Optimization for Machine Learning

Lecture: Ranking

S.V. N. (vishy) Vishwanathan

UCSC
vishy@ucsc.edu

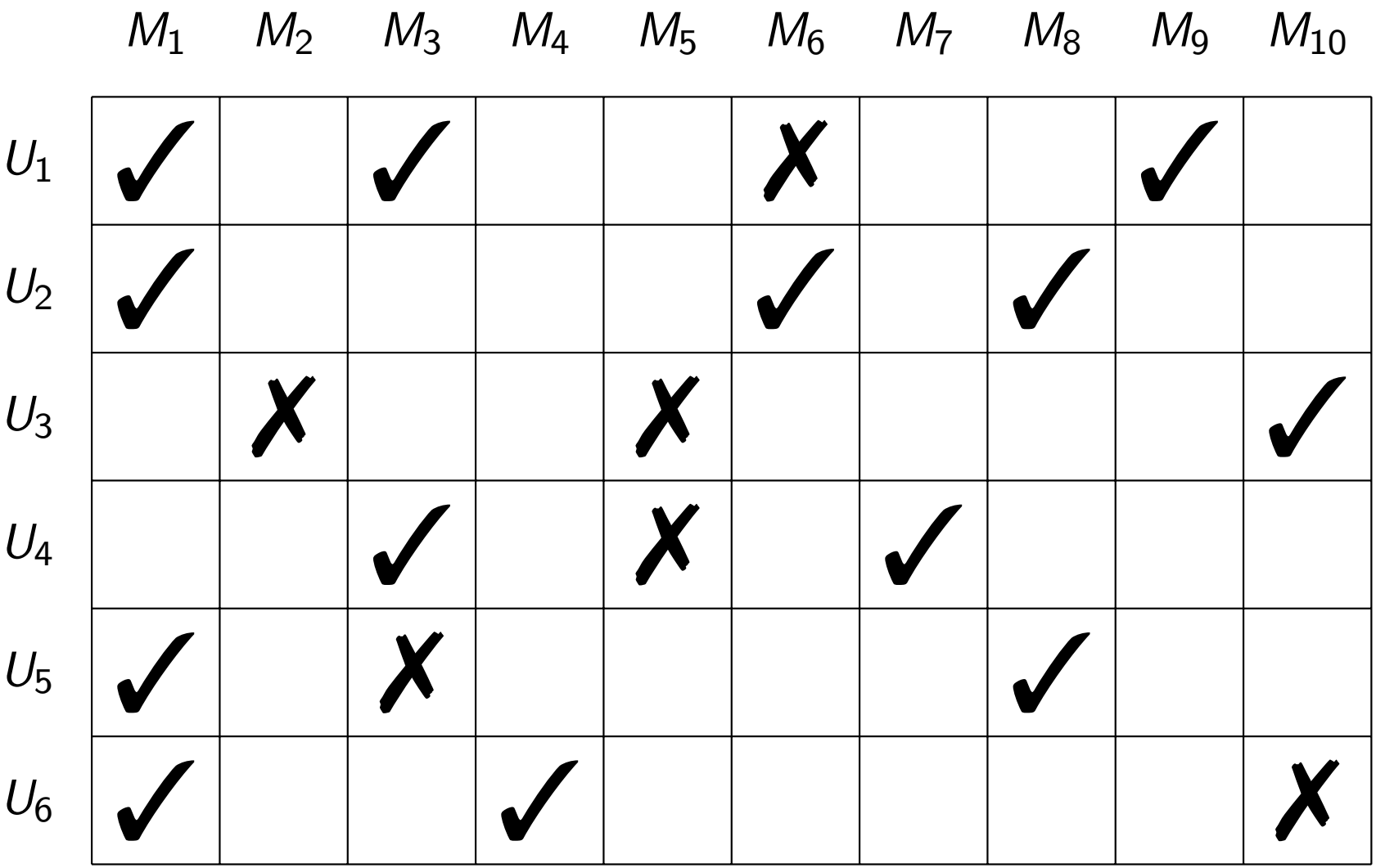
June 11, 2015

Problem setting

Users and Items

- $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ set of users
- $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$ set of items

Data



Problem setting

Users and Items

- $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ set of users
- $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$ set of items

Data

- For each user $x \in \mathcal{X}$ we have
  - $\mathcal{Y}^+ \subseteq \mathcal{Y}$: positive items
  - $\mathcal{Y}^- \subseteq \mathcal{Y}$: negative items
  - $\mathcal{Y}^0 \subseteq \mathcal{Y}$: unobserved

What we want

- A function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ such that
  - $f(x, y^+) \geq f(x, y^0) \geq f(x, y^-)$
- Note: actual value of $f(x, y)$ is not important. Only order matters.

Objective function

Optimize Rank

- Note: Simplifying assumption $\mathcal{Y}^- = \emptyset$
- Minimize the rank of positive items:

$$J(f) := \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}^+} \text{rank}_f(x, y)$$

- Problem: rank is hard to write analytically. How to optimize?

Rewrite the rank

$$\text{rank}_f(x, y) = \sum_{y' \in \mathcal{Y}} I\left(f(x, y) - f(x, y') < 0\right)$$

$$J(f) := \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}^+} \text{rank}_f(x, y)$$

Rewrite the rank

$$\text{rank}_f(x, y) = \sum_{y' \in \mathcal{Y}} I\left(f(x, y) - f(x, y') < 0\right)$$

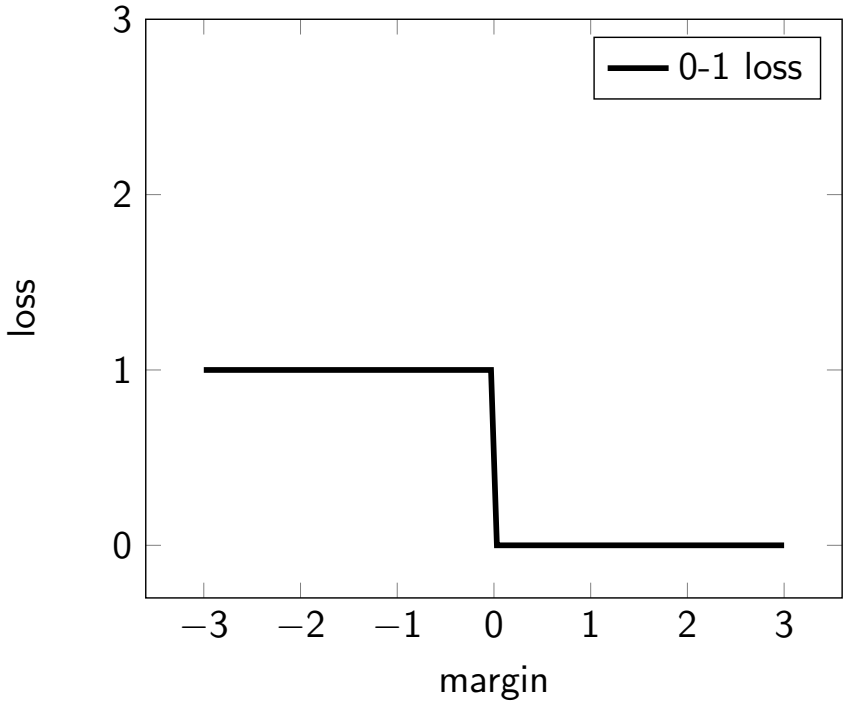
$$J(f) := \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}^+} \sum_{y' \in \mathcal{Y}} I\left(f(x, y) - f(x, y') < 0\right)$$

Rewrite the rank

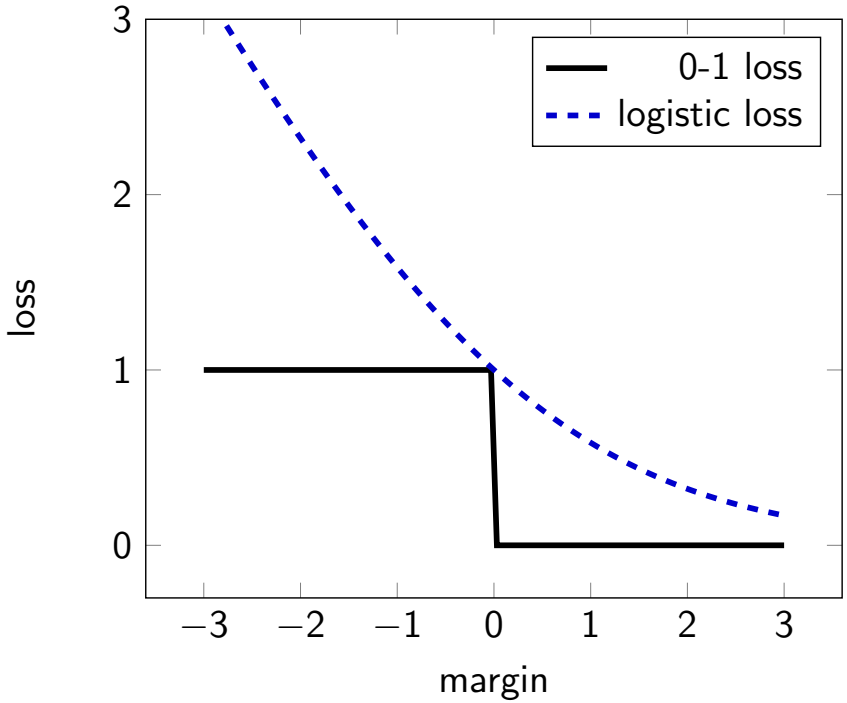
$$\text{rank}_f(x, y) = \sum_{y' \in \mathcal{Y}} I\left(f(x, y) - f(x, y') < 0\right)$$

$$J(f) := \sum_{(x,y) \in \Omega} \sum_{y' \in \mathcal{Y}} I\left(f(x, y) - f(x, y') < 0\right)$$

Upper bound the indicator



Upper bound the indicator



Upper bound the rank

$$\text{rank}_f(x, y) \leq \overline{\text{rank}}_f(x, y) = \sum_{y' \in \mathcal{Y}} \sigma\left(f(x, y) - f(x, y')\right)$$

$$J(f) := \sum_{(x,y) \in \Omega} \sum_{y' \in \mathcal{Y}} I\left(f(x, y) - f(x, y') < 0\right)$$

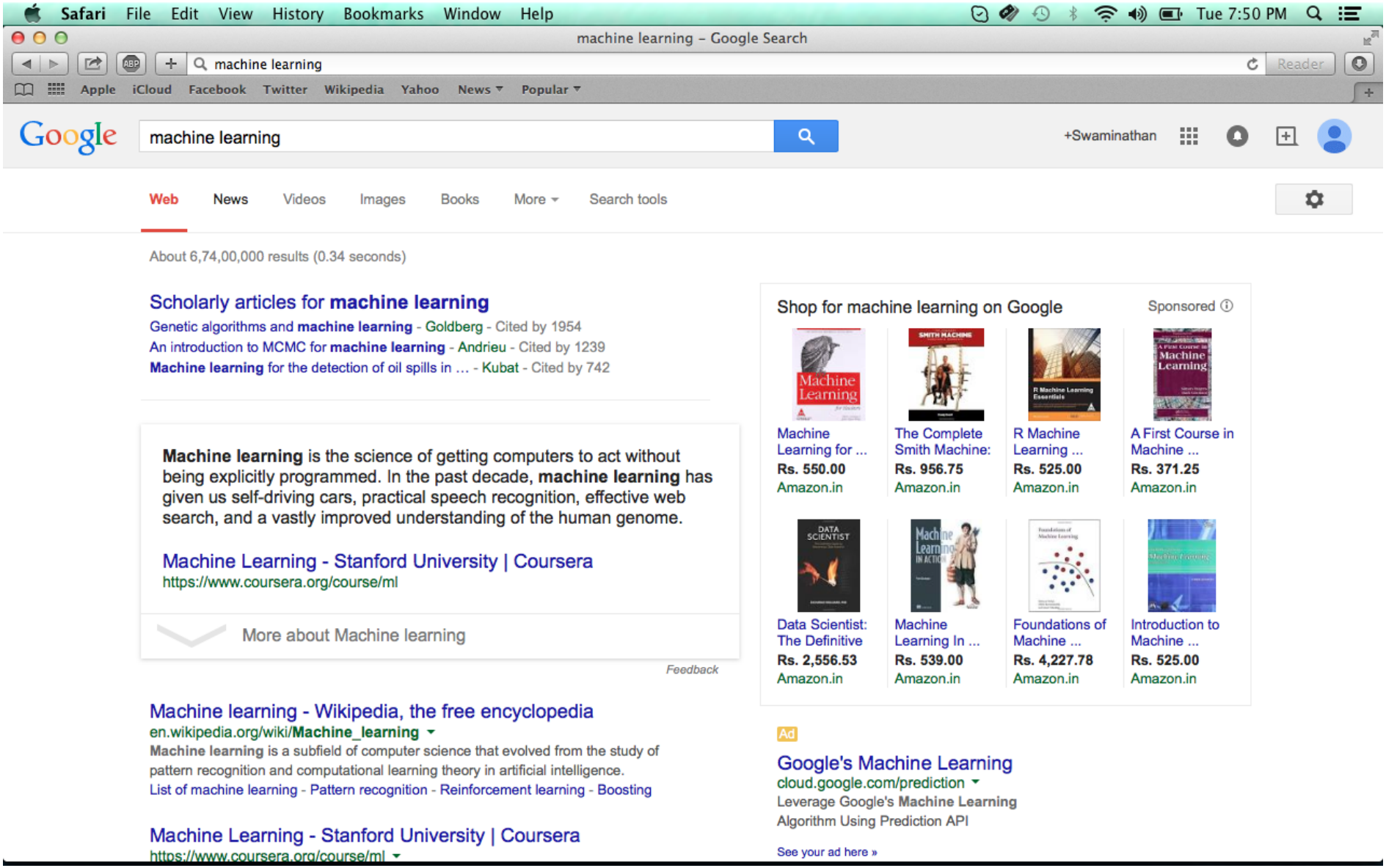
Upper bound the rank

$$\text{rank}_f(x, y) \leq \overline{\text{rank}}_f(x, y) = \sum_{y' \in \mathcal{Y}} \sigma\left(f(x, y) - f(x, y')\right)$$

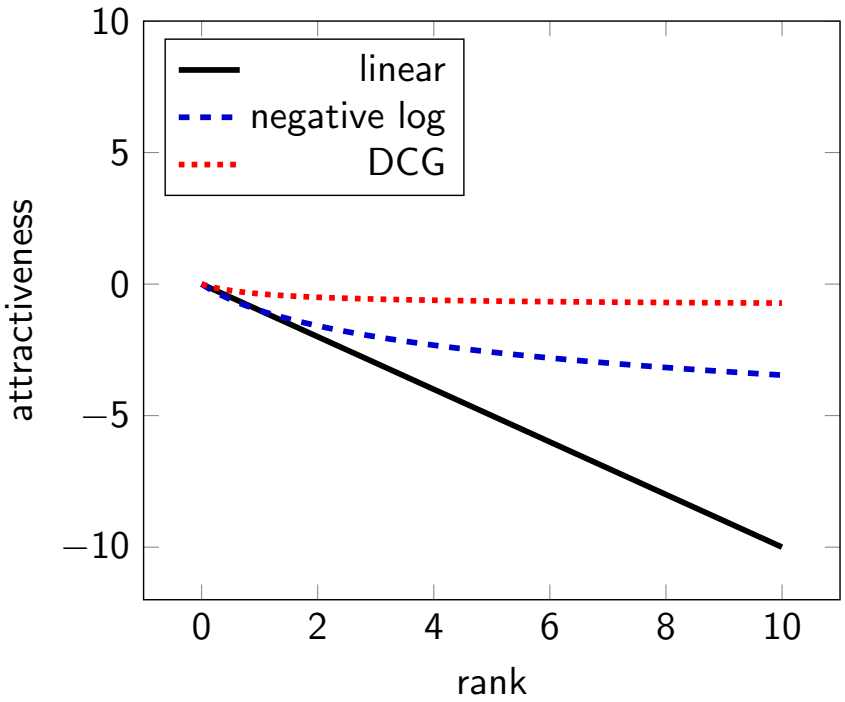
$$J(f) := \sum_{(x,y) \in \Omega} \sum_{y' \in \mathcal{Y}} \sigma\left(f(x, y) - f(x, y')\right)$$

where, for instance, $\sigma(t) = \log_2\left(1 + 2^{-t}\right)$.

