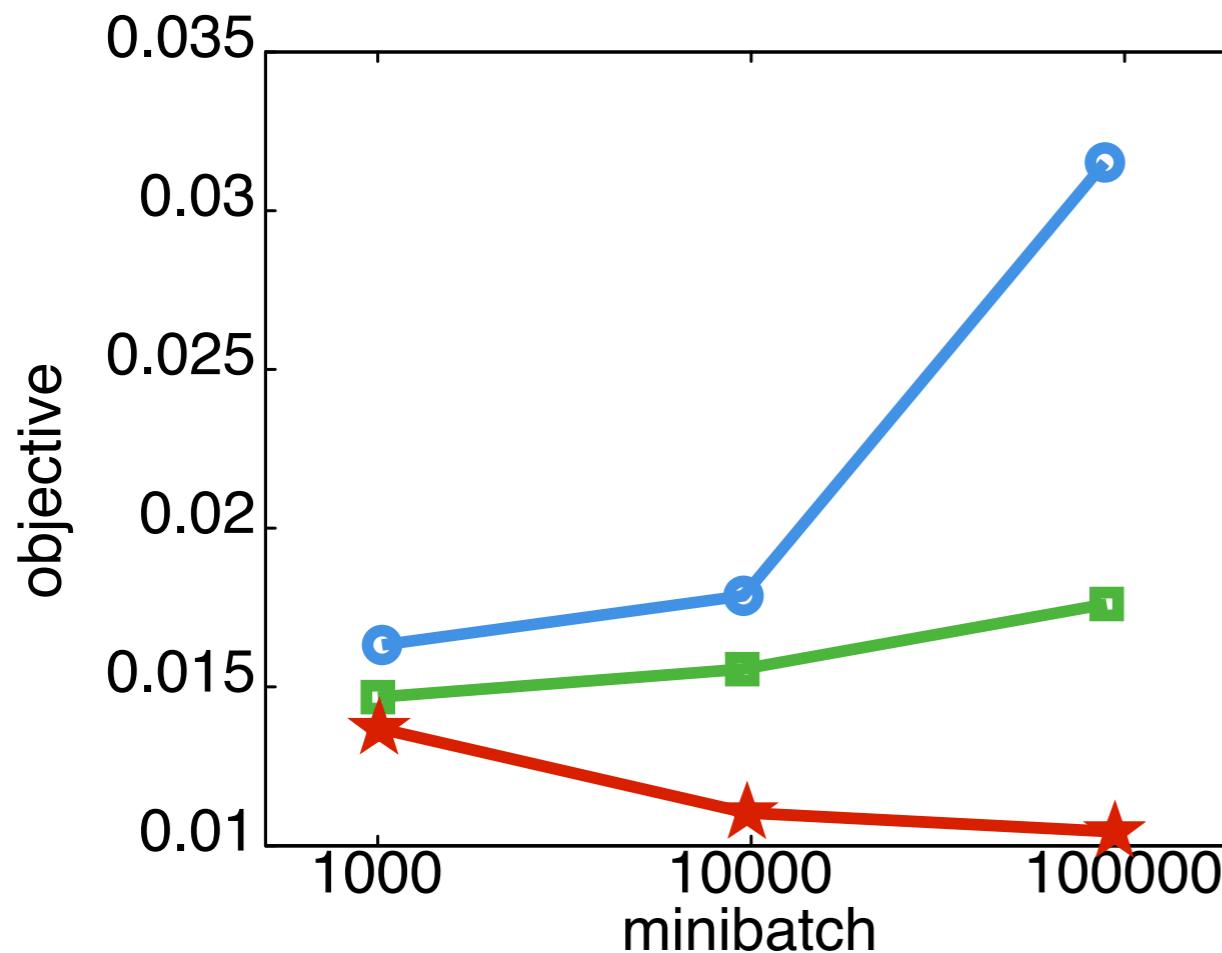
How to solve the subproblem

$$w_t = \operatorname{argmin}_{w \in \Omega} \left[\phi_{I_t}(w) + \frac{\gamma_t}{2} \|w - w_{t-1}\|_2^2 \right].$$

