

# Big Data Analytics

## D. Distributed Machine Learning Algorithms / D.2 Distributed Matrix Factorization

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)  
Institute of Computer Science  
University of Hildesheim, Germany

# Syllabus

- Tue. 4.4. (1) 0. Introduction
- A. Parallel Computing**
- Tue. 11.4. (2) A.1 Threads
- Tue. 18.4. (3) A.2 Message Passing Interface (MPI)
- B. Distributed Storage**
- Tue. 25.4. (4) B.1 Distributed File Systems
- Tue. 2.5. (5) B.2 Partitioning of Relational Databases
- Tue. 9.5. (6) B.3 NoSQL Databases
- Tue. 16.5. (7) A.3 Graphical Processing Units (GPUs)
- C. Distributed Computing Environments**
- Tue. 23.5. (8) C.1 Map-Reduce
- Tue. 30.5. (9) C.2 Resilient Distributed Datasets (Spark)
- Tue. 6.6. — — Pentecoste Break —
- D. Distributed Machine Learning Algorithms**
- Tue. 13.6. (10) D.1 Distributed Stochastic Gradient Descent
- Tue. 20.6. (11) D.2 Distributed Matrix Factorization
- Tue. 27.6. (12) D.3 Alternating Direction Method of Multipliers (ADMM)
- Tue. 4.7. (13) Questions and Answers

# Outline

1. Introduction
2. Matrix Factorization via Distributed SGD
3. NOMAD

# Outline

## 1. Introduction

## 2. Matrix Factorization via Distributed SGD

## 3. NOMAD

# The Matrix Completion Problem

Given

- ▶ the values  $\mathcal{D} \subseteq [N] \times [M] \times \mathbb{R}$  of some cells of an unknown matrix  $Y \in \mathbb{R}^{N \times M}$  (called **data**) and
- ▶ a function  $\ell : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$  (called **loss**),

predict the values of the missing cells,

i.e. find a completion  $\hat{Y} \in \mathbb{R}^{N \times M}$  with minimal error

$$\text{err}(\hat{Y}, Y) := \sum_{n=1}^N \sum_{m=1}^M \ell(Y_{n,m}, \hat{Y}_{n,m})$$

Note:  $[N] := \{1, \dots, N\}$ .

# The Matrix Factorization Model

- ▶ the basic model:

$$\hat{Y} := WH, \quad W \in \mathbb{R}^{N \times K}, H \in \mathbb{R}^{K \times M}$$

$$\text{i.e., } \hat{Y}_{n,m} := W_{n,\cdot} H_{\cdot,m}$$

- ▶  $K$  is called **latent dimension**

# The Matrix Factorization Model

- ▶ the basic model:

$$\hat{Y} := WH, \quad W \in \mathbb{R}^{N \times K}, H \in \mathbb{R}^{K \times M}$$

$$\text{i.e., } \hat{Y}_{n,m} := W_{n,\cdot} H_{\cdot,m}$$

- ▶  $K$  is called **latent dimension**
  
- ▶ parameters are **regularized**, i.e., minimize

$$f(W, H) := \frac{1}{|\mathcal{D}|} \sum_{(n,m,y) \in \mathcal{D}} \ell(y, W_{n,\cdot} H_{\cdot,m}) + \lambda(\|W\|_2^2 + \|H\|_2^2)$$

# The Matrix Factorization Model

- ▶ the basic model:

$$\hat{Y} := WH, \quad W \in \mathbb{R}^{N \times K}, H \in \mathbb{R}^{K \times M}$$

$$\text{i.e., } \hat{Y}_{n,m} := W_{n,\cdot} H_{\cdot,m}$$

- ▶  $K$  is called **latent dimension**
- ▶ parameters are **regularized**, i.e., minimize

$$f(W, H) := \frac{1}{|\mathcal{D}|} \sum_{(n,m,y) \in \mathcal{D}} \ell(y, W_{n,\cdot} H_{\cdot,m}) + \lambda(\|W\|_2^2 + \|H\|_2^2)$$