Detour: DCG


Safari File Edit View History Bookmarks Window Help
Tue 7:50 PM
machine learning – Google Search
machine learning
Reader
Apple iCloud Facebook Twitter Wikipedia Yahoo News Popular
Google
machine learning
+Swaminathan
Web News Videos Images Books More Search tools
About 6,74,00,000 results (0.34 seconds)
Scholarly articles for machine learning
Genetic algorithms and machine learning - Goldberg - Cited by 1954
An introduction to MCMC for machine learning - Andrieu - Cited by 1239
Machine learning for the detection of oil spills in ... - Kubat - Cited by 742
Machine learning is the science of getting computers to act without being explicitly programmed. In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome.
Machine Learning - Stanford University | Coursera
https://www.coursera.org/course/ml
More about Machine learning
Feedback
Machine learning - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Machine_learning
Machine learning is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence.
List of machine learning - Pattern recognition - Reinforcement learning - Boosting
Machine Learning - Stanford University | Coursera
https://www.coursera.org/course/ml
Shop for machine learning on Google
Sponsored
Machine Learning for ... Rs. 550.00 Amazon.in
The Complete Smith Machine: Rs. 956.75 Amazon.in
R Machine Learning ... Rs. 525.00 Amazon.in
A First Course in Machine ... Rs. 371.25 Amazon.in
Data Scientist: The Definitive Rs. 2,556.53 Amazon.in
Machine Learning In ... Rs. 539.00 Amazon.in
Foundations of Machine ... Rs. 4,227.78 Amazon.in
Introduction to Machine ... Rs. 525.00 Amazon.in
Ad
Google's Machine Learning
cloud.google.com/prediction
Leverage Google's Machine Learning Algorithm Using Prediction API
See your ad here »

Attractiveness



Bending the loss

$$\text{rank}_f(x, y) \leq \overline{\text{rank}}_f(x, y) = \sum_{y' \in \mathcal{Y}} \sigma\left(f(x, y) - f(x, y')\right)$$

$$J(f) := \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}^+} \sum_{y' \in \mathcal{Y}} \sigma\left(f(x, y) - f(x, y')\right)$$

where, for instance, $\sigma(t) = \log_2\left(1 + 2^{-t}\right)$.

Bending the loss

$$\text{rank}_f(x, y) \leq \overline{\text{rank}}_f(x, y) = \sum_{y' \in \mathcal{Y}} \sigma\left(f(x, y) - f(x, y')\right)$$

$$J(f) := \sum_{(x,y) \in \Omega} \rho\left(\sum_{y' \in \mathcal{Y}} \sigma\left(f(x, y) - f(x, y')\right)\right)$$

where, for instance, $\sigma(t) = \log_2\left(1 + 2^{-t}\right)$ and $\rho(t) = \log_2(t + 1)$

Bending the loss

$$\text{rank}_f(x, y) \leq \overline{\text{rank}}_f(x, y) = \sum_{y' \in \mathcal{Y}} \sigma\left(f(x, y) - f(x, y')\right)$$

$$J(f) := \sum_{(x,y) \in \Omega} \rho\left(\sum_{y' \in \mathcal{Y}} \sigma\left(f(x, y) - f(x, y')\right)\right)$$

where $\sigma(t) = \log_2\left(1 + 2^{-t}\right)$ and $\rho(t) = \log_2(t + 1)$

Different ways to parameterize $f(x, y)$

Joint features

- If joint features of users and items available

$$f(x, y) = \langle \phi(x, y), w \rangle$$

Independent features

- If independent features of users and items available

$$f(x, y) = \phi(x)^\top W \phi'(y)$$

Latent models

- If only interactions are available

$$f(x, y) = \langle u_x, v_y \rangle$$

Optimization: Stochastic gradients

$$J(f) := \sum_{(x,y)\in\Omega} \rho \left(\sum_{y'\in\mathcal{Y}} \sigma \left(f(u_x, v_y) - f(u_x, v_{y'}) \right) \right)$$

$$J_{x,y}(f) := |\Omega| \, \rho \left(\sum_{y'\in\mathcal{Y}} \sigma \left(f(u_x, v_y) - f(u_x, v_{y'}) \right) \right)$$

$$\nabla J_{x,y}(f) := |\Omega| \, \nabla\rho \left(\sum_{y'\in\mathcal{Y}} \sigma \left(f(u_x, v_y) - f(u_x, v_{y'}) \right) \right)$$

Bounding ρ

Since ρ is concave

$$\rho(t) \leq \rho\left(\xi^{-1}\right) + \rho'\left(\xi^{-1}\right) \cdot \left(t - \xi^{-1}\right)$$

Bound tight at $\xi = t^{-1}$

$$J(f) := \sum_{(x,y)\in\Omega} \rho\left(\sum_{y'\in\mathcal{Y}} \sigma\left(f(u_x, v_y) - f(u_x, v_{y'})\right)\right)$$

Bounding ρ

Since ρ is concave

$$\rho(t) \leq \rho\left(\xi^{-1}\right) + \rho'\left(\xi^{-1}\right) \cdot \left(t - \xi^{-1}\right)$$

Bound tight at $\xi = t^{-1}$

$$J(f) := \sum_{(x,y)\in\Omega} \rho\left(\xi_{xy}^{-1}\right) + \rho'\left(\xi_{xy}^{-1}\right) \cdot \left(\sum_{y'} \sigma(f(u_x, v_y) - f(u_x, v_{y'})) - \xi_{xy}^{-1}\right)$$

Alternate minimization

(U, V)-step

$$\nabla J(u, v|\xi) := \sum_{(x,y)\in\Omega} \sum_{y'} \rho'\left(\xi_{xy}^{-1}\right) \nabla\sigma(f(u_x, v_y) - f(u_x, v_{y'}))$$

- Sample (x, y) uniformly from Ω
- Sample y' uniformly from $\mathcal{Y}$
- Estimate the gradient by

$$|\Omega| \cdot |\mathcal{Y}| \cdot \rho'\left(\xi_{xy}^{-1}\right) \cdot \nabla_{u_x, v_y}\sigma(f(u_x, v_y) - f(u_x, v_{y'}))$$

