Large-Scale Distributed Bayesian Matrix Factorization using Stochastic Gradient MCMC

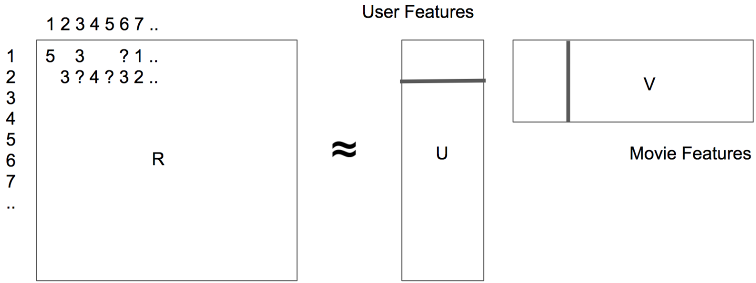
Sungjin Ahn, Anoop Korattikara, Nathan Liu,
Suju Rajan, Max Welling

May 30, 2017

Bayesian MF Literature

Recap: Bayesian PMF

PMF:

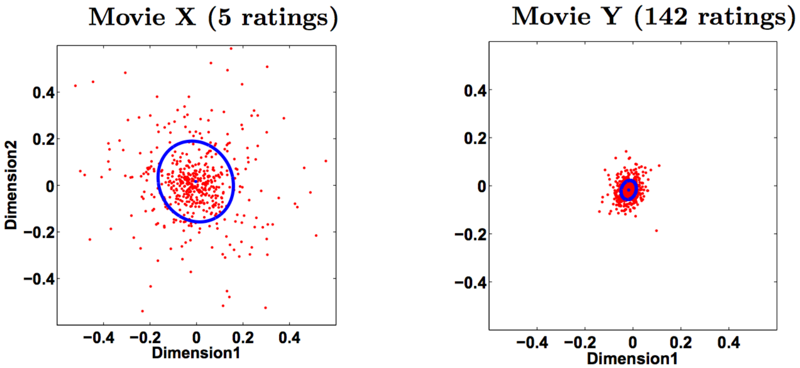


- Rating Matrix: $R \in \mathbb{R}^{L\times M}$ (# users = L, # items = M)
- User latent features: $U \in \mathbb{R}^{D\times L}$
- User latent features: $V \in \mathbb{R}^{D\times M}$

Recap: Bayesian PMF

PMF → Bayesian PMF

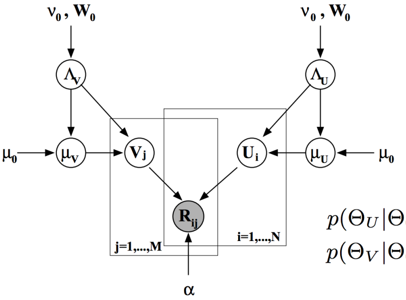
- In reality, R is very sparse!
- MLE estimation for U and V lead to severe over-fitting
- We want to model the uncertainty in the items



Recap: Bayesian PMF

Model

- Introduce priors for the parameters
- Allow model complexity to be controlled automatically



$$p(\mathbf{U}|\mu_U, \Lambda_U) = \prod_{i=1}^{L} \mathcal{N}(U_i|\mu_U, \Lambda_U^{-1}),$$

$$p(\mathbf{V}|\mu_V, \Lambda_V) = \prod_{j=1}^{M} \mathcal{N}(V_j|\mu_V, \Lambda_V^{-1}).$$

$$p(\Theta_U|\Theta_0) = \mathcal{N}(\mu_U|\mu_0, (\beta_0\Lambda_U)^{-1})\mathcal{W}(\Lambda_U|W_0, \nu_0),$$

$$p(\Theta_V|\Theta_0) = \mathcal{N}(\mu_V|\mu_0, (\beta_0\Lambda_V)^{-1})\mathcal{W}(\Lambda_V|W_0, \nu_0),$$

Recap: Bayesian PMF

Inference

- predictive distribution:

$$p(R_{ij}^*|\mathbf{R},\Theta_0) = \int\int p(R_{ij}^*|\underline{U_i,V_j)p(\mathbf{U},\mathbf{V}}|\mathbf{R},\Theta_U,\Theta_V) \\ p(\Theta_U,\Theta_V|\Theta_0)\mathrm{d}\{\mathbf{U},\mathbf{V}\}\mathrm{d}\{\Theta_U,\Theta_V\}$$

