Big Data Analytics

9. Distributed Matrix Factorization

Lars Schmidt-Thieme

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

Outline

Outline

The Matrix Completion Problem

Given

- the values $\mathcal{D} \subseteq [N] \times [M] \times \mathbb{R}$ of some cells of an unknown matrix $Y \in \mathbb{R}^{N \times M}$ 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: $[N] := \{1, \ldots, N\}$.

The Matrix Factorization Model

- the basic model:

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

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

  - K is called **latent dimension**

The Matrix Factorization Model

- the basic model:

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

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

  - K is called **latent dimension**

- parameters are regularized, i.e., minimize

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

The Matrix Factorization Model

- the basic model:

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

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

  - K is called **latent dimension**

- parameters are regularized, i.e., minimize

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

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

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

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

$$= a + b_n + c_m + w_n^T h_m$$

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

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

$$\propto \sum_{(n,m,y)\in\mathcal{D}} \left(\ell(y, W_{n,.} H_{.,m}) + \lambda(||W||_2^2 + ||H||_2^2)\right)$$

Stochastic Gradient Descent for Matrix Factorization

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

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

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

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

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

Stochastic Gradient Descent for Matrix Factorization

$$f(W,H) = \sum_{(n,m,y)\in\mathcal{D}} \left(\ell(y, w_n^T h_m) + \lambda_n^1 ||w_n||_2^2 + \lambda_m^2 ||h_m||_2^2\right)$$

$$=: \sum_{(n,m,y)\in\mathcal{D}} f_{n,m,y}(w_n, h_m)$$

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

Stochastic Gradient Descent for Matrix Factorization

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

$$=: \sum_{(n,m,y)\in\mathcal{D}} f_{n,m,y}(w_n, h_m)$$

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

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

Stochastic Gradient Descent for Matrix Factorization

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

$$=: \sum_{(n,m,y)\in\mathcal{D}} f_{n,m,y}(w_n, h_m)$$

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

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

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

$$\ell(y, \hat{y}) := (y - \hat{y})^2$$

$$\rightsquigarrow \partial_{\hat{y}} \ell(y, \hat{y}) = 2(y - \hat{y})$$

Stochastic Gradient Descent for Matrix Factorization

sgd-mf($\partial_{\hat{y}}\ell, N, M, \mathcal{D}, K, \lambda, T, \eta$):

  $a := \frac{1}{|\mathcal{D}|} \sum_{(n,m,y)\in\mathcal{D}} y$

  randomly initialize $W \in \mathbb{R}^{N\times K}$ and $H \in \mathbb{R}^{M\times K}$

  $W_{n,1} := 1$ for all $n := 1, \ldots, N$

  $H_{m,2} := 1$ for all $m := 1, \ldots, M$

  $\lambda^1_n := 2\lambda \frac{|D|}{|\{(n',m',y)\in\mathcal{D} \mid n'=n\}|}$ for all $n := 1, \ldots, N$

  $\lambda^2_m := 2\lambda \frac{|D|}{|\{(n',m',y)\in\mathcal{D} \mid m'=m\}|}$ for all $m := 1, \ldots, M$

  for $t = 1, \ldots, T$:

    for $(n, m, y) \in \mathcal{D}$ (in random order):

      $e := \partial_{\hat{y}}\ell(y, a + w_n^T h_m)$

      $w_n := w_n - \eta(e h_m + \lambda^1_n w_n)$

      $h_m := h_m - \eta(e w_n + \lambda^2_m h_m)$

      $w_{n,1} := 1, h_{m,2} := 1$

  return (a, W, H)

Outline

Idea

- partition the data into row subsets $R_1, \ldots, R_P$, 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 $C_1, \ldots, C_P$
  - 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

$$W_{|R_p} := (W_{r,k})_{r \in R_p, k \in \{1,\ldots,K\}}$$
$$H_{|C_q} := (H_{c,k})_{c \in C_q, k \in \{1,\ldots,K\}}$$
$$\mathcal{D}^p_{|C_q} := \{(n, m, y) \in \mathcal{D}^p \mid m \in C_q\}$$