- Update u_x and v_y

Alternate minimization

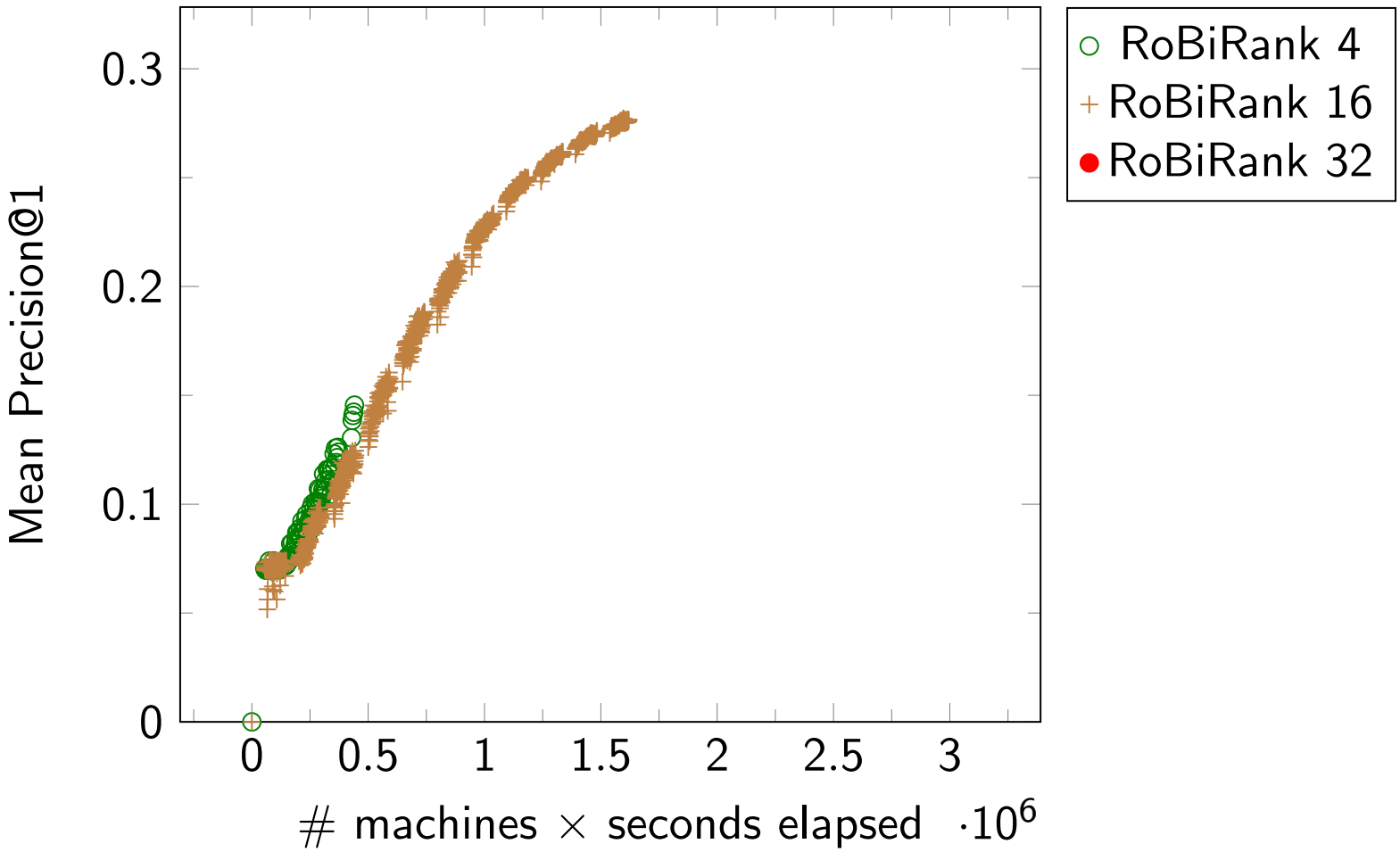
ξ_{xy}-step

If U and V are fixed then

$$\xi_{xy} = \frac{1}{\sum_{y'} \sigma(f(u_x, v_y) - f(u_x, v_{y'}))}$$

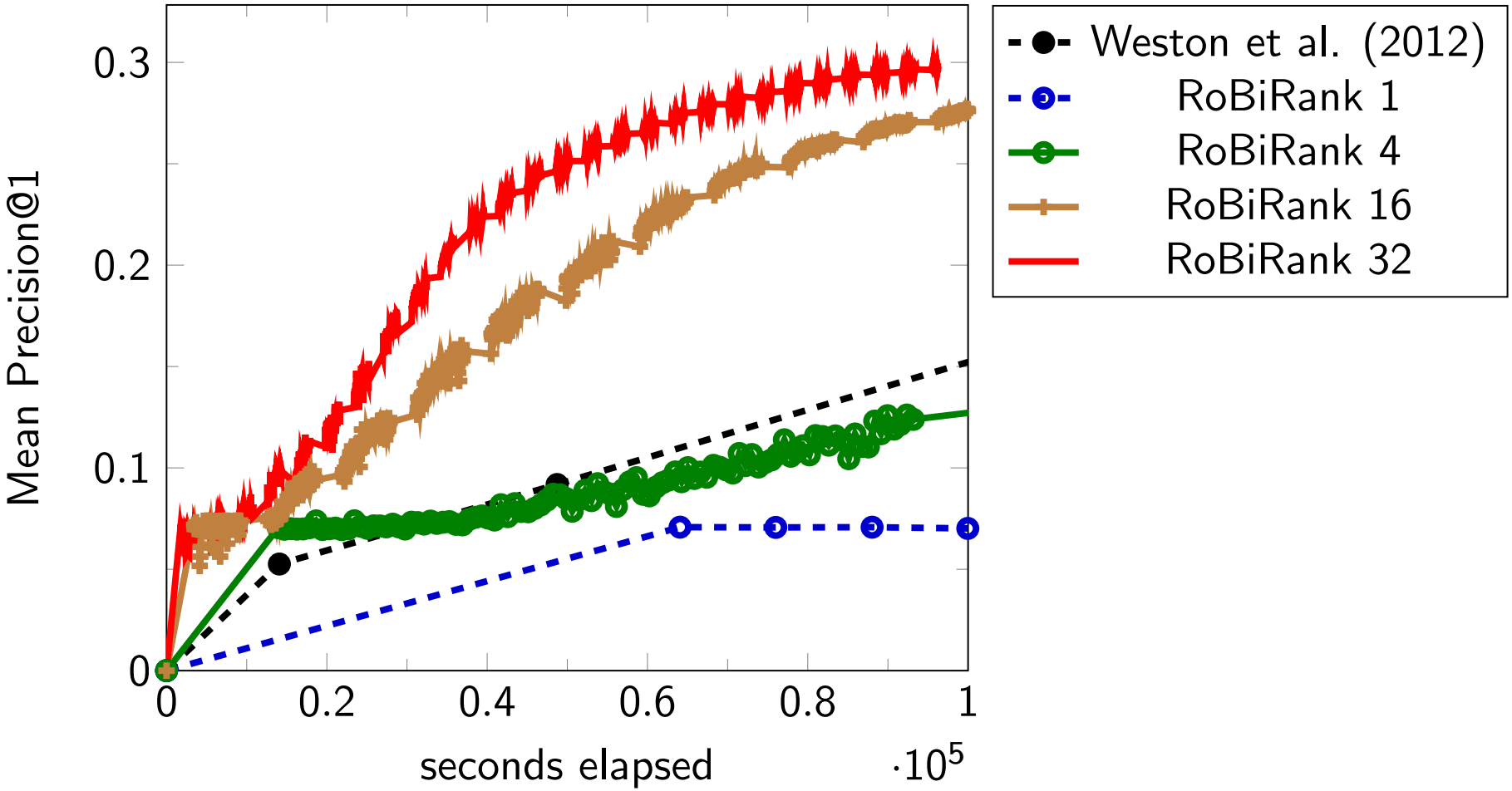
Experiments: Million song dataset

1,129,318 users, 386,133 songs, and 49,824,519 records



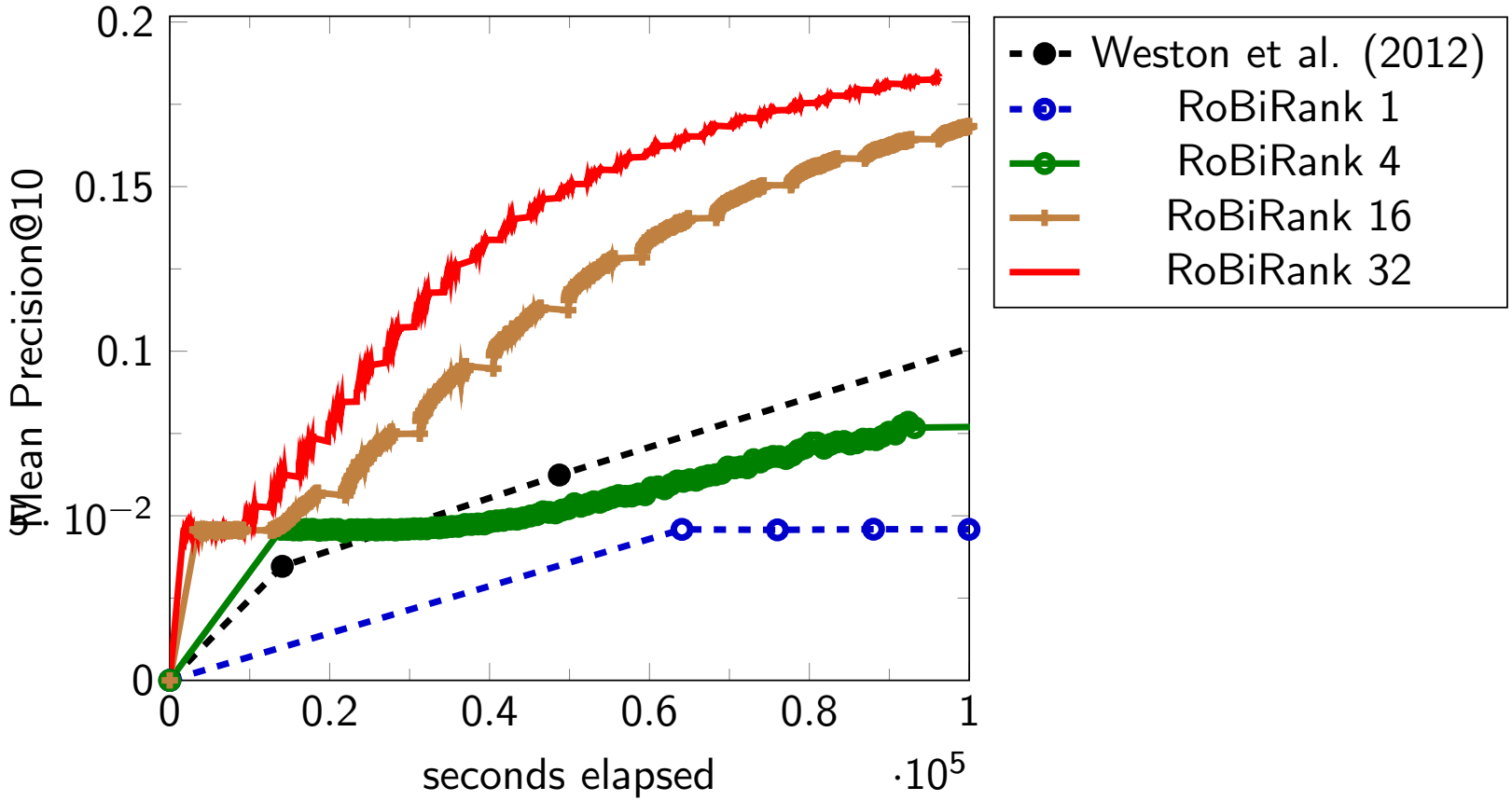
Experiments: Million song dataset

1,129,318 users, 386,133 songs, and 49,824,519 records

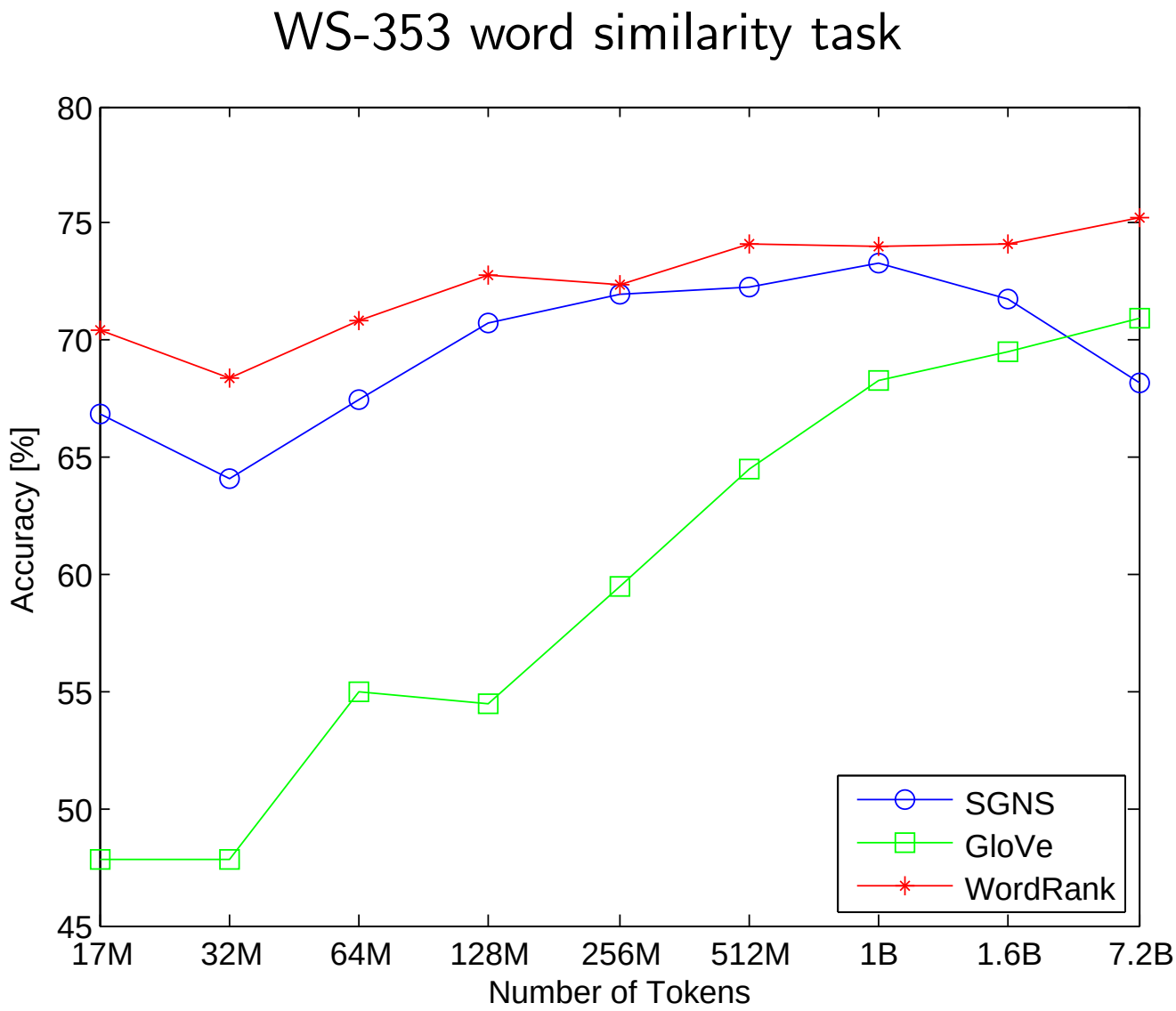


Experiments: Million song dataset

1,129,318 users, 386,133 songs, and 49,824,519 records



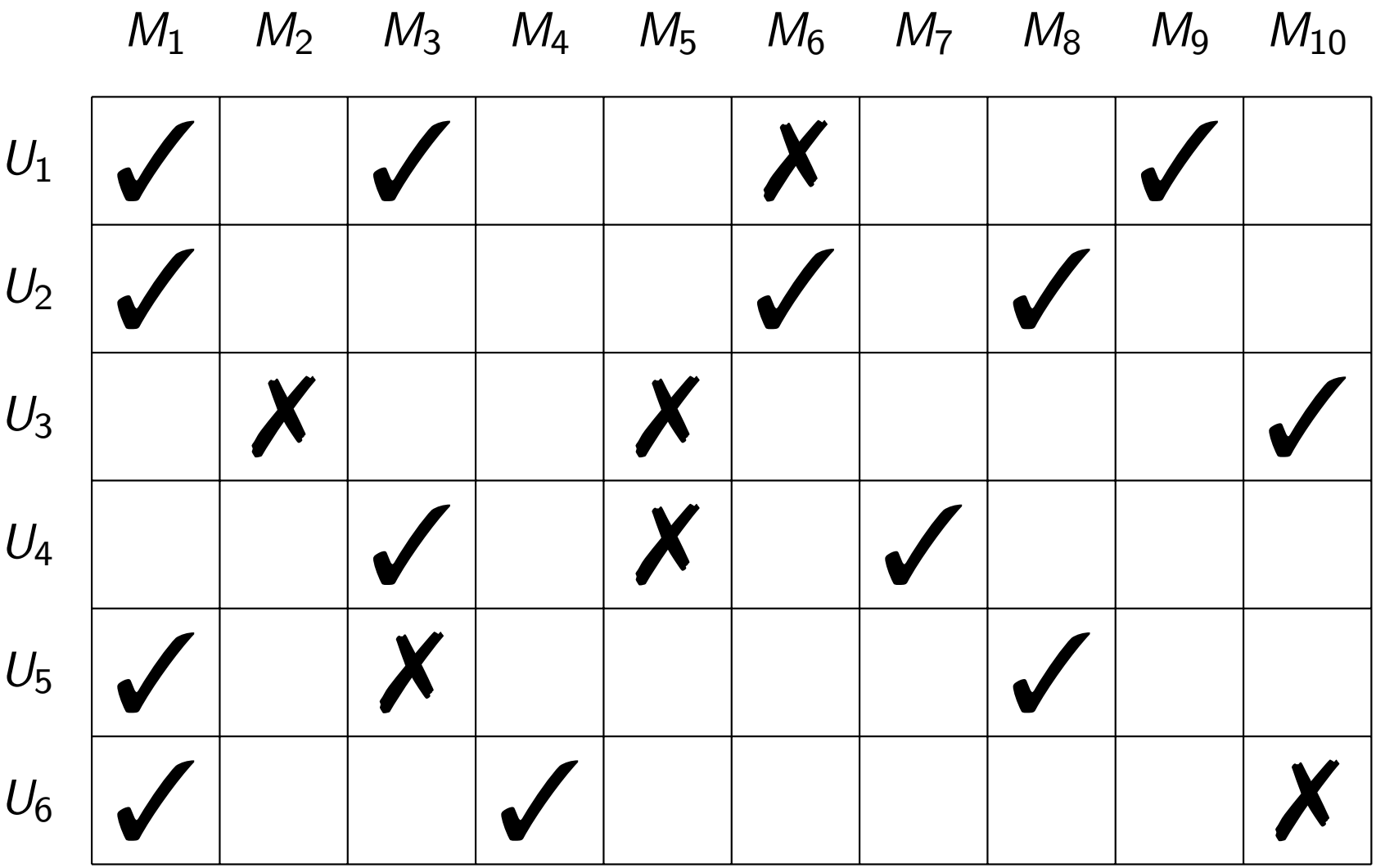
WordRank


WS-353 word similarity task
Accuracy [%]
80
75
70
65
60
55
50
45
17M
32M
64M
128M
256M
512M
1B
1.6B
7.2B
Number of Tokens
SGNS
GloVe
WordRank