- ▶ usually a global offset  $a$  and **bias terms** are used, i.e., fix

$$W_{n,1} := 1, \quad H_{2,m} := 1$$

$$\text{yielding } \hat{Y}_{n,m} = a + W_{n,2} + H_{1,m} + W_{n,3:K} H_{3:K,m}$$



# Problem Equivalence

- ▶ The matrix completion problem really is a prediction problem with

$$\mathcal{X} := \{0, 1\}^N \times \{0, 1\}^M, \quad \mathcal{Y} := \mathbb{R}$$

# Stochastic Gradient Descent for Matrix Factorization

$$\begin{aligned} f(W, H) &:= \frac{1}{|\mathcal{D}|} \sum_{(n,m,y) \in \mathcal{D}} \ell(y, W_{n,\cdot}, H_{\cdot,m}) + \lambda(\|W\|_2^2 + \|H\|_2^2) \\ &\propto \sum_{(n,m,y) \in \mathcal{D}} (\ell(y, W_{n,\cdot}, H_{\cdot,m}) + \lambda(\|W\|_2^2 + \|H\|_2^2)) \end{aligned}$$

# Stochastic Gradient Descent for Matrix Factorization

$$\begin{aligned}
 f(W, H) &:= \frac{1}{|\mathcal{D}|} \sum_{(n,m,y) \in \mathcal{D}} \ell(y, W_{n,\cdot}, H_{\cdot,m}) + \lambda(\|W\|_2^2 + \|H\|_2^2) \\
 &\propto \sum_{(n,m,y) \in \mathcal{D}} (\ell(y, W_{n,\cdot}, H_{\cdot,m}) + \lambda(\|W\|_2^2 + \|H\|_2^2)) \\
 &= \sum_{(n,m,y) \in \mathcal{D}} (\ell(y, W_{n,\cdot}, H_{\cdot,m}) + \lambda(\frac{|\mathcal{D}|}{\text{freq}^1(\mathcal{D}, n)} \|W_{n,\cdot}\|_2^2 \\
 &\quad + \frac{|\mathcal{D}|}{\text{freq}^2(\mathcal{D}, m)} \|H_{\cdot,m}\|_2^2))
 \end{aligned}$$

with  $\text{freq}^1(\mathcal{D}, n) := |\{(n', m', y) \in \mathcal{D} \mid n' = n\}|$ ,

$\text{freq}^2(\mathcal{D}, m) := |\{(n', m', y) \in \mathcal{D} \mid m' = m\}|$

# Stochastic Gradient Descent for Matrix Factorization

$$\begin{aligned}
 f(W, H) &:= \frac{1}{|\mathcal{D}|} \sum_{(n,m,y) \in \mathcal{D}} \ell(y, W_{n,\cdot}, H_{\cdot,m}) + \lambda(\|W\|_2^2 + \|H\|_2^2) \\
 &\propto \sum_{(n,m,y) \in \mathcal{D}} (\ell(y, W_{n,\cdot}, H_{\cdot,m}) + \lambda(\|W\|_2^2 + \|H\|_2^2)) \\
 &= \sum_{(n,m,y) \in \mathcal{D}} (\ell(y, W_{n,\cdot}, H_{\cdot,m}) + \lambda \left( \frac{|\mathcal{D}|}{\text{freq}^1(\mathcal{D}, n)} \|W_{n,\cdot}\|_2^2 \right. \\
 &\quad \left. + \frac{|\mathcal{D}|}{\text{freq}^2(\mathcal{D}, m)} \|H_{\cdot,m}\|_2^2 \right)) \\
 &= \sum_{(n,m,y) \in \mathcal{D}} (\ell(y, w_n^T h_m) + \lambda_n^1 \|w_n\|_2^2 + \lambda_m^2 \|h_m\|_2^2)
 \end{aligned}$$

with  $\text{freq}^1(\mathcal{D}, n) := |\{(n', m', y) \in \mathcal{D} \mid n' = n\}|$ ,  $\lambda_n^1 := \lambda |\mathcal{D}| / \text{freq}^1(\mathcal{D}, n)$