- ♦ The conservative subproblem can be solved by standard technologies:
 - ✓ EMSO-GD: by gradient descent
 - ✓ EMSO-CD: by coordinate descent
- ♦ No need to solve it exactly
- ♦ Early stopping:
 - ✓ fix the #iterations be a small constant

Convergence does not slow down with minibatch size

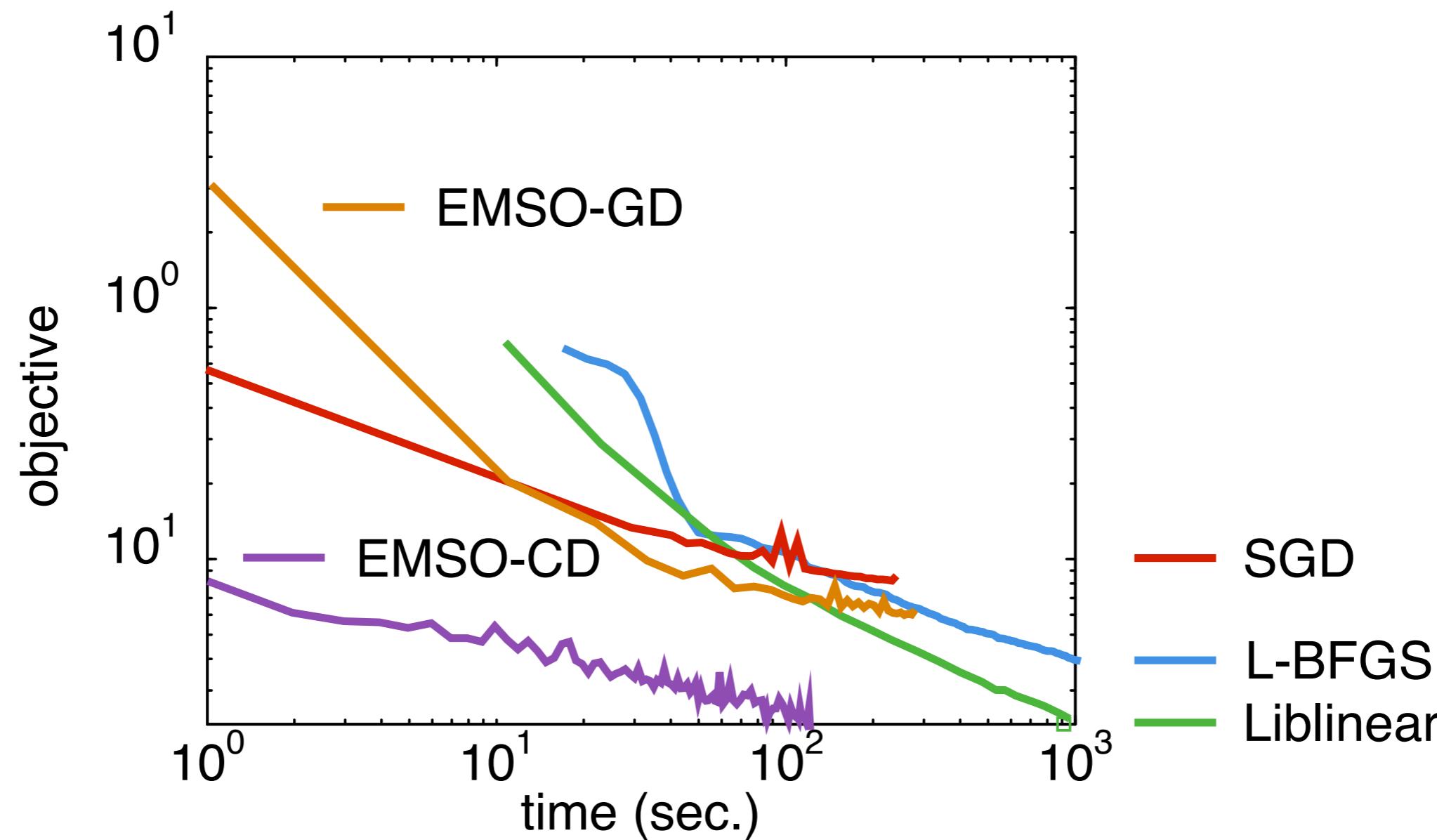
- ♦ Fix #iterations
- ♦ Logistic regression
- ♦ Dataset KDD04: 146K examples, 76 features



minibatch SGD
EMSO-GD:
pass the minibatch 5 times
EMSO-CD:
pass the minibatch 2 times