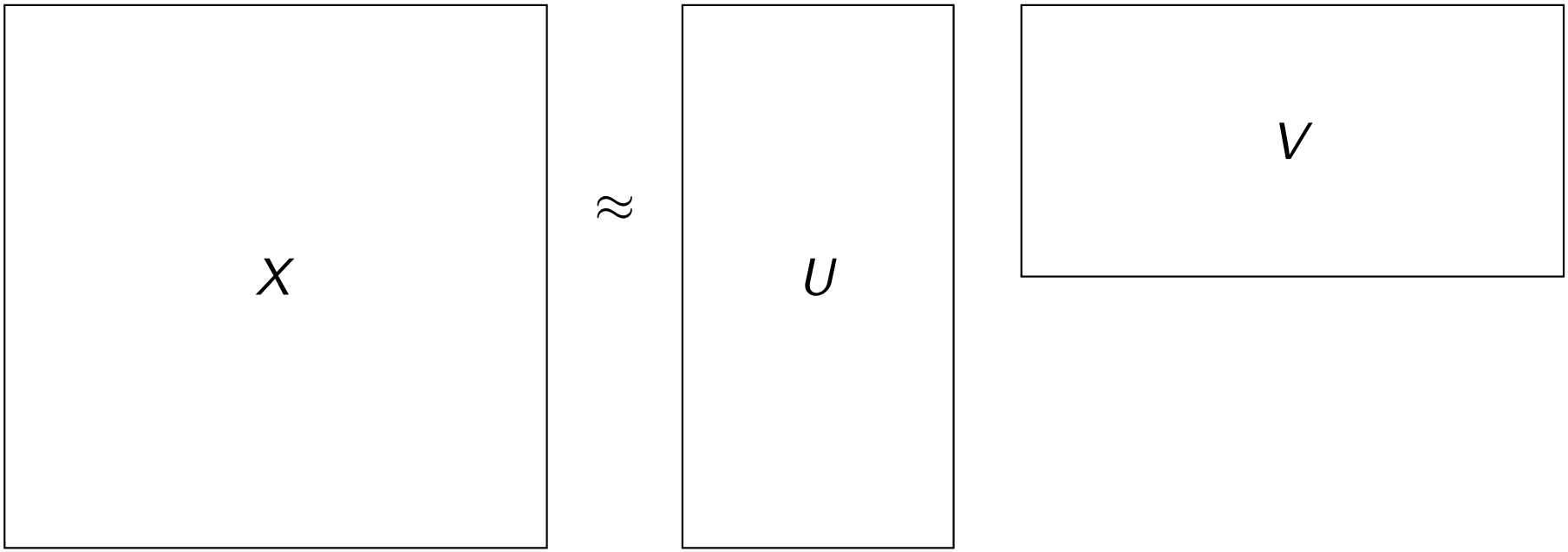
WordRank


Google word analogy task
Accuracy [%]
80
75
70
65
60
55
50
45
40
35
30
17M
32M
64M
128M
256M
512M
1B
1.6B
7.2B
Number of Tokens
SGNS
GloVe
WordRank

Collaborative filtering



Matrix completion



Matrix completion

$$\min_{\substack{U \in \mathbb{R}^{m \times k} \\ V \in \mathbb{R}^{n \times k}}} J(U, V),$$

$$J(U, V) = \frac{1}{2} \sum_{(x,y) \in \Omega} \left\{ (X_{xy} - \langle u_x, v_y \rangle)^2 + \lambda \left(\|u_x\|^2 + \|v_y\|^2 \right) \right\}.$$

Stochastic approximation

$$J(U,V) \approx J_{x,y}(U,V) = \frac{1}{2}\left\{(X_{xy} - \langle u_x, v_y \rangle)^2 + \lambda\left(\|u_x\|^2 + \|v_y\|^2\right)\right\}$$

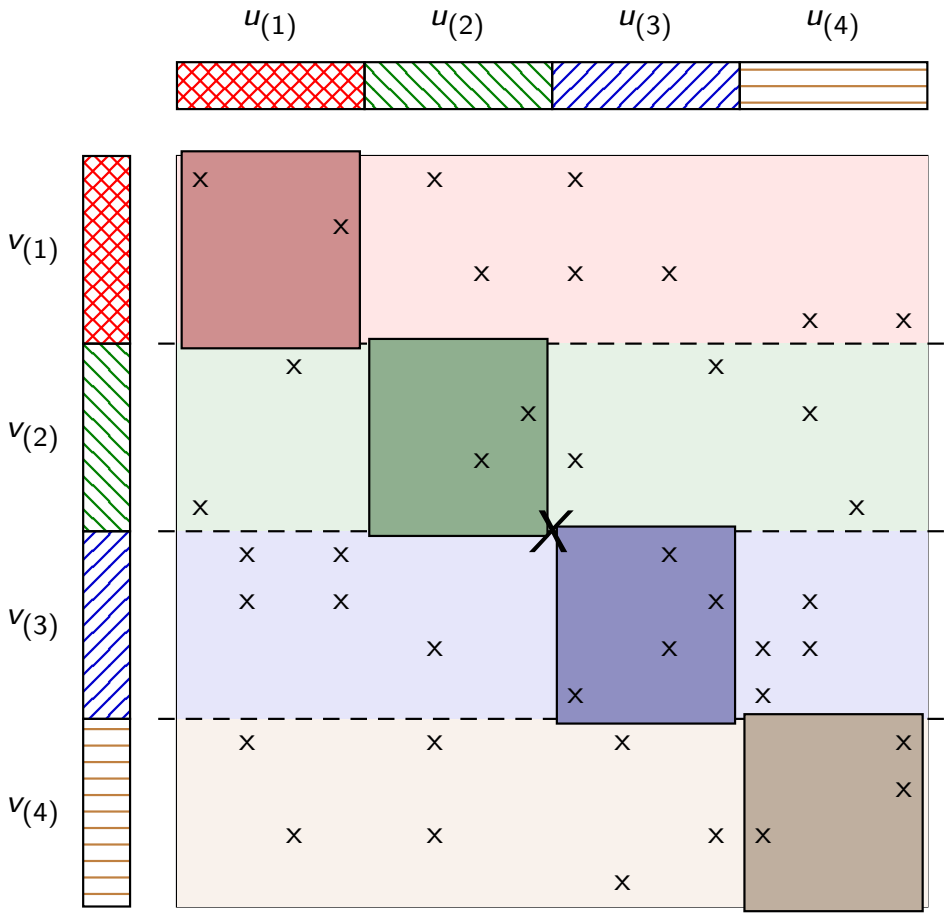
$$\nabla_{u_{y'}} J_{x,y}(U,V) = \begin{cases} (X_{uv} - \langle u_x, v_y \rangle)\, v_j + \lambda u_x, & \text{for } x = x' \\ 0 & \text{otherwise} \end{cases}$$

$$\nabla_{v_{y'}} J_{x,y}(U,V) = \begin{cases} (X_{xy} - \langle u_x, v_y \rangle)\, u_x + \lambda v_y, & \text{for } y = y' \\ 0 & \text{otherwise} \end{cases}$$

Stochastic updates

$$u_x \leftarrow u_x - \eta \left(\left(X_{xy} - \langle u_x, v_y \rangle \right) v_y + \lambda u_x \right)$$
$$v_y \leftarrow v_y - \eta \left(\left(X_{xy} - \langle u_x, v_y \rangle \right) u_x + \lambda v_y \right)$$

Distributed SGD (Gemulla et al., 2011, Re and Recht, 2012)



Some observations

The good

- Updates are decoupled and easy to parallelize
- Easy to implement using map-reduce

The bad

- Communication and computation are interleaved
  - When network is active then CPU is idle
  - When CPU is active then network is active

Question: Can we keep CPU and network simultaneously busy?