- MCMC

$$p(R_{ij}^*|\mathbf{R},\Theta_0) \approx \frac{1}{T}\sum_{t=1}^{T} p(R_{ij}^*|U_i^{(t)},V_j^{(t)}).$$

where $\{U_i^t, V_j^t\}$ is generated by a markov-chain whose stationary distribution is the posterior distribution over the model parameters:

$$p(\mathbf{U},\mathbf{V},\Theta_U,\Theta_V|\mathbf{R},\Theta_0) \\ = p(\mathbf{U},\mathbf{V}|\mathbf{R},\Theta_U,\Theta_V)\ p(\Theta_U,\Theta_V|\Theta_0)$$

Recap: Bayesian PMF

Inference

Algorithm 1 Gibbs Sampling for BPMF

Initialize model parameters $\mathbf{U}^{(1)}, \mathbf{V}^{(1)}$

for $t = 1 : T$ **do**

// Sample hyperparameters

$\Theta_U^{(t)} \sim p(\Theta_U | \mathbf{U}^{(t)}, \Theta_0), \quad \Theta_V^{(t)} \sim p(\Theta_V | \mathbf{V}^{(t)}, \Theta_0)$

for $i = 1 : L$ **in parallel do**

$U_i^{(t+1)} \sim p(U_i | \mathbf{R}, \mathbf{V}^{(t)}, \Theta_U^{(t)})$ // sample user features

end for

for $j = 1 : M$ **in parallel do**

$V_j^{(t+1)} \sim p(V_j | \mathbf{R}, \mathbf{U}^{(t)}, \Theta_V^{(t)})$ // sample item features

end for

end for

Recap: Bayesian PMF

Inference

- Sampling from conditionals for U_i, V_j in Gibbs, involve $O(D^3)$ computation per iteration (inverting $D \times D$ precision matrix)

$$p(U_i|R, V, \Theta_U, \alpha) = \mathcal{N}\left(U_i|\mu_i^*, \left[\Lambda_i^*\right]^{-1}\right)$$
$$\sim \prod_{j=1}^{M} \left[\mathcal{N}(R_{ij}|U_i^T V_j, \alpha^{-1})\right]^{I_{ij}} p(U_i|\mu_U, \Lambda_U),$$

where

$$\Lambda_i^* = \Lambda_U + \alpha \sum_{j=1}^{M} \left[V_j V_j^T\right]^{I_{ij}}$$
$$\mu_i^* = [\Lambda_i^*]^{-1}\left(\alpha \sum_{j=1}^{M} \left[V_j R_{ij}\right]^{I_{ij}} + \Lambda_U \mu_U\right).$$

- Each iteration of MCMC requires computations over the entire dataset

Langevin Dynamics

$$\theta_{t+1} \leftarrow \theta_t + \frac{\epsilon_t}{2}\left(\nabla_{\theta_t} \log p(\theta_t) + \sum_{x \in X} \nabla_{\theta_t} \log p(x|\theta)\right) + \mathcal{N}(0, \epsilon_t)$$

- Converges to true posterior distribution if step size ϵ_t is annealed to 0 at a rate satisfying:
  $\sum_{t=1}^{\infty} \epsilon_t = \infty, \quad \sum_{t=1}^{\infty} \epsilon_t^2 < \infty$
- No accept-reject tests since acceptance rate $\to 1$ as $\epsilon_t \to 0$.
- Requires $O(N)$ computation per iteration.

Stochastic Gradient Langevin Dynamics (SGLD)

[Welling and Teh, 2011]

$$\theta_{t+1} \leftarrow \theta_t + \frac{\epsilon_t}{2}\left(\text{Stochastic Gradient}\right) + \mathcal{N}(0, \epsilon_t)$$

$$\theta_{t+1} \leftarrow \theta_t + \frac{\epsilon_t}{2}\left(\nabla_{\theta_t} \log p(\theta_t) + \frac{N}{m}\sum_{x\in\mathcal{M}} \nabla_{\theta_t} \log p(x|\theta)\right) + \mathcal{N}(0, \epsilon_t)$$

- O($m << N$) computation per iteration.