$\text{freq}^2(\mathcal{D}, m) := |\{(n', m', y) \in \mathcal{D} \mid m' = m\}|$ ,  $\lambda_m^2 := \lambda |\mathcal{D}| / \text{freq}^2(\mathcal{D}, m)$

# Stochastic Gradient Descent for Matrix Factorization

$$\begin{aligned} f(W, H) &= \sum_{(n,m,y) \in \mathcal{D}} (\ell(y, w_n^T h_m) + \lambda_n^1 \|w_n\|_2^2 + \lambda_m^2 \|h_m\|_2^2) \\ &=: \sum_{(n,m,y) \in \mathcal{D}} f_{n,m,y}(w_n, h_m) \end{aligned}$$

$$\partial_{w_n} f_{n,m,y}(w_n, h_m) = \partial_{\hat{y}} \ell(y, w_n^T h_m) h_m + 2\lambda_n^1 w_n$$

# Stochastic Gradient Descent for Matrix Factorization

$$\begin{aligned}
 f(W, H) &= \sum_{(n,m,y) \in \mathcal{D}} (\ell(y, w_n^T h_m) + \lambda_n^1 \|w_n\|_2^2 + \lambda_m^2 \|h_m\|_2^2) \\
 &=: \sum_{(n,m,y) \in \mathcal{D}} f_{n,m,y}(w_n, h_m)
 \end{aligned}$$

$$\partial_{w_n} f_{n,m,y}(w_n, h_m) = \partial_{\hat{y}} \ell(y, w_n^T h_m) h_m + 2\lambda_n^1 w_n$$

$$\partial_{h_m} f_{n,m,y}(w_n, h_m) = \partial_{\hat{y}} \ell(y, w_n^T h_m) w_n + 2\lambda_m^2 h_m$$

# Stochastic Gradient Descent for Matrix Factorization

$$\begin{aligned}
 f(W, H) &= \sum_{(n,m,y) \in \mathcal{D}} (\ell(y, w_n^T h_m) + \lambda_n^1 \|w_n\|_2^2 + \lambda_m^2 \|h_m\|_2^2) \\
 &=: \sum_{(n,m,y) \in \mathcal{D}} f_{n,m,y}(w_n, h_m)
 \end{aligned}$$

$$\partial_{w_n} f_{n,m,y}(w_n, h_m) = \partial_{\hat{y}} \ell(y, w_n^T h_m) h_m + 2\lambda_n^1 w_n$$

$$\partial_{h_m} f_{n,m,y}(w_n, h_m) = \partial_{\hat{y}} \ell(y, w_n^T h_m) w_n + 2\lambda_m^2 h_m$$

The derivative of the loss needs to be computed once, e.g., for

$$\begin{aligned}
 \ell(y, \hat{y}) &:= (\hat{y} - y)^2 \\
 \rightsquigarrow \partial_{\hat{y}} \ell(y, \hat{y}) &= 2(\hat{y} - y)
 \end{aligned}$$