Illustration of NOMAD communication

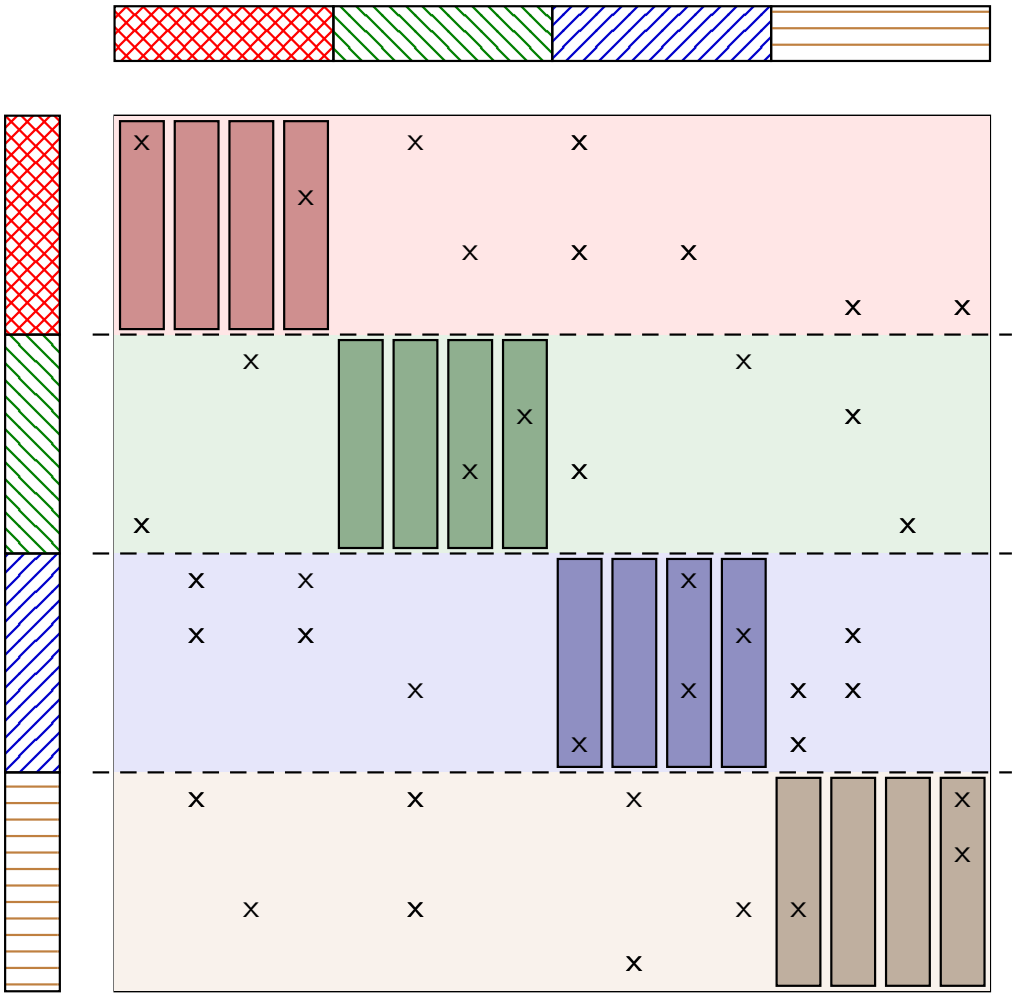


Illustration of NOMAD communication

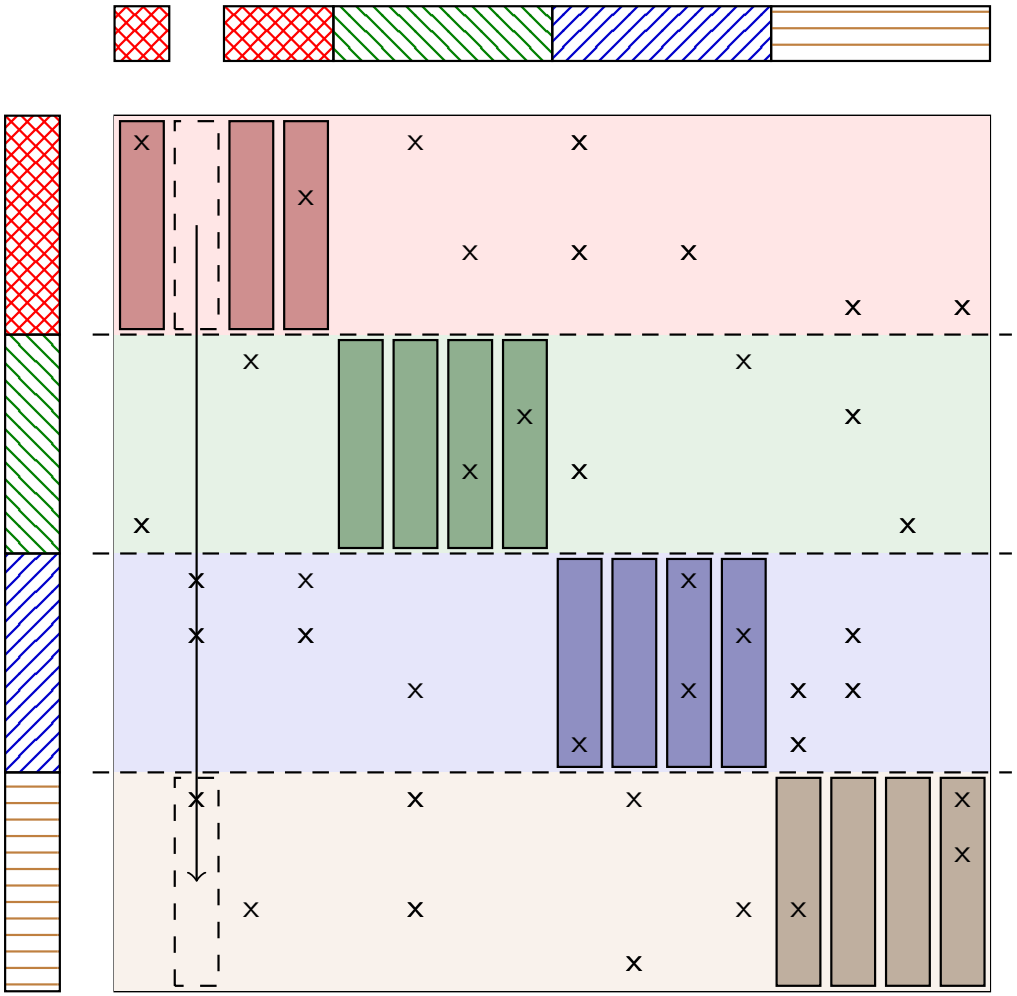


Illustration of NOMAD communication

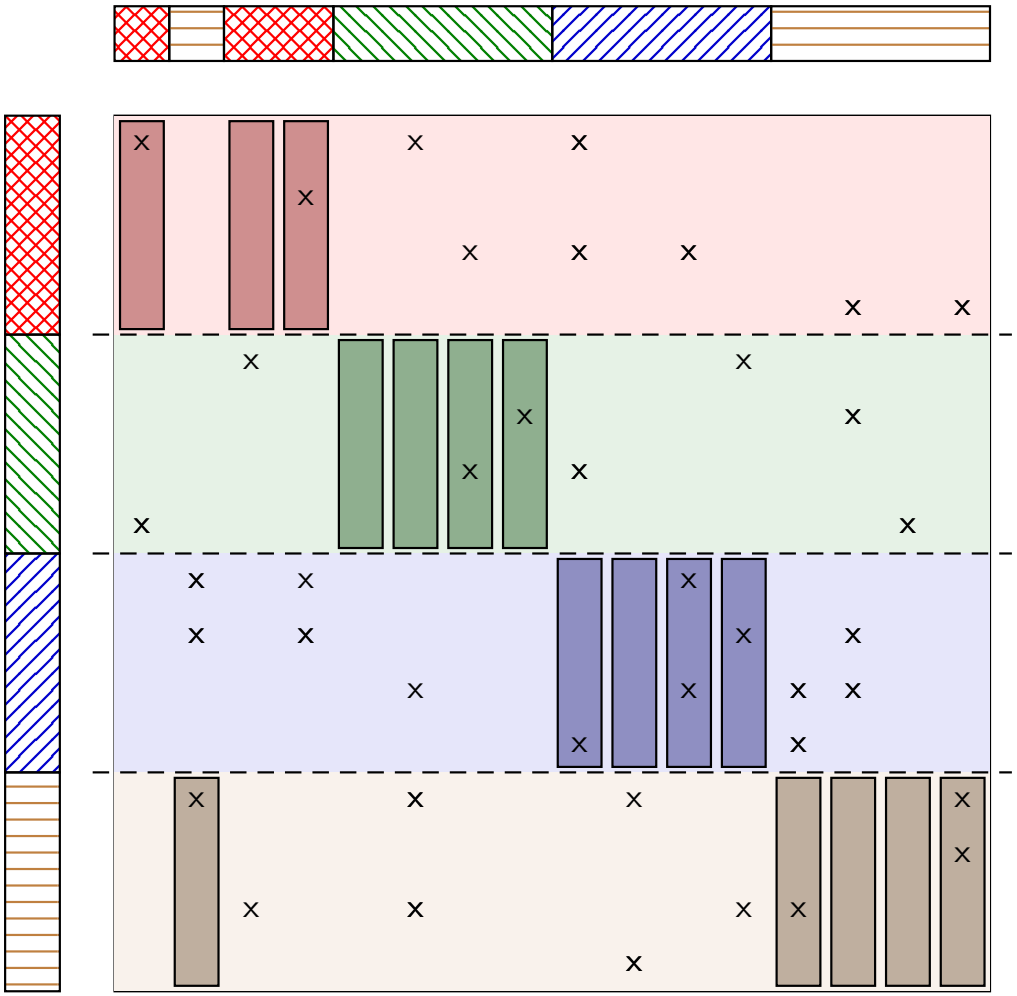


Illustration of NOMAD communication

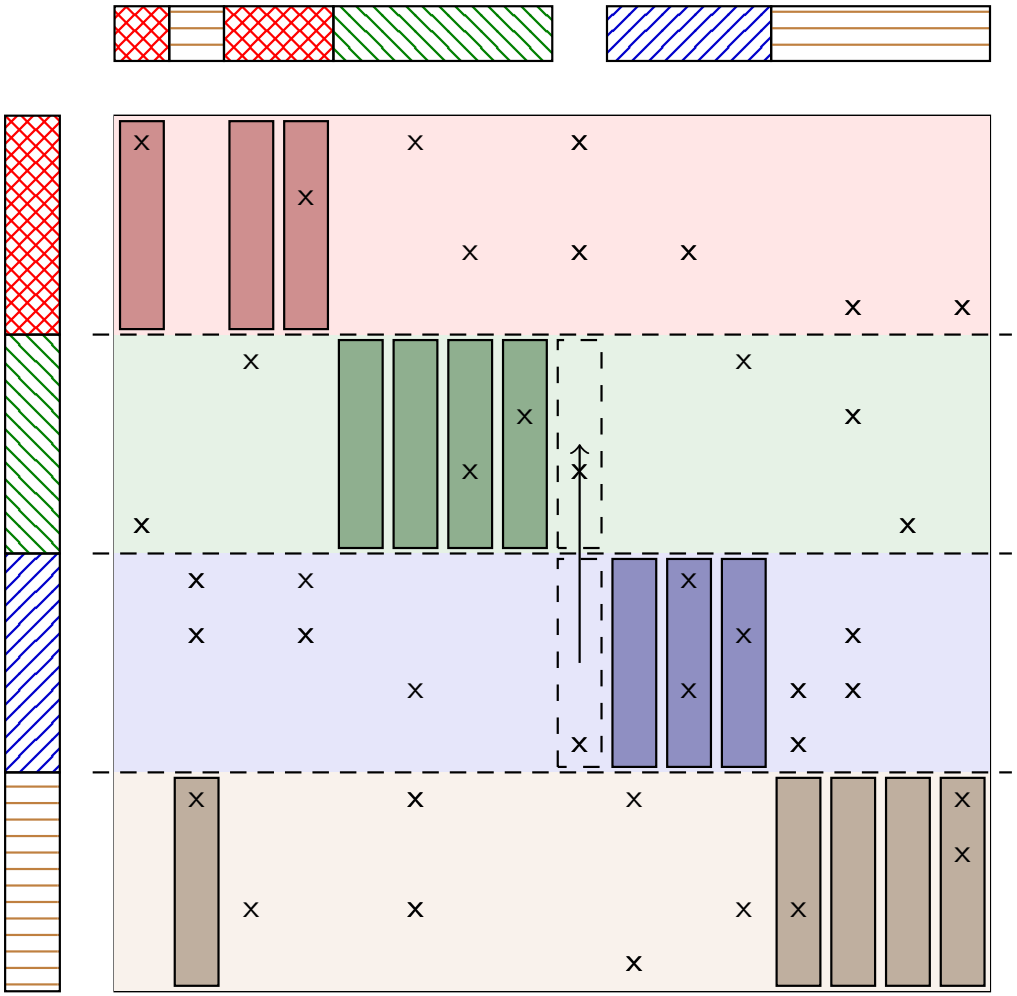


Illustration of NOMAD communication

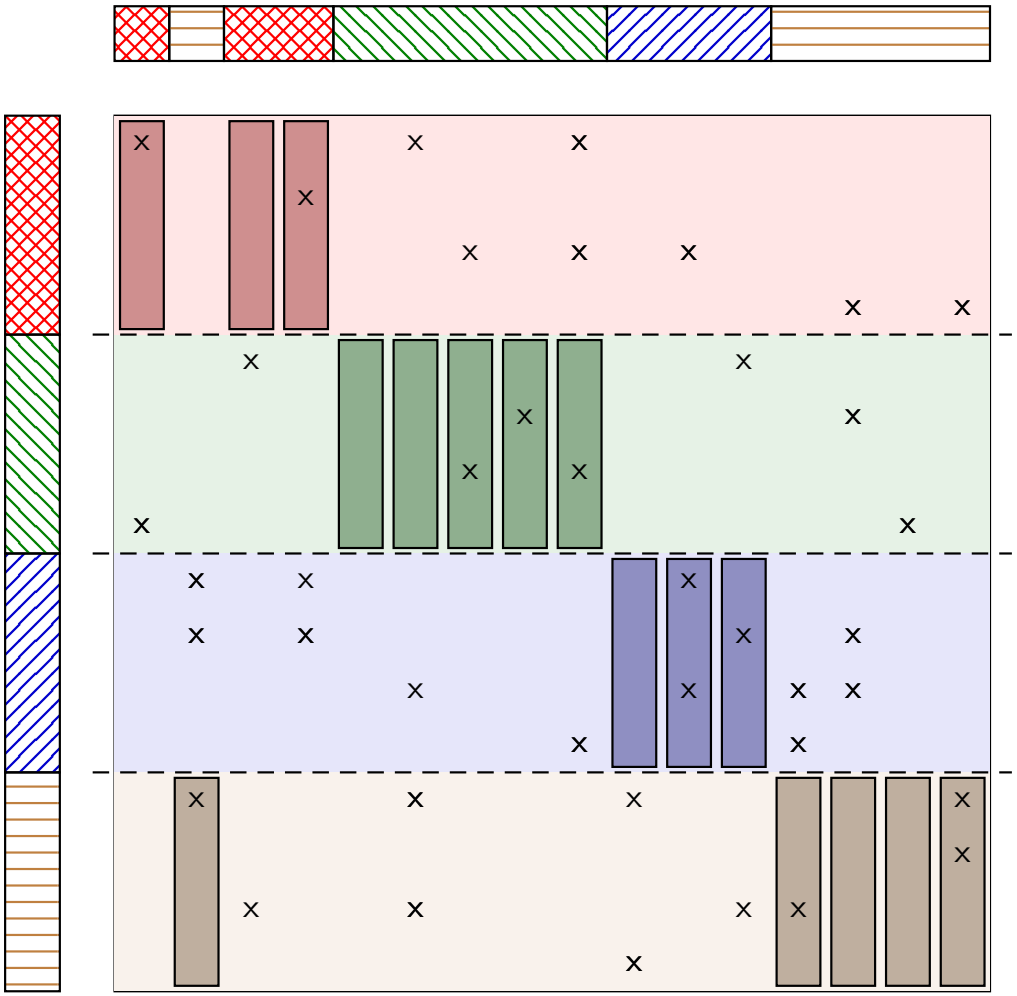


Illustration of NOMAD communication