Bayesian PMF using SGLD

Model

$$
\begin{aligned}
p(\mathbf{U}|\Lambda_U) &= \prod_{i=1}^{L} \mathcal{N}(U_i|0, \Lambda_U^{-1}), \\
p(\mathbf{V}|\Lambda_V) &= \prod_{j=1}^{M} \mathcal{N}(V_j|0, \Lambda_V^{-1}).
\end{aligned}
$$

$$
\lambda_{U_d}, \lambda_{V_d} \sim \mathrm{Gamma}(\alpha_0, \beta_0).
$$

- Slightly simplified model
- Λ_U and Λ_V are D-dimensional diagonal matrices with diagonal elements λ_{U_d} and λ_{V_d} respectively
- Each latent vector can be updated in $\mathrm{O}(D)$ instead of $\mathrm{O}(D^2)$ in the full-covariance case

Bayesian PMF using SGLD

Inference

- Recall the full posterior:

$$p(\mathbf{U}, \mathbf{V}, \Theta_U, \Theta_V | \mathbf{R}, \Theta_0)$$
$$= p(\mathbf{U}, \mathbf{V} | \mathbf{R}, \Theta_U, \Theta_V) \; p(\Theta_U, \Theta_V | \Theta_0)$$

- Alternate b/w steps:
  - Step 1: Sample from $p(U, V | R, \Theta_U, \Theta_V)$ using SGLD
  - Step 2: Sample from $p(\Theta_U, \Theta_V | \Theta_0)$ using Gibbs

Bayesian PMF using SGLD

Step 1: Sample from $p(U, V|R, \Theta_U, \Theta_V)$ using SGLD

- Suppose rating matrix is: $\mathcal{X} = \{x_n = (p_n, q_n, r_n)\}_{n=1}^{N}$
- Gradient of log-posterior w.r.t U_i

$$G(\mathcal{X}) = \sum_{n=1}^{N} g_n(U_i; \mathcal{X}) - \Lambda_U U_i$$

$$g_n(U_i; \mathcal{X}) = \mathbb{I}[p_n = i|\mathcal{X}](r_n - U_{p_n}^{\top} V_{q_n}) V_{q_n}$$

- To use SGLD, need unbiased estimate of this gradient from a mini-batch $\mathcal{M} = \{(p_n, q_n, r_n)\}_{n=1}^{m}$ of m tuples of $\mathcal{X}$

$$G_1(\mathcal{M}) = N\bar{g}(U_i; \mathcal{M}) - \Lambda_U U_i$$

$$\bar{g}(U_i; \mathcal{M}) = \frac{1}{m} \sum_{n=1}^{m} g_n(U_i; \mathcal{M})$$

Bayesian PMF using SGLD

Step 1: Sample from $p(U, V|R, \Theta_U, \Theta_V)$ using SGLD

- Unbiased estimate of the gradient which needs only updating users in the mini-batch $\mathcal{M}$

$$G_3(\mathcal{M}) = N\bar{g}(U_i; \mathcal{M}) - \mathbb{I}[i \in \mathcal{M}_p] h_{i*}^{-1} \Lambda_U U_i.$$

$$h_{i*} = 1 - \left(1 - \frac{N_{i*}}{N}\right)^m$$

$$N_{i*} = \textstyle\sum_{n=1}^{N} \mathbb{I}[p_n = i|\mathcal{X}]$$

- Overall update for U_i and V_j (s is a worker)

$$U_{i,t+1} \leftarrow U_{i,t} + \frac{\epsilon_t}{2}\left\{\frac{N^{(s)}}{\upsilon^{(s)}}\bar{g}(U_{i,t}; \mathcal{M}_t^{(s)}) - \frac{\Lambda_U U_{i,t}}{\bar{h}_{i*}}\right\} + \nu_t \quad (29)$$

$$V_{j,t+1} \leftarrow V_{j,t} + \frac{\epsilon_t}{2}\left\{\frac{N^{(s)}}{\upsilon^{(s)}}\bar{g}(V_{j,t}; \mathcal{M}_t^{(s)}) - \frac{\Lambda_V V_{j,t}}{\bar{h}_{*j}}\right\} + \nu_t. \quad (30)$$

Bayesian PMF using SGLD

Mini-batch for Step 1

2	1	3
2	2	1
3	2	2
1	3	5
2	4	2
3	3	4
4	1	2
5	1	4
6	2	1
4	4	1
5	3	5
6	4	3

Bayesian PMF using SGLD

Step 2: Sample from $p\left(\Theta_U, \Theta_V | \Theta_0\right)$ using Gibbs

- This is easy because of conjugacy:

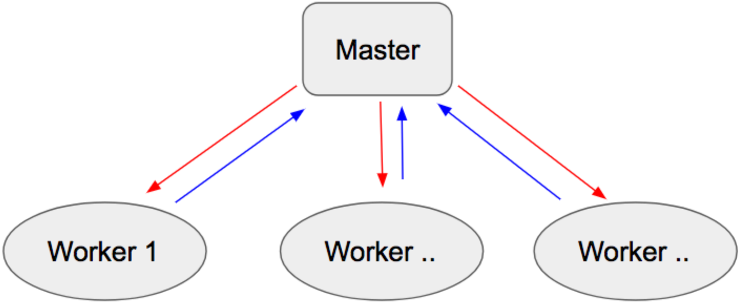
$$\lambda_{U_d}|\mathbf{U},\mathbf{V} \sim \text{Gamma}\left(\alpha_0 + \frac{L}{2}, \beta_0 + \frac{1}{2}\sum_{i=1}^{L} U_{di}^2\right), \tag{24}$$

$$\lambda_{V_d}|\mathbf{U},\mathbf{V} \sim \text{Gamma}\left(\alpha_0 + \frac{M}{2}, \beta_0 + \frac{1}{2}\sum_{i=1}^{M} V_{dj}^2\right). \tag{25}$$

Distributed SGLD

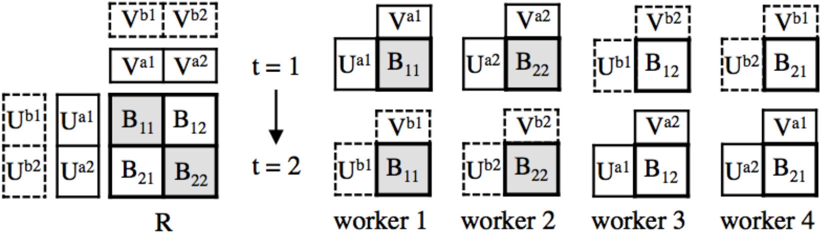
Parameter Server Architecture

- Worker s perform Step 1 - Sample from $p\left(U_s, V_s | R, \Theta_{U_s}, \Theta_{V_s}\right)$
- Master performs Step 2 - Sample from $p\left(\Theta_U, \Theta_V | \Theta_0\right)$



Distributed SGLD

Distribution of data (rating matrix R) and U, V among workers



- Rating matrix R is partitioned into mutually-exclusive blocks B_{ij}
- Run 2 independent chains with parameters $\{U^a, V^a\}$ and $\{U^b, V^b\}$

DSGLD: Algorithm (parameter server)

Algorithm 2 DSGLD at parameter server

Initialize model parameters of each chain $\{\mathbf{U}_1^c, \mathbf{V}_1^c, \Lambda_1^c\}_{c=1}^{C}$, step sizes $\{\epsilon_t\}$
for each chain c ***parallel*** **do**
    for t=1:max_iter **do**
        $\mathcal{B}_c \leftarrow$ GET_ORTHO_BLOCK_GROUP(c, t)
        for worker $s \in \mathcal{B}_c$ **do**
            $\mathbf{U}_{t+1}^{(c,s)}, \mathbf{V}_{t+1}^{(c,s)} \leftarrow$ WKR_ROUND$(\mathbf{U}_t^{(c,s)}, \mathbf{V}_t^{(c,s)}, \Lambda_t^{(c)}, \epsilon_t)$
        end for
        if not burn-in **then**
            Store $\mathbf{U}_{t+1}^{(c)}, \mathbf{V}_{t+1}^{(c)}$ as a sample of chain c
            Sample $\Lambda_{t+1}^{(c)} | \mathbf{U}_{t+1}^{(c)}, \mathbf{V}_{t+1}^{(c)}$ using Eqn. (24) and (25)
        end if
    end for
end for

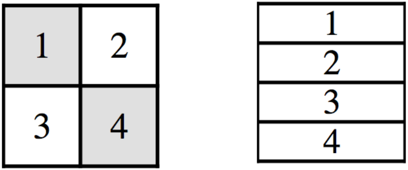
DSGLD: Algorithm (worker)

Algorithm 3 DSGLD at worker s

```
Initialize h̄_{i*}, h̄_{*j}, round length γ, mini-batch size m
function WKR_ROUND(U^{(c,s)}, V^{(c,s)}, Λ^{(c)}, ε_t)
    for t = 1 : γ do
        Sample a mini-batch M_t from X^{(s)}
        for each user i and item j in M_t parallel do
            Update U_i, V_j using Eqn. (29) and (30)
        end for
    end for
    Send updated U^{(c,s)} and V^{(c,s)} to the parameter server
end function
```