# Stochastic Gradient Descent for Matrix Factorization

```

1 sgd-mf( $\partial_{\hat{y}}\ell, N, M, \mathcal{D}, K, \lambda, T, \eta$ ):
2    $a := \frac{1}{|\mathcal{D}|} \sum_{(n,m,y) \in \mathcal{D}} y$ 
3   randomly initialize  $W \in \mathbb{R}^{N \times K}$  and  $H \in \mathbb{R}^{M \times K}$ 
4    $W_{n,1} := 1$  for all  $n := 1, \dots, N$ 
5    $H_{m,2} := 1$  for all  $m := 1, \dots, M$ 
6    $\lambda_n^1 := 2\lambda \frac{|\mathcal{D}|}{|\{(n',m',y) \in \mathcal{D} | n'=n\}|}$  for all  $n := 1, \dots, N$ 
7    $\lambda_m^2 := 2\lambda \frac{|\mathcal{D}|}{|\{(n',m',y) \in \mathcal{D} | m'=m\}|}$  for all  $m := 1, \dots, M$ 
8
9   for  $t = 1, \dots, T$ :
10    for  $(n, m, y) \in \mathcal{D}$  (in random order):
11       $e := \partial_{\hat{y}}\ell(y, a + w_n^T h_m)$ 
12       $w_n := w_n - \eta(eh_m + \lambda_n^1 w_n)$ 
13       $h_m := h_m - \eta(ew_n + \lambda_m^2 h_m)$ 
14       $w_{n,1} := 1, h_{m,2} := 1$ 
15
16   return  $(a, W, H)$ 
  
```



# Outline

1. Introduction

2. Matrix Factorization via Distributed SGD

3. NOMAD

# Idea

- ▶ partition the data into row subsets  $R_1, \dots, R_P$ , i.e.,

$$\mathcal{D}^P := \mathcal{D}|_{R_p} := \{(n, m, y) \in \mathcal{D} \mid n \in R_p\}$$

across  $P$  workers

- ▶ avoid conflicting distributed SGD updates by
  - ▶ also partitioning the columns into  $P$  subsets  $C_1, \dots, C_P$
  - ▶ for each epoch, make  $P$  passes over the data (at each worker), in every pass working on each worker on a different column subset, making sure every column subset finally is worked on every worker

$$W|_{R_p} := (W_{r,k})_{r \in R_p, k \in \{1, \dots, K\}}$$

$$H|_{C_q} := (H_{c,k})_{c \in C_q, k \in \{1, \dots, K\}}$$

$$\mathcal{D}|_{C_q}^P := \{(n, m, y) \in \mathcal{D}^P \mid m \in C_q\}$$



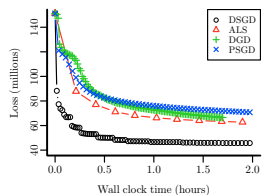
# Distributed SGD for Matrix Factorization

```

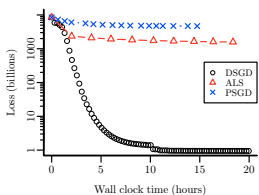
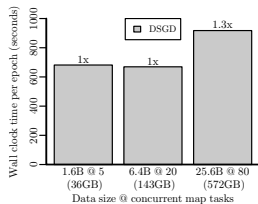
1 sgd-dsgd( $\partial_{\hat{y}} \ell, N, M, \mathcal{D}, K, \lambda, T, \eta, R, C$ ):
2   for  $p \in \{1, \dots, P\}$  (in parallel):
3     randomly initialize  $W^p \in \mathbb{R}^{R_p \times K}$ 
4      $W_{n,1}^p := 1$  for all  $n \in R_p$ 
5      $a^p := \sum_{(n,m,y) \in \mathcal{D}^p} y$ 
6     push  $a^p$  to server
7
8    $a := \frac{1}{|\mathcal{D}|} \sum_{p=1}^P a^p$ 
9   randomly initialize  $H \in \mathbb{R}^{M \times K}$ 
10   $H_{m,2} := 1$  for all  $m := 1, \dots, M$ 
11
12  for  $t = 1, \dots, T$ :
13    for  $q = 1, \dots, P$ :
14      for  $p := 1, \dots, P$  in parallel:
15         $q^p := 1 + (q + p - 2 \bmod P)$ 
16        pop  $C^p := C_{q^p}, H^p := H_{|C_{q^p}}$  from server
17        sgd-mf-update( $\partial_{\hat{y}} \ell, R_p, C^p, \mathcal{D}_{|C^p}^p, K, \lambda, T = 1, \eta, a; W^p, H^p$ )
18        push  $H_{|C_{q^p}} := H^p$  to server
19
20  for  $p := 1, \dots, P$  in parallel:
21    push  $W_{|R_p} := W^p$  to server
22
23  return  $(a, W, H)$ 