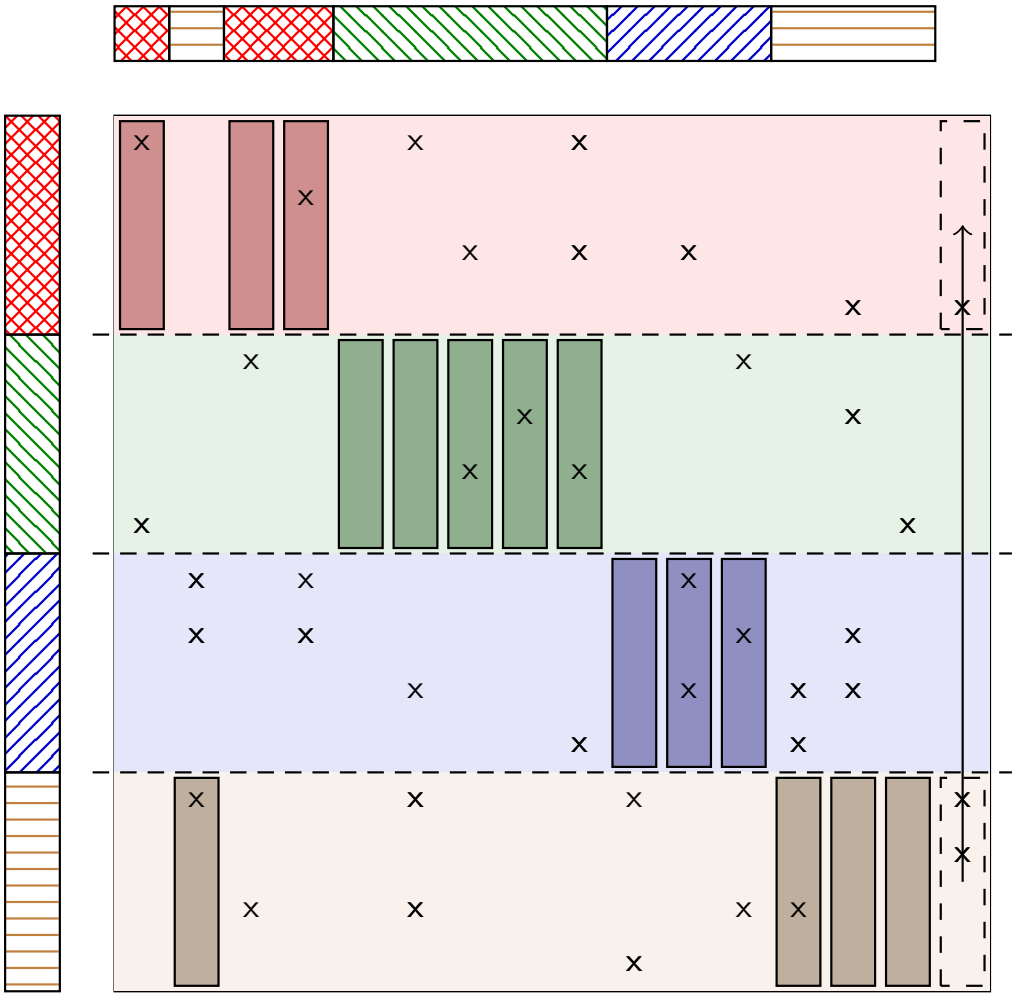


Illustration of NOMAD communication

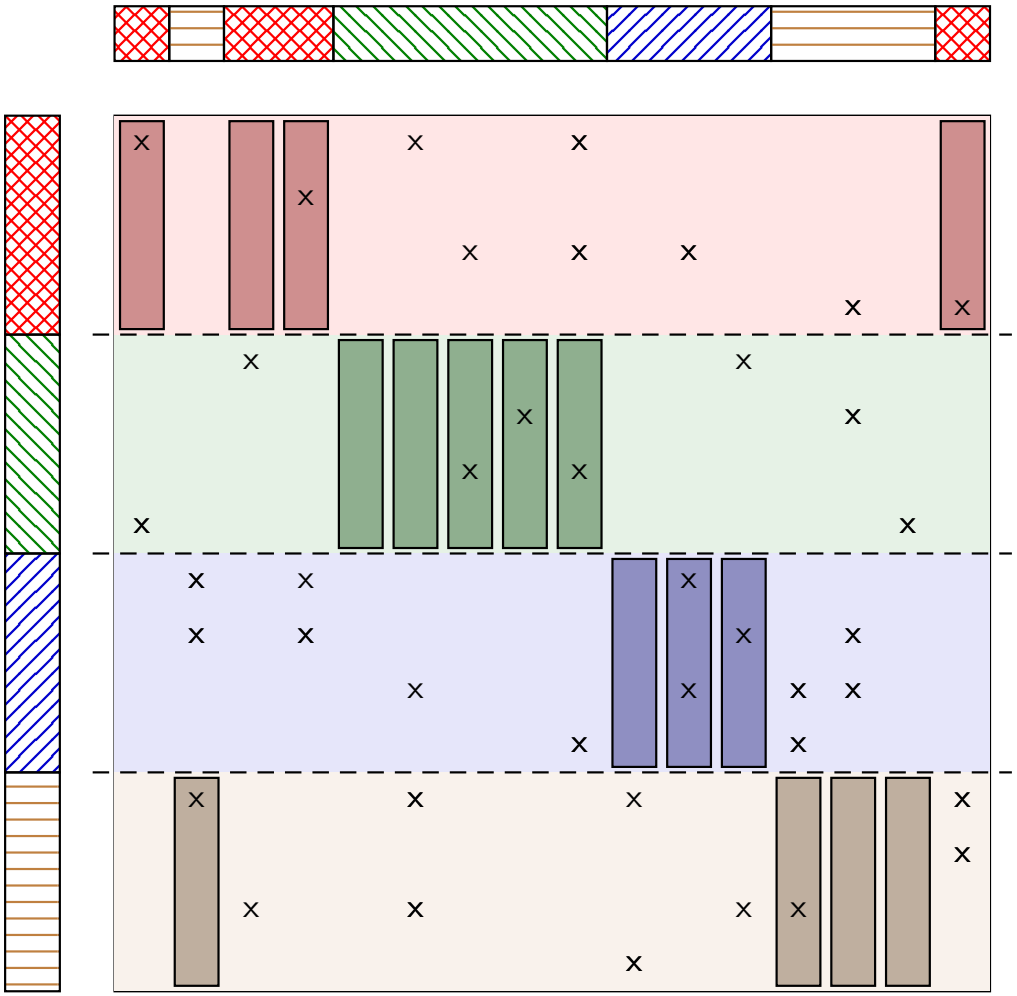


Illustration of NOMAD communication

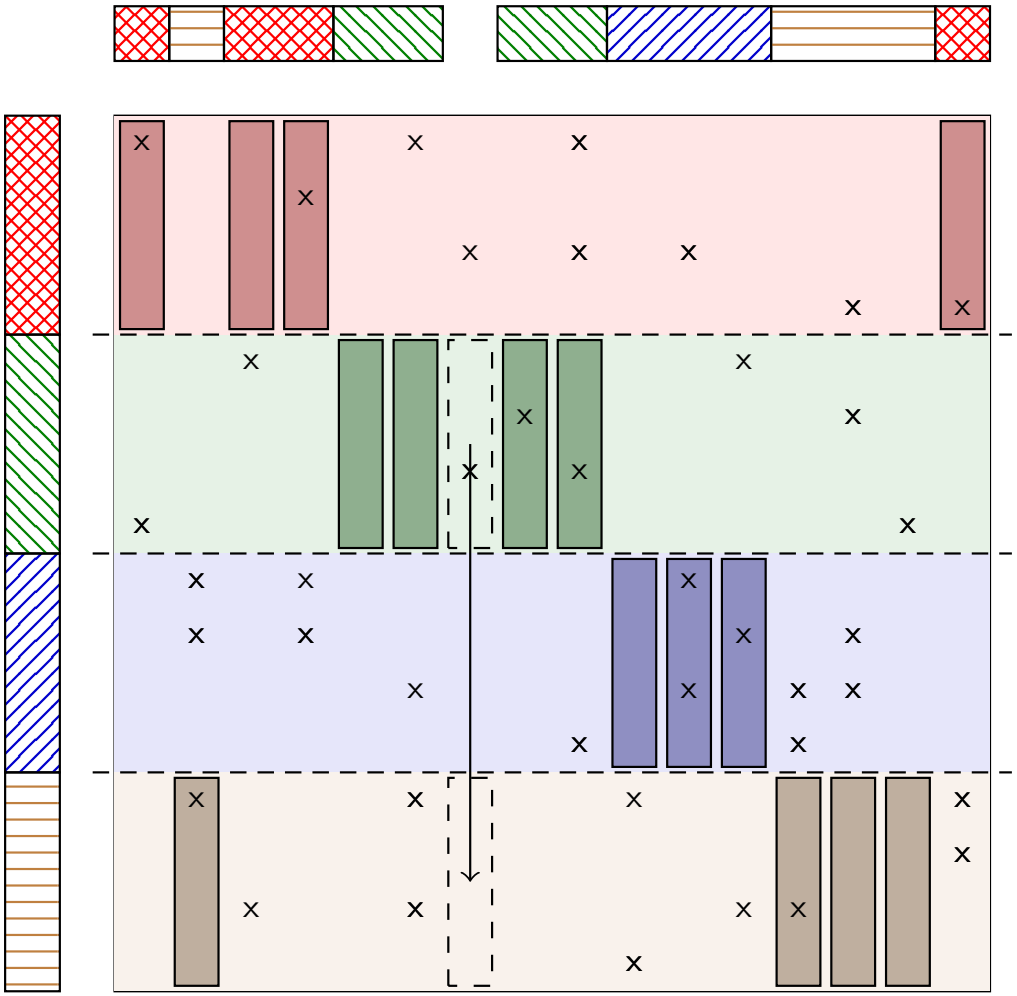


Illustration of NOMAD communication

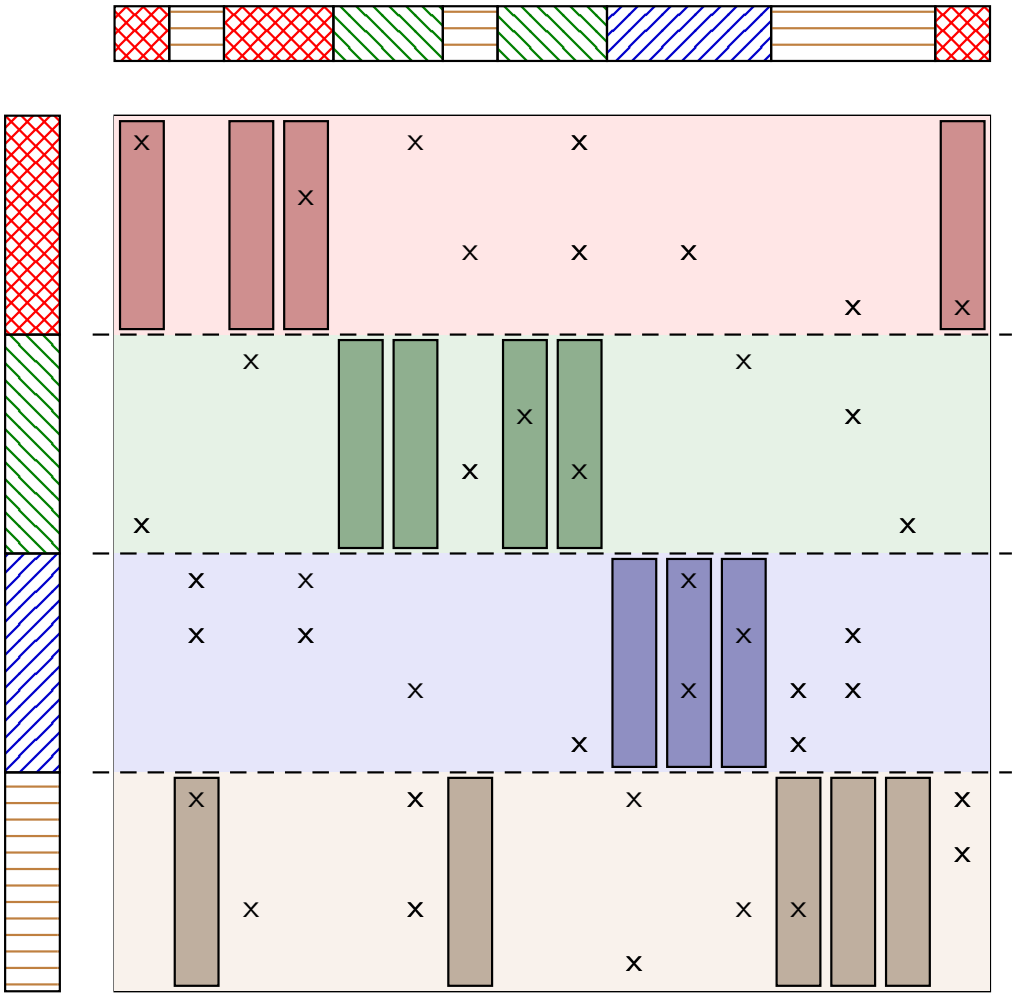


Illustration of NOMAD communication

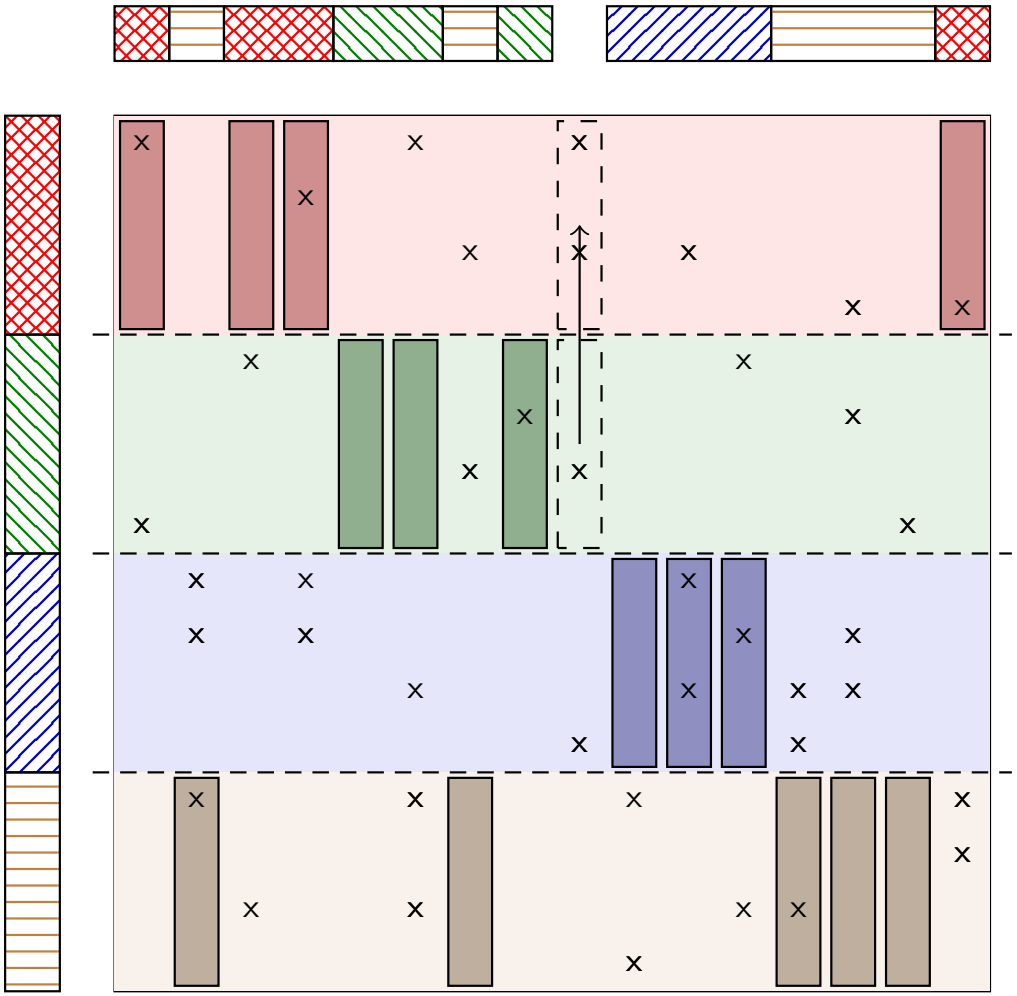
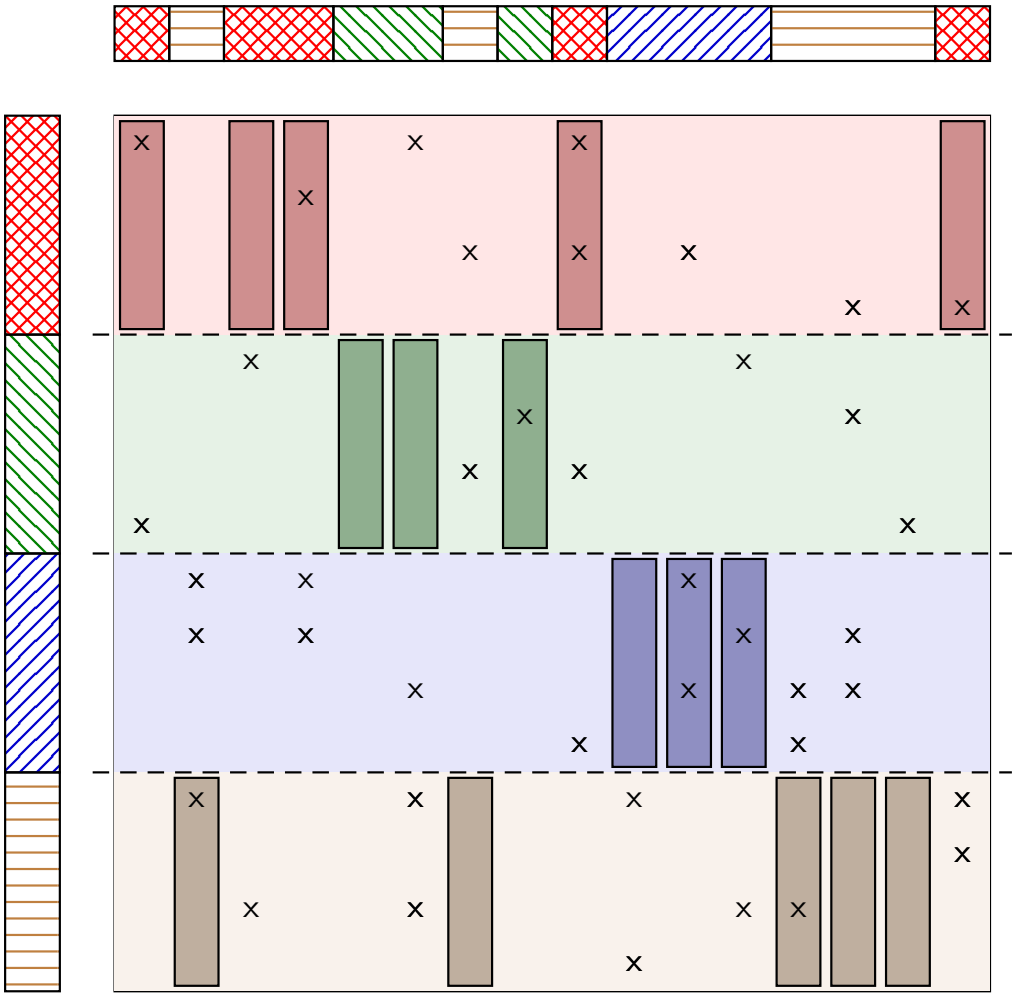
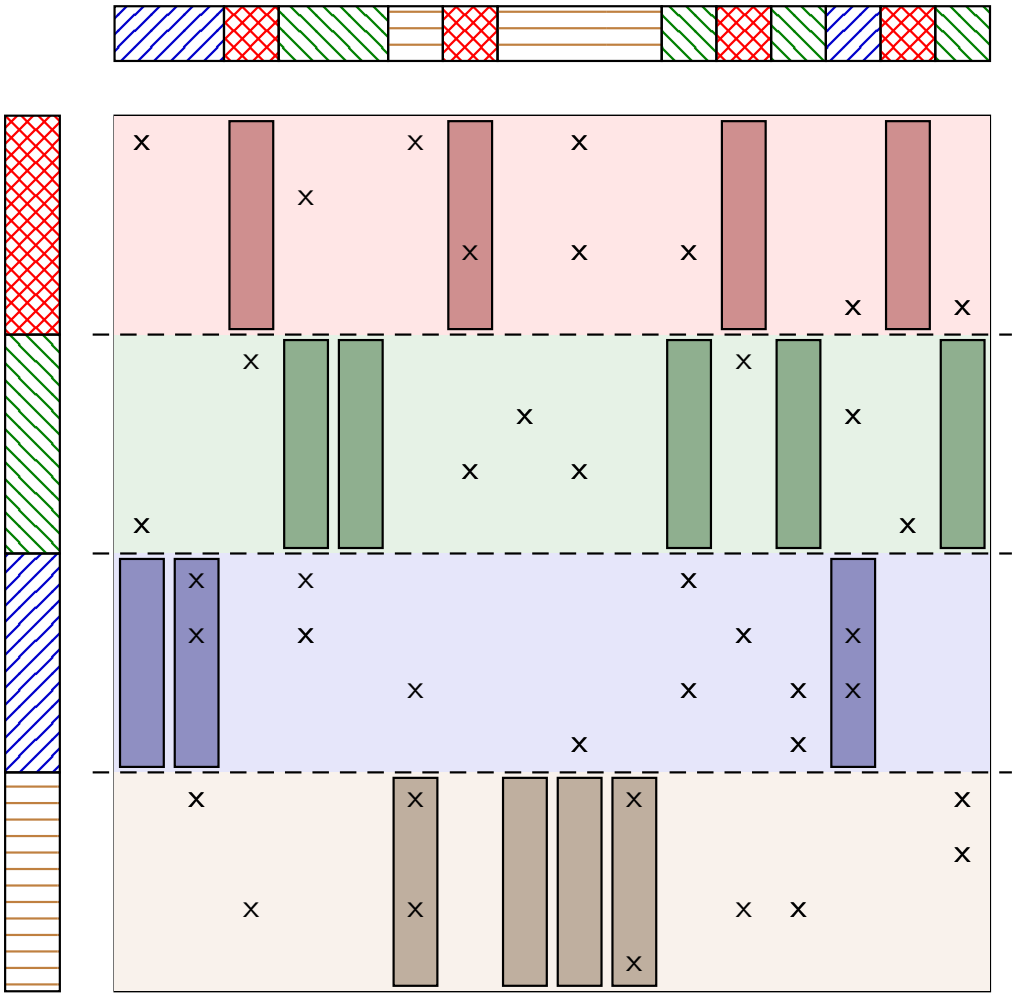