Distributed SGD for Matrix Factorization

sgd-dsgd($\partial_{\hat{y}}\ell, N, M, \mathcal{D}, K, \lambda, T, \eta, R, C$):
    for $p \in \{1, \dots, P\}$ (in parallel):
        randomly initialize $W^p \in \mathbb{R}^{R_p \times K}$
        $W^p_{n,1} := 1$ for all $n \in R_p$
        $a^p := \sum_{(n,m,y)\in\mathcal{D}^p} y$
        push a^p to server

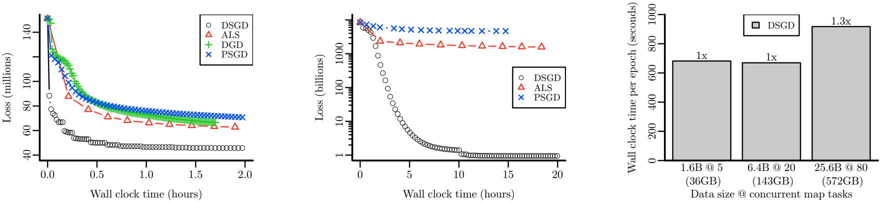
    $a := \frac{1}{|\mathcal{D}|}\sum_{p=1}^{P} a^p$
    randomly initialize $H \in \mathbb{R}^{M \times K}$
    $H_{m,2} := 1$ for all $m := 1, \dots, M$

    for $t = 1, \dots, T$:
        for $q = 1, \dots, P$:
            $q^p := 1 + (q + p - 2 \bmod P)$ for $p = 1, \dots, P$
            for $p := 1, \dots, P$ in parallel:
                pop $C^p := C_{q^p}, H^p := H_{|C_{q^p}}$ from server
                sgd-mf-update($\partial_{\hat{y}}\ell, R_p, C^p, \mathcal{D}^p_{|C^p}, K, \lambda, T = 1, \eta, a; W^p, H^p$)
                push $H_{|C_{q^p}} := H^p$ to server

    for $p := 1, \dots, P$ in parallel:
        push $W_{|R_p} := W^p$ to server

    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

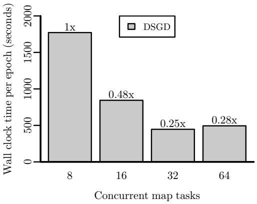


Figure 3: Speed-up experiment (Hadoop cluster, 143GB data)

[source: Gemulla et al. [2011]]

Outline

...

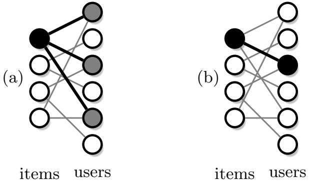
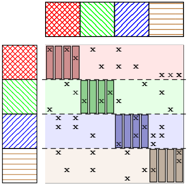


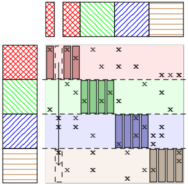
Figure 1: Illustration of updates used in matrix completion. Three algorithms are shown here: (a) alternating least squares and coordinate descent, (b) stochastic gradient descent. Black indicates that the value of the node is being updated, gray indicates that the value of the node is being read. White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

[source: Yun et al. [2014]]

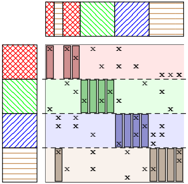
...



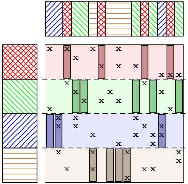
(a) Initial assignment of W and H. Each worker works only on the diagonal active area in the beginning.



(b) After a worker finishes processing column j, it sends the corresponding item parameter $\mathbf{h}_j$ to another worker. Here, $\mathbf{h}_2$ 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 $\mathbf{h}_j$ changes.