```

# Experiments / Results



(a) Netflix data (NZSL, R cluster @ 64)

(b) Synth. data (L2,  $\lambda = 0.1$ , R cluster @ 64)

(c) Scalability (Hadoop cluster)

Figure 2: Experimental results

[source: Gemulla et al. [2011]]

# Experiments / Results

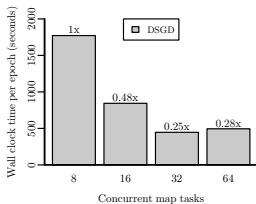


Figure 3: Speed-up experiment (Hadoop cluster, 143GB data)

[source: Gemulla et al. [2011]]

# Outline

1. Introduction

2. Matrix Factorization via Distributed SGD

3. NOMAD

# Parameter Updates

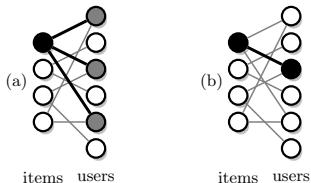


Figure 1: Illustration of updates used in matrix completion. Three algorithms are shown here: (a) alternating least squares and coordinate descent, (b) stochastic gradient descent. Black indicates that the value of the node is being updated, gray indicates that the value of the node is being read. White nodes are neither being read nor updated.

[source: Yun et al. [2014]]



# Parameter Partitioning

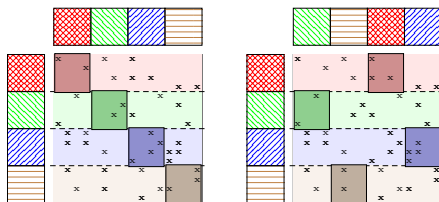
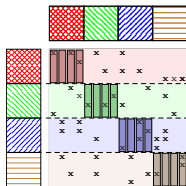


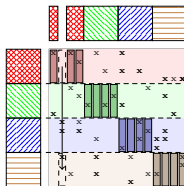
Figure 3: Illustration of DSGD algorithm with 4 workers. Initially  $W$  and  $H$  are partitioned as shown on the left. Each worker runs SGD on its active area as indicated. After each worker completes processing data points in its own active area, the columns of item parameters  $H^\top$  are exchanged randomly, and the active area changes. This process is repeated for each iteration.

[source: Yun et al. [2014]]

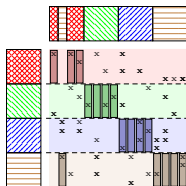
# Parameter Partitioning



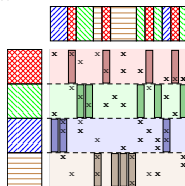
(a) Initial assignment of  $W$  and  $H$ . Each worker works only on the diagonal active area in the beginning.



(b) After a worker finishes processing column  $j$ , it sends the corresponding item parameter  $h_j$  to another worker. Here,  $h_2$  is sent from worker 1 to 4.

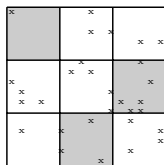


(c) Upon receipt, the column is processed by the new worker. Here, worker 4 can now process column 2 since it owns the column.

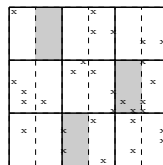


(d) During the execution of the algorithm, the ownership of the item parameters  $h_j$  changes.