Illustration of NOMAD communication



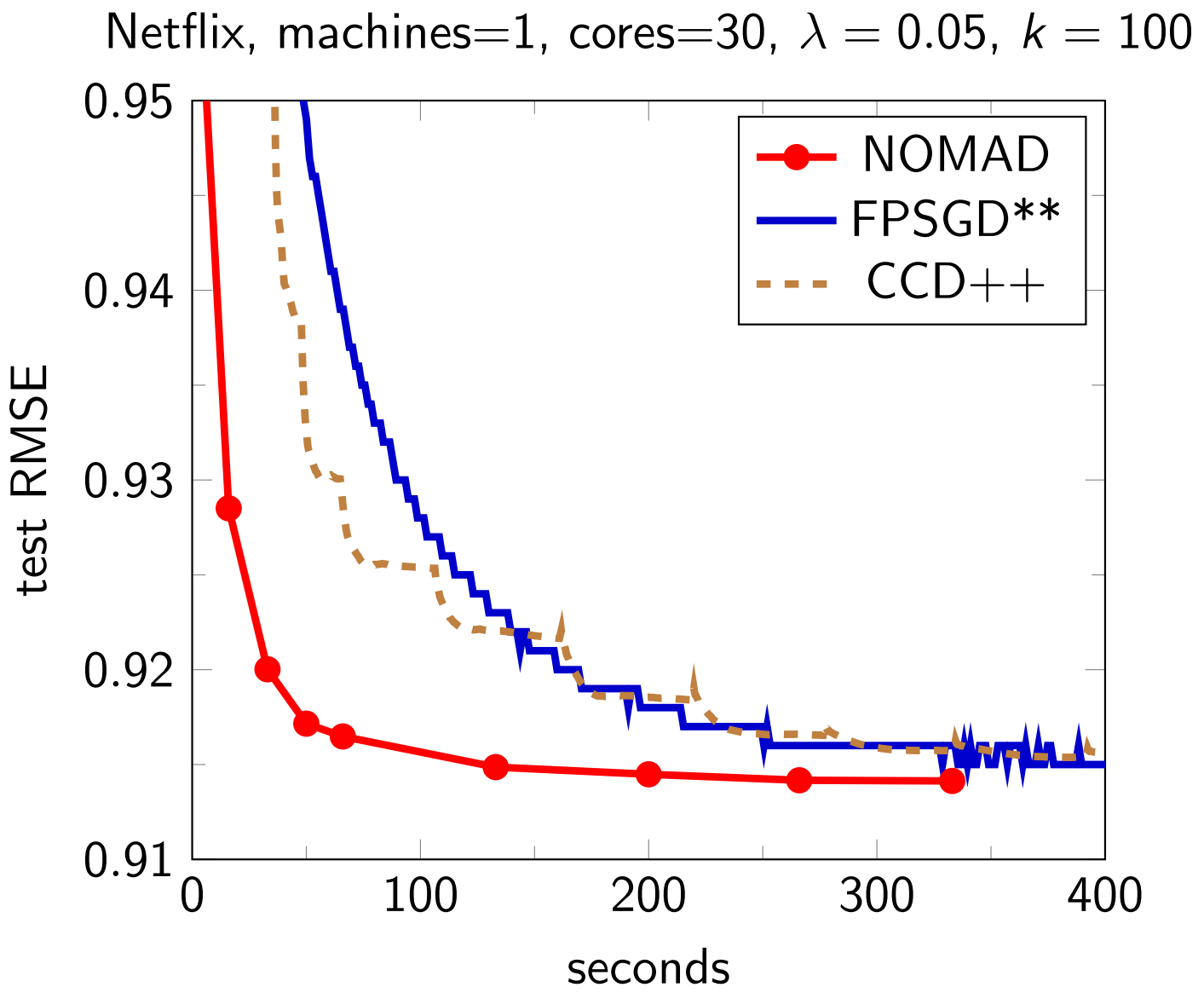
Eventually . . .



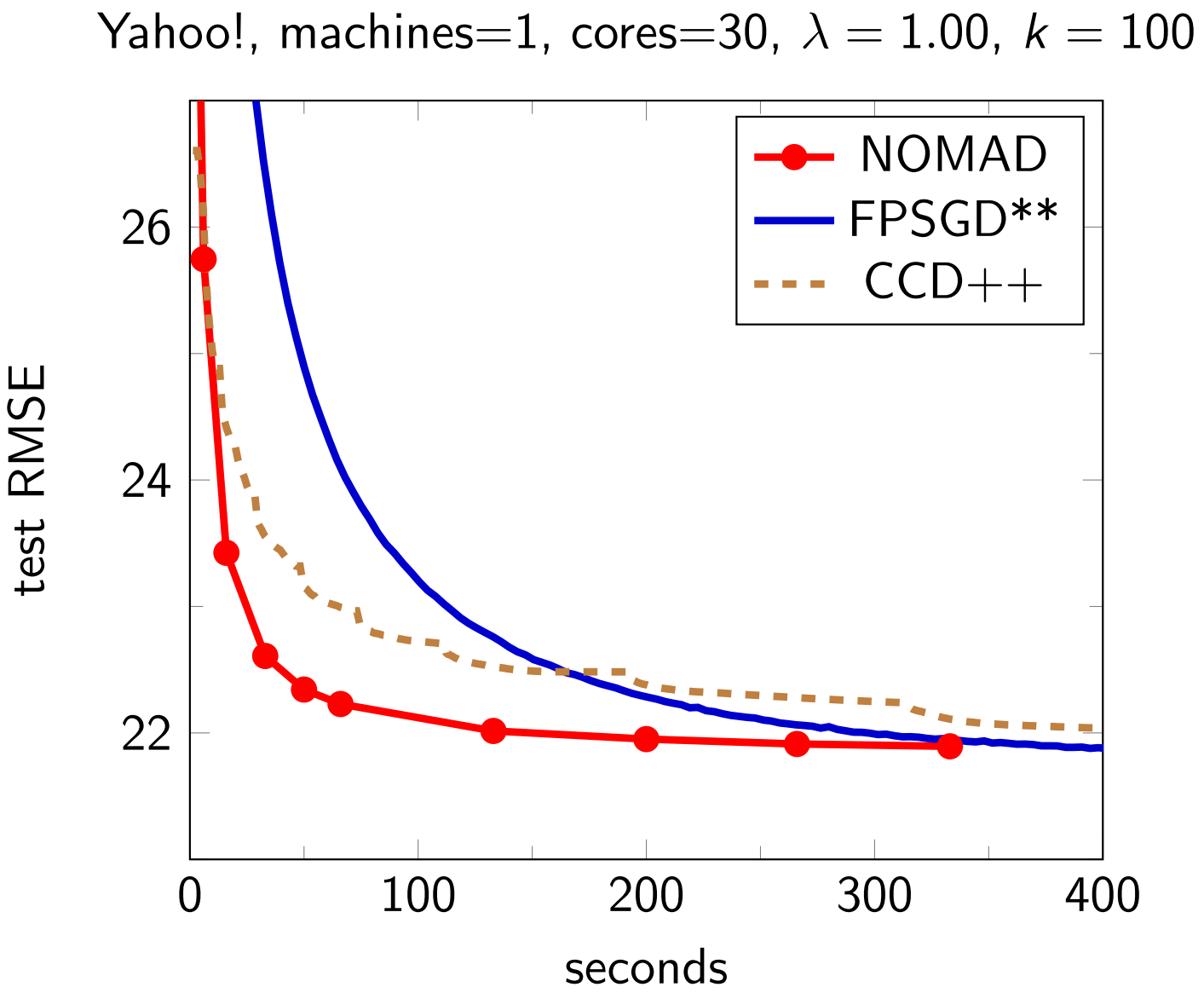
Experiments: Datasets

Name	Rows	Columns	Non-zeros
Netflix	2,649,429	17,770	99,072,112
Yahoo! Music	1,999,990	624,961	252,800,275
Hugewiki	50,082,603	39,780	2,736,496,604

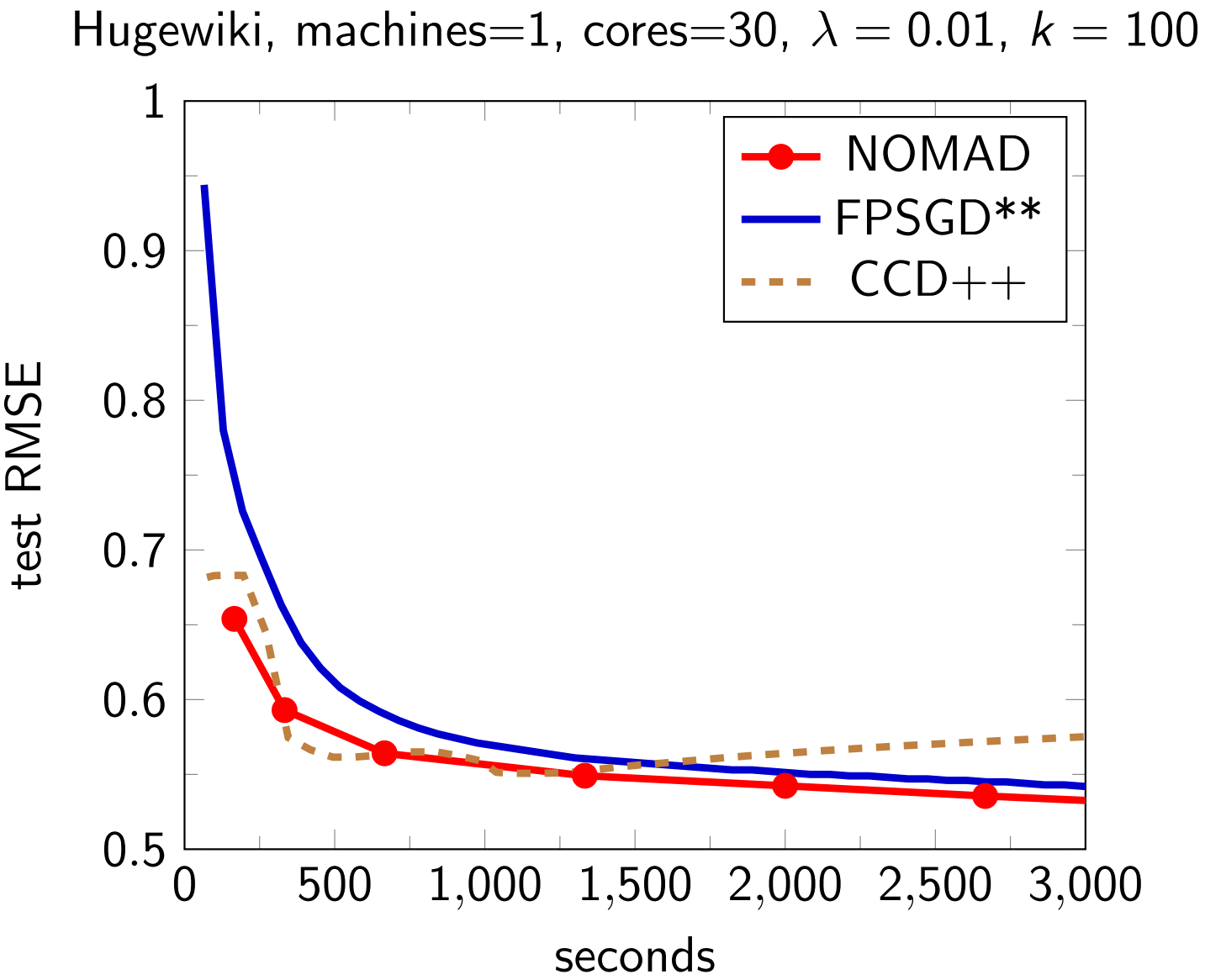
Experiments: Scaling on a single machine