EMSO-CD outperforms other algorithms

- Dataset URL: 2.4M #examples, 3.2M #features



Distributed model averaging

assume d machines

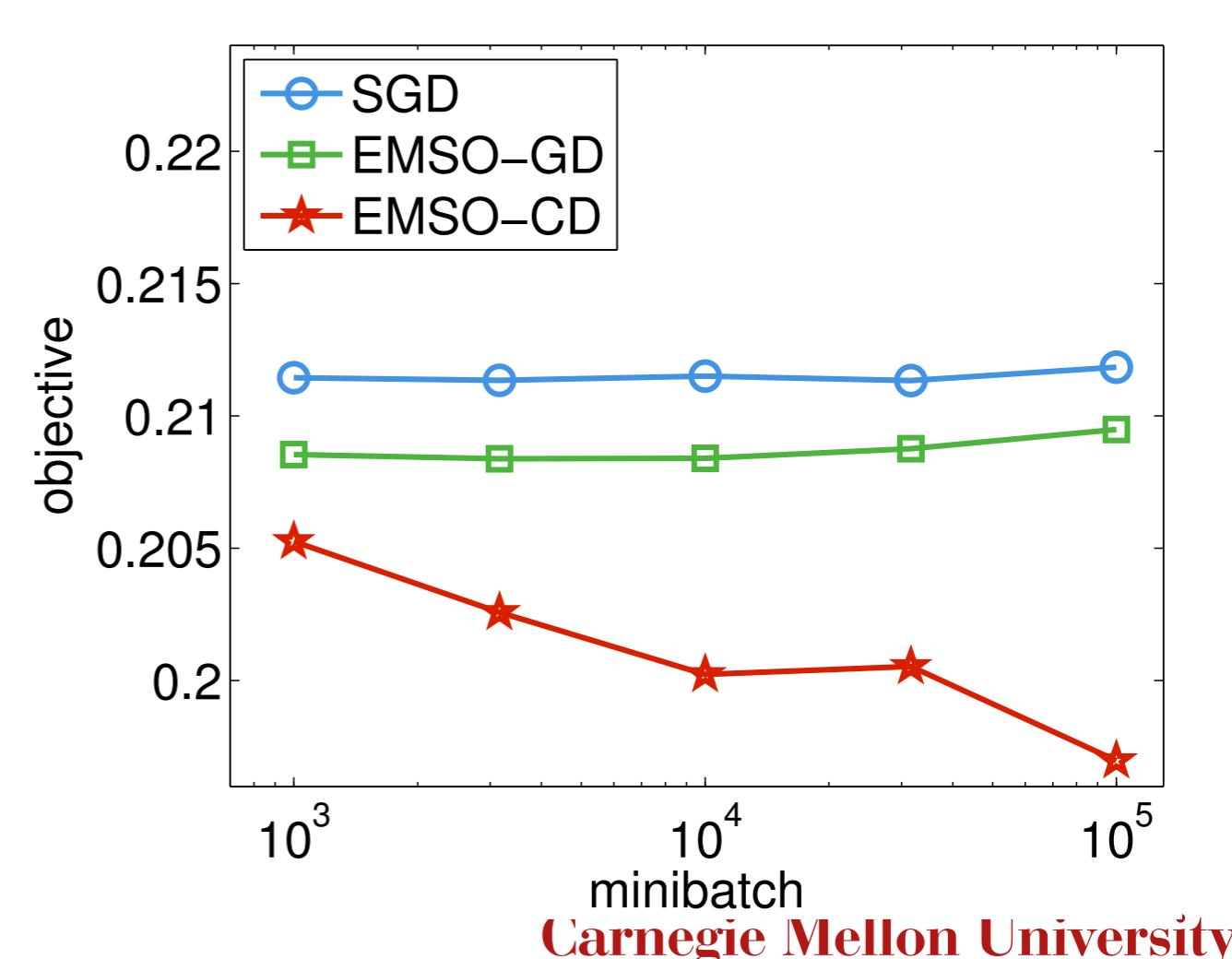
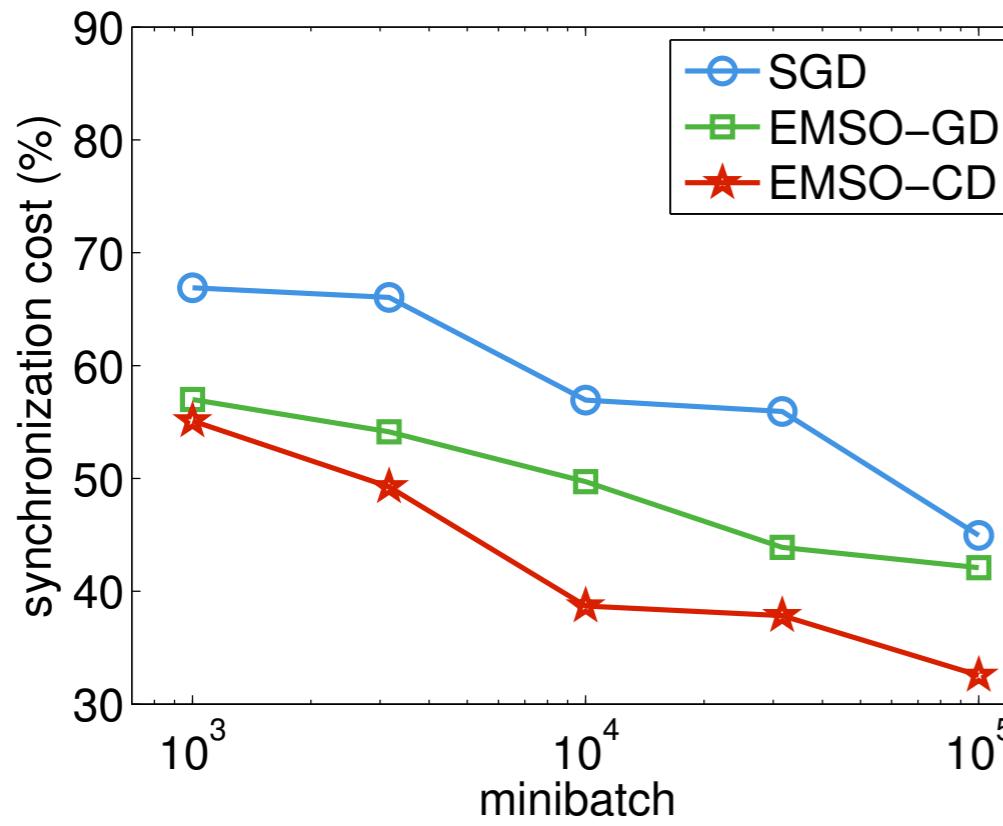
for each minibatch I_t :