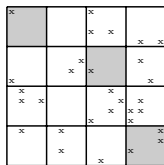
# Parameter Partitioning



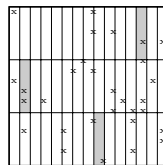
(a) DSGD



(b) DSGD++



(c) FPSGD\*\*



(d) NOMAD

Figure 4: Comparison of data partitioning schemes between algorithms. Example active area of stochastic gradient sampling is marked as gray.

# Algorithm

---

**Algorithm 1** the basic NOMAD algorithm
 

---

```

1:  $\lambda$ : regularization parameter
2:  $\{s_t\}$ : step size sequence
3: // initialize parameters
4:  $w_{il} \sim \text{UniformReal}\left(0, \frac{1}{\sqrt{k}}\right)$  for  $1 \leq i \leq m, 1 \leq l \leq k$ 
5:  $h_{jl} \sim \text{UniformReal}\left(0, \frac{1}{\sqrt{k}}\right)$  for  $1 \leq j \leq n, 1 \leq l \leq k$ 
6: // initialize queues
7: for  $j \in \{1, 2, \dots, n\}$  do
8:    $q \sim \text{UniformDiscrete}\{1, 2, \dots, p\}$ 
9:    $\text{queue}[q].\text{push}((j, \mathbf{h}_j))$ 
10: end for
11: // start  $p$  workers
12: Parallel Foreach  $q \in \{1, 2, \dots, p\}$ 
13:   while stop signal is not yet received do
14:     if  $\text{queue}[q]$  not empty then
15:        $(j, \mathbf{h}_j) \leftarrow \text{queue}[q].\text{pop}()$ 
16:       for  $(i, j) \in \bar{\Omega}_j^{(q)}$  do
17:         // SGD update
18:          $t \leftarrow$  number of updates on  $(i, j)$ 
19:          $\mathbf{w}_i \leftarrow \mathbf{w}_i - s_t \cdot [(A_{ij} - \mathbf{w}_i \mathbf{h}_j) \mathbf{h}_j + \lambda \mathbf{w}_i]$ 
20:          $\mathbf{h}_j \leftarrow \mathbf{h}_j - s_t \cdot [(A_{ij} - \mathbf{w}_i \mathbf{h}_j) \mathbf{w}_j + \lambda \mathbf{h}_j]$ 
21:       end for
22:        $q' \sim \text{UniformDiscrete}\{1, 2, \dots, p\}$ 
23:        $\text{queue}[q'].\text{push}((j, \mathbf{h}_j))$ 
24:     end if
25:   end while
26: Parallel End
  
```

---

# Results

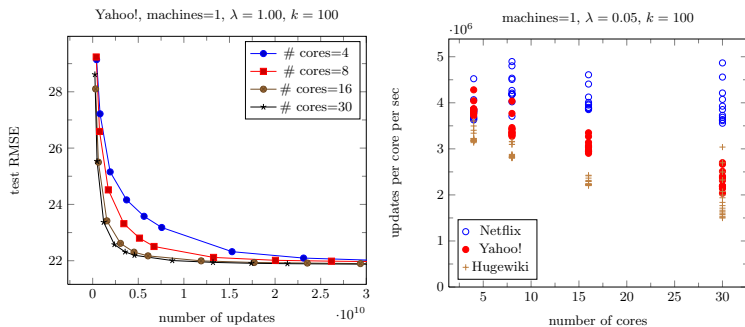


Figure 6: Left: Test RMSE of NOMAD as a function of the number of updates on Yahoo! Music, when the number of cores is varied. Right: Number of updates of NOMAD per core per second as a function of the number of cores.