Where:
- Line 1: Initialize $\bar{h}_{i*}, \bar{h}_{*j}$, round length γ, mini-batch size m
- Line 2: **function** WKR_ROUND$(\mathbf{U}^{(c,s)}, \mathbf{V}^{(c,s)}, \Lambda^{(c)}, \epsilon_t)$
- Line 3: **for** $t = 1 : \gamma$ **do**
- Line 4: Sample a mini-batch $\mathcal{M}_t$ from $\mathcal{X}^{(s)}$
- Line 5: **for** each user i and item j in $\mathcal{M}_t$ ***parallel*** **do**
- Line 6: Update U_i, V_j using Eqn. (29) and (30)
- Line 9: Send updated $\mathbf{U}^{(c,s)}$ and $\mathbf{V}^{(c,s)}$ to the parameter server

DSGLD: Block Split Schemes



(a) square (b) column

Two levels of parallelism

- (a) within-chain (both X and $\{U, V\}$ are partitioned)
- (b) between-chain (X and U are partitioned, entire V is transmitted to parameter server)

Experiments

	Optimization	MCMC
Single Machine	SGD	SGLD, Gibbs
Distributed	DSGD	DSGLD

Dataset	# users	# items	# ratings
Netflix	480K	18K	100M
Yahoo	1.8M	136K	700M

Experiments

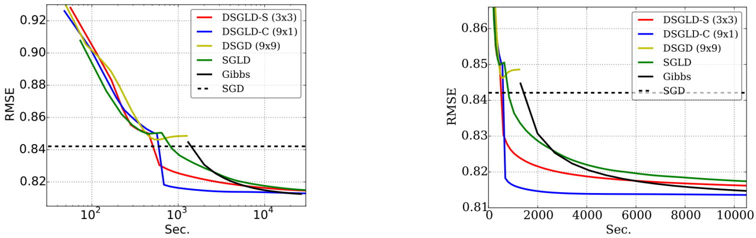


Figure: Netflix dataset (D = 30)

- Single node, multi core

Experiments

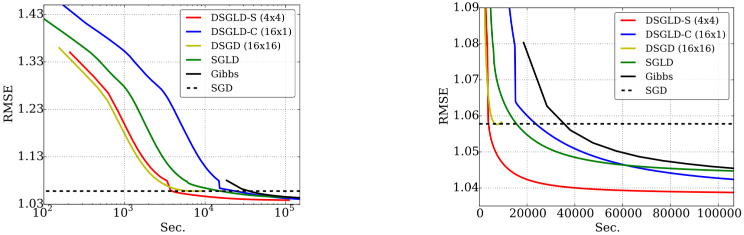
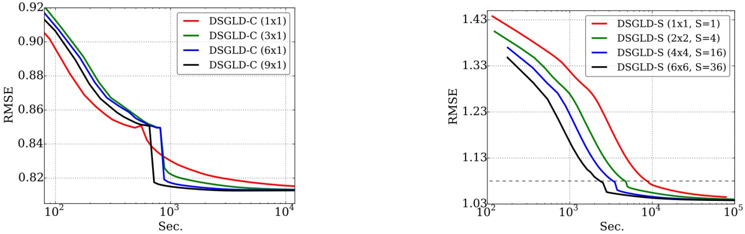


Figure: Yahoo Music Rating dataset (D = 30)

- 16 nodes

Experiments

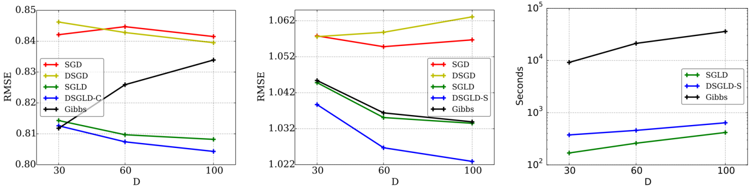


(a) DSGLD-C on the Netflix dataset

(b) DSGLD-S on the Yahoo dataset

Figure: The effect of the number of chains, number of workers, and block split

Experiments



(a) RMSE on Netflix (b) RMSE on Yahoo (c) Required time per sample

Figure: The effect of the latent feature dimension

Conclusion

- Paper combines ideas from DSGLD to scale Bayesian MF
- Avoids O(N) computation over the entire dataset
- Using multiple chains and multiple workers in parallel gives the method better and faster mixing
  - More chains help explore a broader space
  - Updating orthogonal blocks in parallel helps chain mix faster
  - Averaging predictions from independent chains gives better generalization
- Limitations: Slow worker, Not fully model parallel (model parallelism at worker)