1. partition I_t into $\{I_{t,1}, \dots, I_{t,d}\}$
2. machine i solve the conservative subproblem
on its own data partition $I_{t,i}$
3. average model via communication

$$w_t = \frac{1}{d} \sum_{i=1}^d w_t^{(i)}$$

EMSO-CD outperforms other algorithms

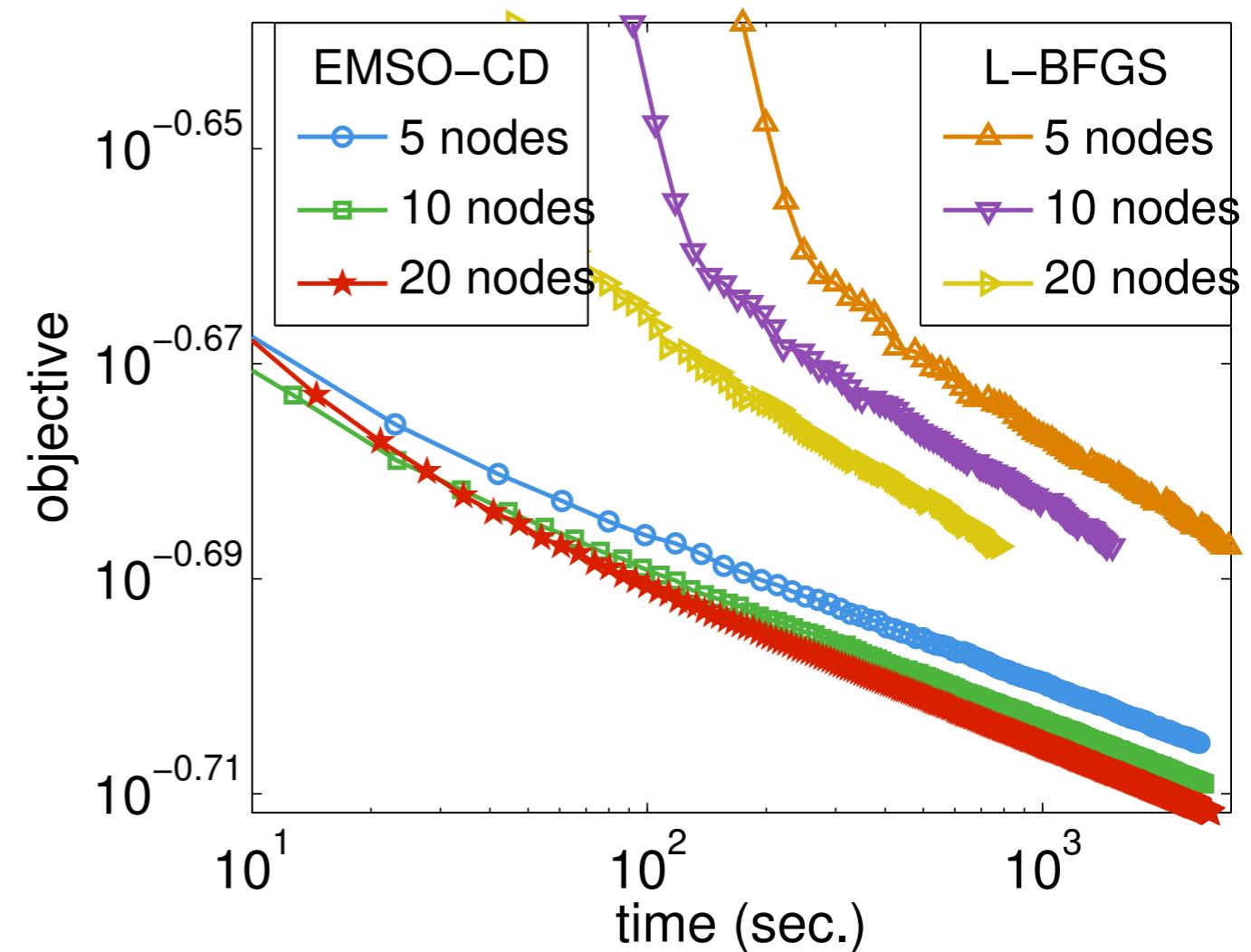
- ♦ Dataset CTR: 142M #examples, 28M #features, raw text data size 300GB
- ♦ Distributed over 12 machines
 - ♦ Synchronization cost↓ when minibatch size↑
 - ♦ Fix run time



Scalability of EMSO-CD

- ◆ Dataset CTR
- ◆ Fix target objective
- ◆ Compare to L-BFGS

#machines	time (sec)	speedup
5	2439	1x
10	1367	1.78x
20	962	2.54x



Conclusion

- ♦ EMSO: solve the conservative subproblem in each minibatch:

$$w_t = \operatorname{argmin}_{w \in \Omega} \left[\phi_{I_t}(w) + \frac{\gamma_t}{2} \|w - w_{t-1}\|_2^2 \right].$$

- ♦ Faster convergence rate $\mathcal{O}(1/\sqrt{bT})$
 - ✓ Does not slow down when minibatch size ↑
- ♦ Improve the effective workload
 - ✓ Demonstrated in experiments with real large scale datasets