## Results / Comparison with Baselines

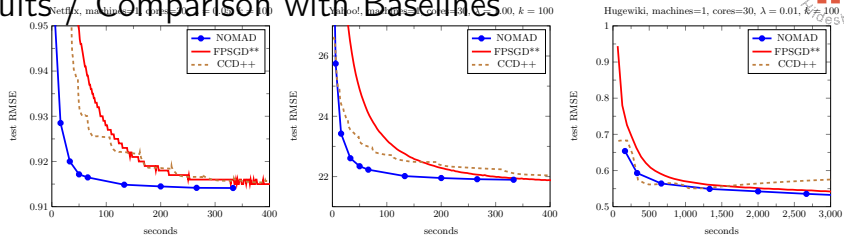
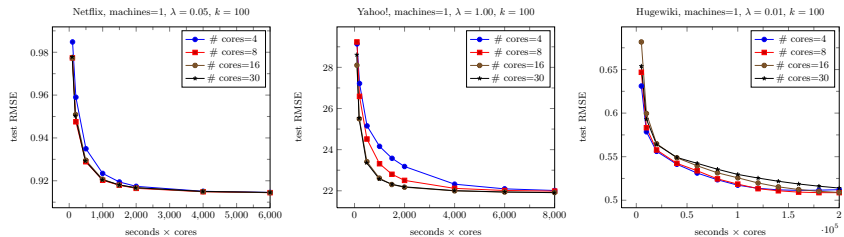


Figure 5: Comparison of NOMAD, FPSGD\*\*, and CCD++ on a single-machine with 30 computation cores.

Figure 7: Test RMSE of NOMAD as a function of computation time (time in seconds  $\times$  the number of cores), when the number of cores is varied.

## Results

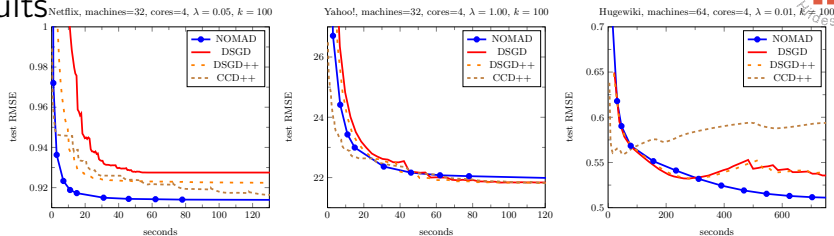
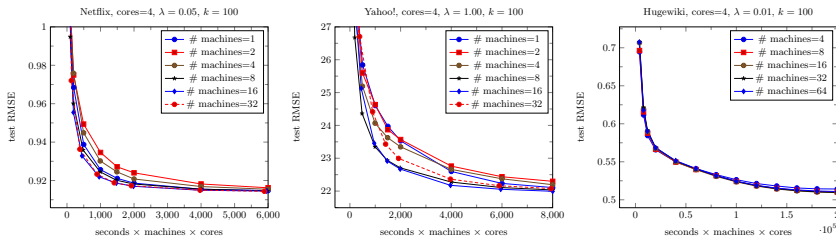


Figure 8: Comparison of NOMAD, DSGD, DSGD++, and CCD++ on a HPC cluster.

Figure 9: Test RMSE of NOMAD as a function of computation time (time in seconds  $\times$  the number of machines  $\times$  the number of cores per each machine) on a HPC cluster, when the number of machines is varied.

# Results

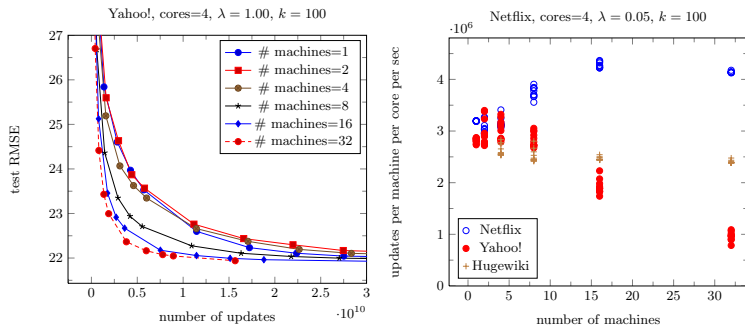


Figure 10: Results on HPC cluster when the number of machines is varied. Left: Test RMSE of NOMAD as a function of the number of updates on Netflix and Yahoo! Music. Right: Number of updates of NOMAD per machine per core per second as a function of the number of machines.