Algorithm

Algorithm 1 the basic NOMAD algorithm

λ: regularization parameter
$\{s_t\}$: step size sequence
`// initialize parameters`
$w_{il} \sim \text{UniformReal}\left(0, \frac{1}{\sqrt{k}}\right)$ for $1 \le i \le m, 1 \le l \le k$
$h_{jl} \sim \text{UniformReal}\left(0, \frac{1}{\sqrt{k}}\right)$ for $1 \le j \le n, 1 \le l \le k$
`// initialize queues`
for $j \in \{1, 2, \ldots, n\}$ **do**
  $q \sim \text{UniformDiscrete}\{1, 2, \ldots, p\}$
  `queue`$[q]$`.push`$((j, \mathbf{h}_j))$
end for
`// start` p `workers`
Parallel Foreach $q \in \{1, 2, \ldots, p\}$
  while stop signal is not yet received **do**
    if `queue`$[q]$ not empty **then**
      $(j, \mathbf{h}_j) \leftarrow$ `queue`$[q]$`.pop()`
      for $(i, j) \in \bar{\Omega}_j^{(q)}$ **do**
        `// SGD update`
        $t \leftarrow$ number of updates on (i, j)
        $\mathbf{w}_i \leftarrow \mathbf{w}_i - s_t \cdot [(A_{ij} - \mathbf{w}_i \mathbf{h}_j)\mathbf{h}_j + \lambda \mathbf{w}_i]$
        $\mathbf{h}_j \leftarrow \mathbf{h}_j - s_t \cdot [(A_{ij} - \mathbf{w}_i \mathbf{h}_j)\mathbf{w}_j + \lambda \mathbf{h}_j]$.
      end for
      $q' \sim \text{UniformDiscrete}\{1, 2, \ldots, p\}$
      `queue`$[q']$`.push`$((j, \mathbf{h}_j))$
    end if
  end while
Parallel End

...

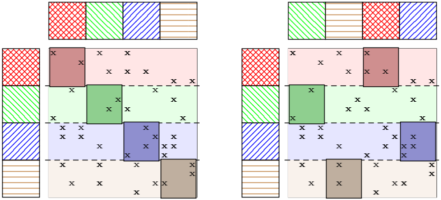
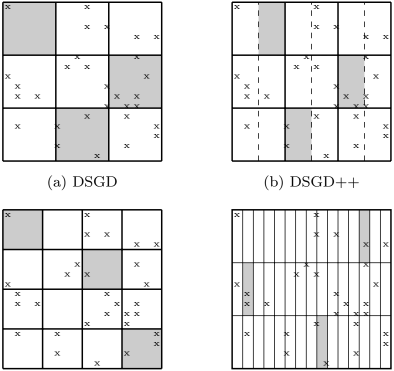


Figure 3: Illustration of DSGD algorithm with 4 workers. Initially W and H are partitioned as shown on the left. Each worker runs SGD on its active area as indicated. After each worker completes processing data points in its own active area, the columns of item parameters $H^\top$ are exchanged randomly,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

[source: Yun et al. [2014]]

...



(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.

[source: Yun et al. [2014]]

...

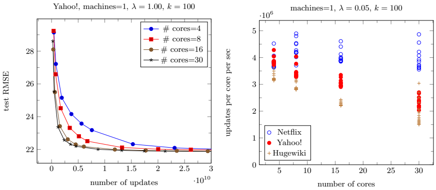


Figure 6: Left: Test RMSE of NOMAD as a function of the number of updates on Yahoo! Music, when the number of cores is varied. Right: Number of updates of NOMAD per core per second as a function of the number of cores.

[source: Yun et al. [2014]]

...

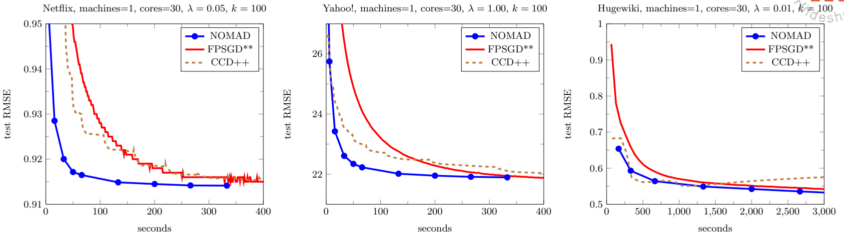


Figure 5: Comparison of NOMAD, FPSGD**, and CCD++ on a single-machine with 30 computation cores.

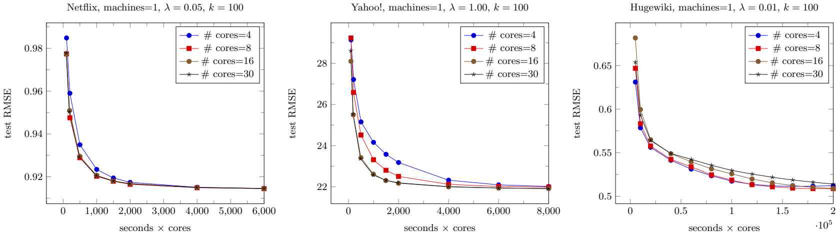


Figure 7: Test RMSE of NOMAD as a function of computation time (time in seconds × the number of cores), when the number of cores is varied.