# Results

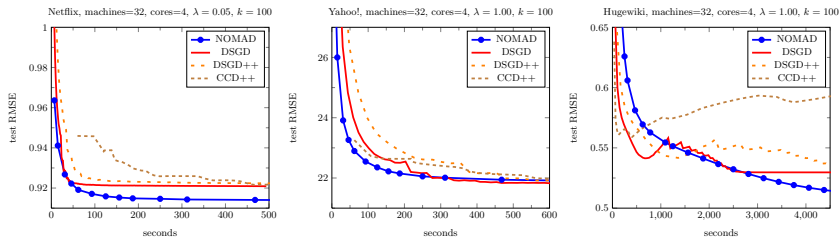


Figure 11: Comparison of NOMAD, DSGD, DSGD++, and CCD++ on a commodity hardware cluster.

[source: Yun et al. [2014]]

# Results

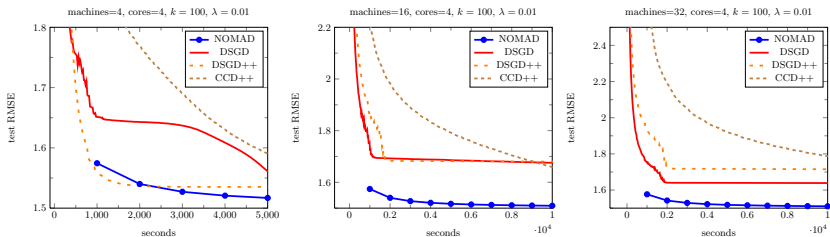


Figure 12: Comparison of algorithms when both dataset size and the number of machines grows. Left: 4 machines, middle: 16 machines, right: 32 machines

[source: Yun et al. [2014]]

## Summary (1/2)

- ▶ The **Matrix Completion Problem** is about predicting the non-observed cells in a partially observed matrix.
  - ▶ It is equivalent to a regression problem with only two nominal predictors:
    - ▶ row index
    - ▶ column index
- ▶ **Matrix Factorization Models** usually are used to solve matrix completion problems.
  - ▶ associating **latent features** to each nominal row/column index
  - ▶ global, per row and per column **bias terms**
- ▶ **Stochastic Gradient Descent** can be used to learn matrix factorization models.
- ▶ Any distributed SGD algorithm from the last lecture could be used to learn the model on distributed data.

## Summary (2/2)

- ▶ When data is partitioned by rows, latent row features can stay at the host where the respective data is.
- ▶ **Distributed SGD for Matrix Factorization (DSGD):**
  - ▶ also partitions the latent column features in  $P$  blocks
  - ▶ assigns them in a round robin fashion to all  $P$  data nodes
  - ▶ avoids conflicting parallel updates
  - ▶ nodes with little data for a column partition have to wait for nodes having larger data for their column partition
- ▶ **NOMAD:**
  - ▶ partitions the latent column features in  $M$  blocks (one for each column)
  - ▶ assigns them in a random fashion to all data nodes
  - ▶ avoids conflicting parallel updates
  - ▶ nodes with little data for a column can proceed with other columns while nodes with larger data for their column run in parallel

# References I

- Rainer Gemulla, Erik Nijkamp, Peter J. Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77. ACM, 2011. URL <http://dl.acm.org/citation.cfm?id=2020426>.
- Hyokun Yun, Hsiang-Fu Yu, Cho-Jui Hsieh, S. V. N. Vishwanathan, and Inderjit Dhillon. NOMAD: Non-locking, stOchastic Multi-machine algorithm for Asynchronous and Decentralized matrix completion. *Proceedings of the VLDB Endowment*, 7(11):975–986, 2014. URL <http://dl.acm.org/citation.cfm?id=2732973>.