...

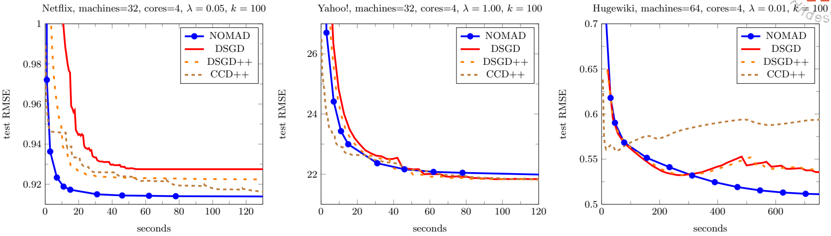


Figure 8: Comparison of NOMAD, DSGD, DSGD++, and CCD++ on a HPC cluster.

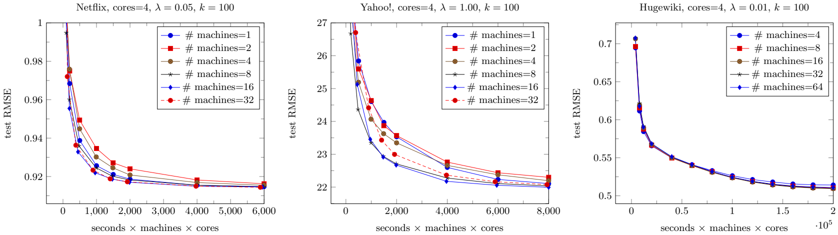


Figure 9: Test RMSE of NOMAD as a function of computation time (time in seconds × the number of machines × the number of cores per each machine) on a HPC cluster, when the number of machines is varied.

...

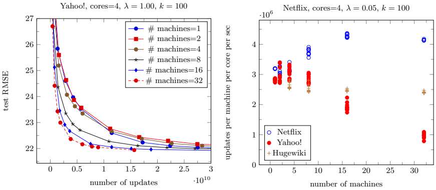


Figure 10: Results on HPC cluster when the number of machines is varied. Left: Test RMSE of NOMAD as a function of the number of updates on Netflix and Yahoo! Music. Right: Number of updates of NOMAD per machine per core per second as a function of the number of machines.

[source: Yun et al. [2014]]

...

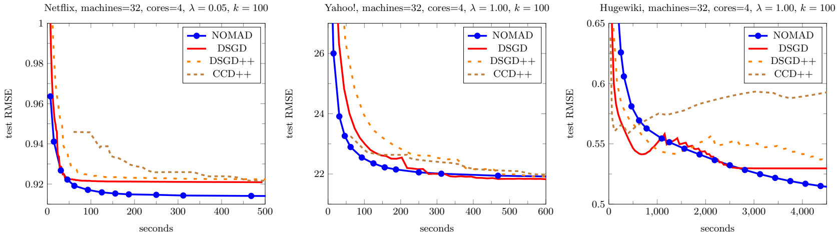


Figure 11: Comparison of NOMAD, DSGD, DSGD++, and CCD++ on a commodity hardware cluster.

[source: Yun et al. [2014]]

...

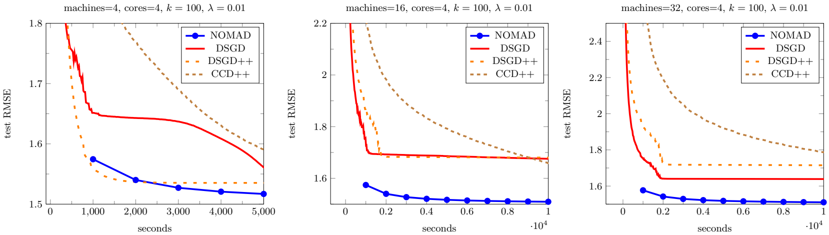


Figure 12: Comparison of algorithms when both dataset size and the number of machines grows. Left: 4 machines, middle 16 machines, right: 32 machines

[source: Yun et al. [2014]]

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`.