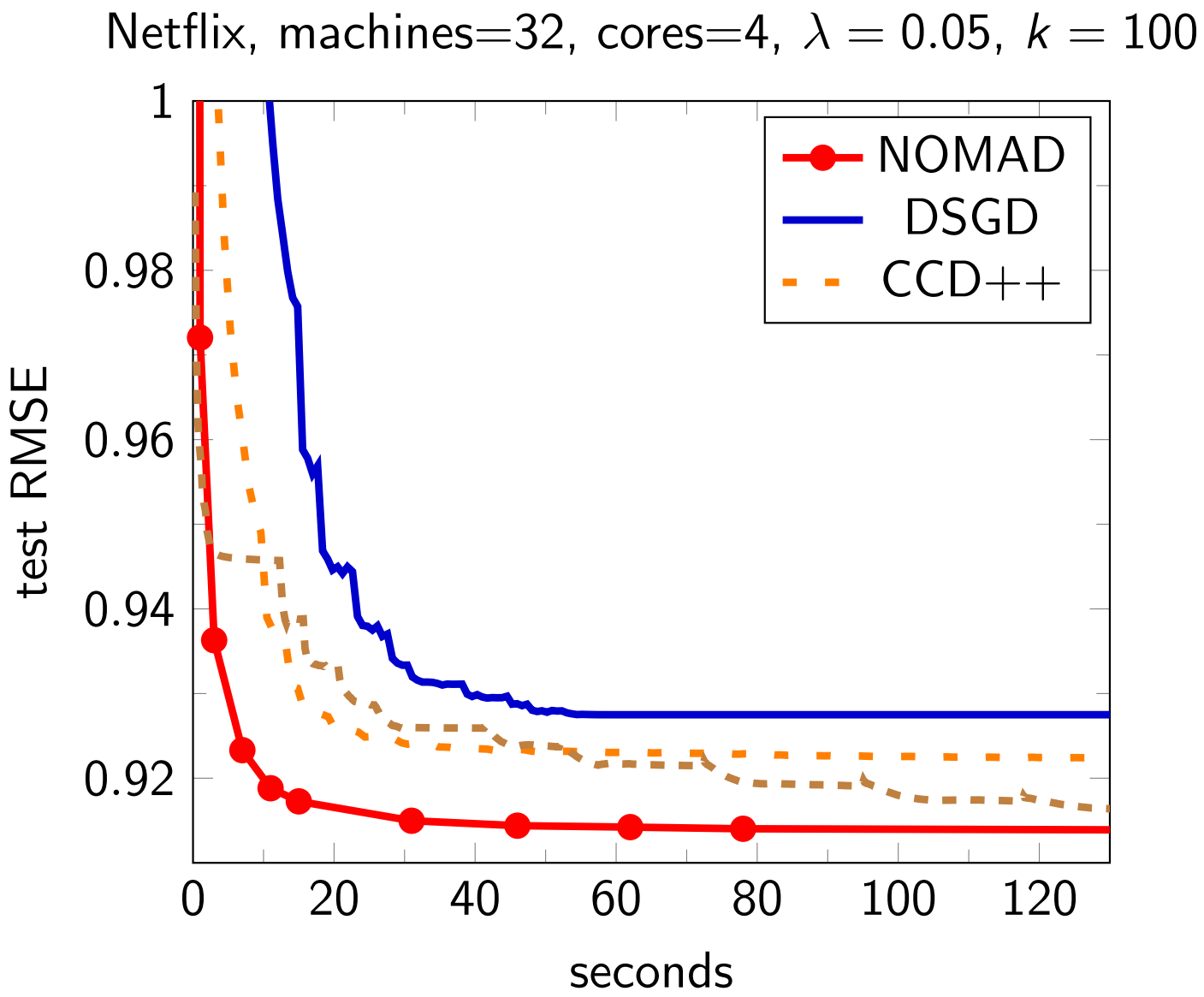
Experiments: Scaling on a single machine



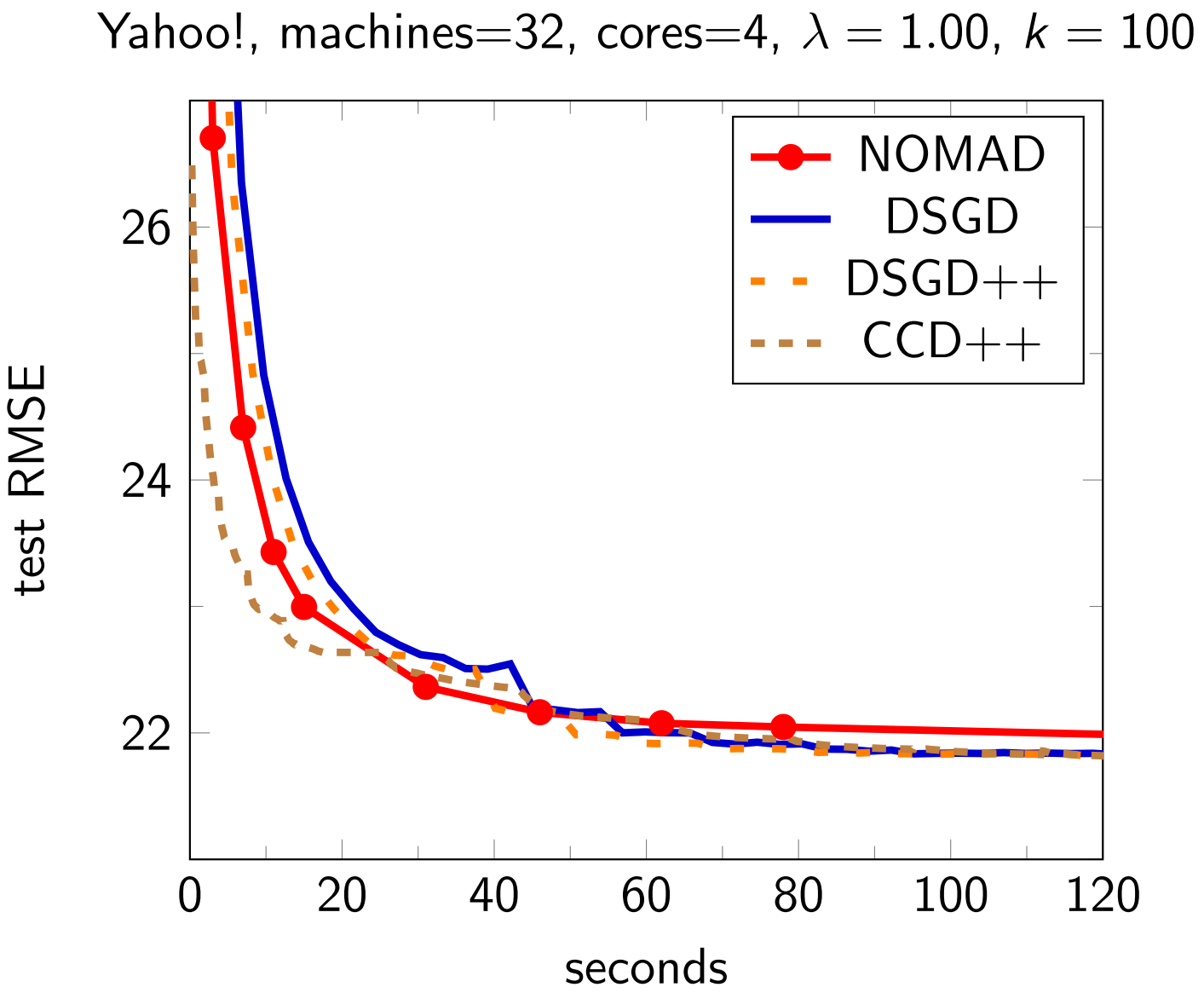
Experiments: Scaling on a single machine



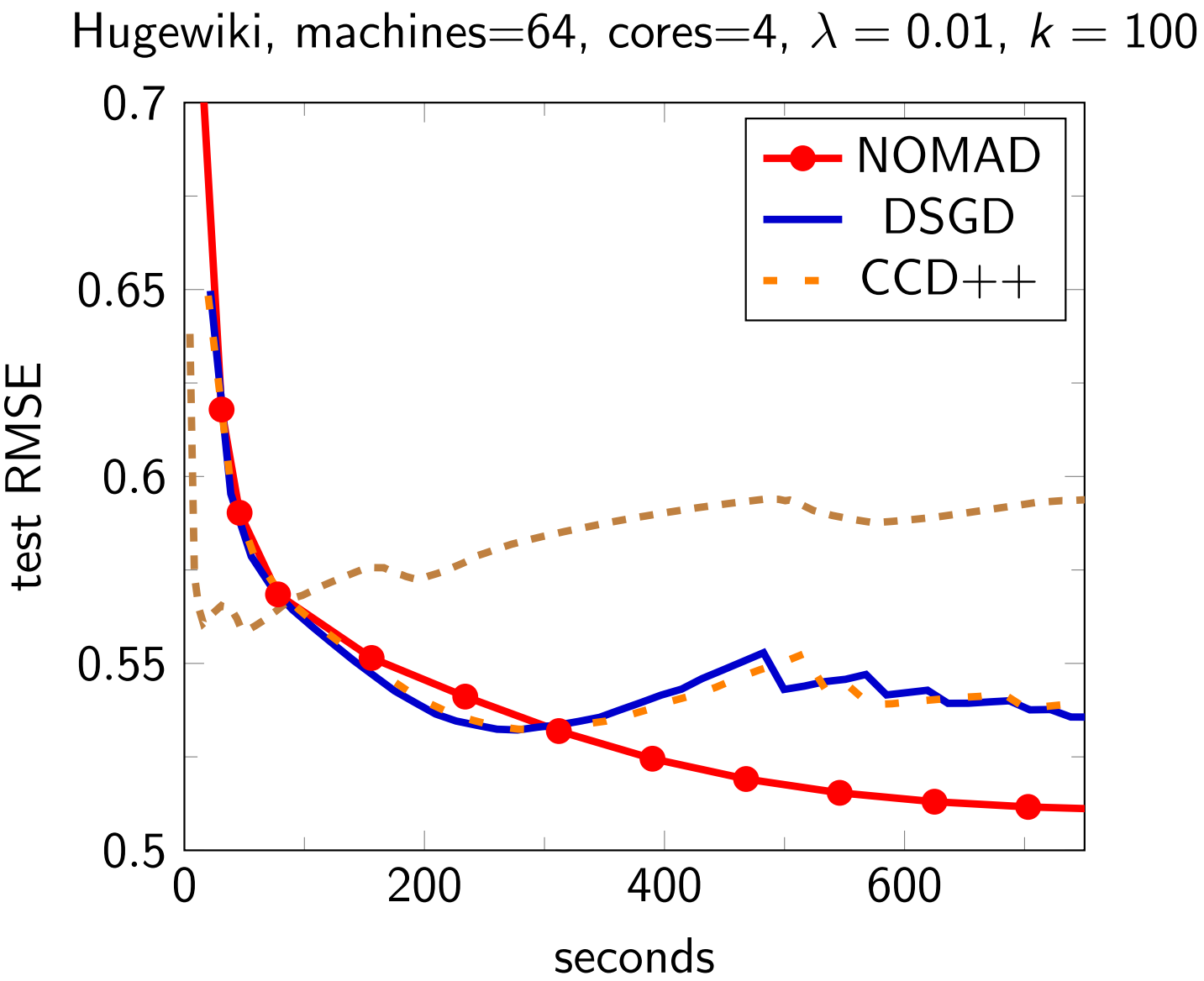
Experiments: Scaling across multiple machines (HPC cluster)



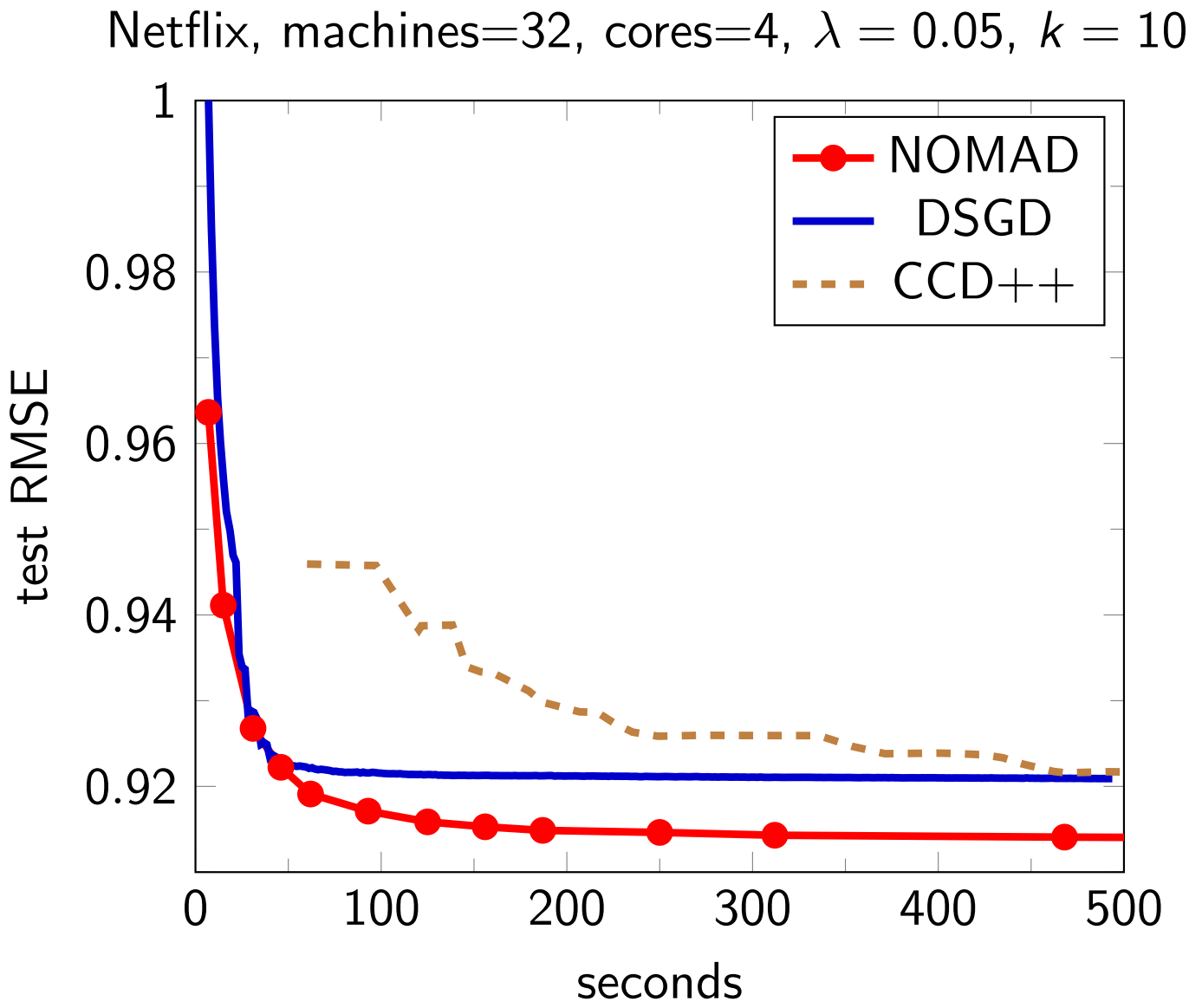
Experiments: Scaling across multiple machines (HPC cluster)



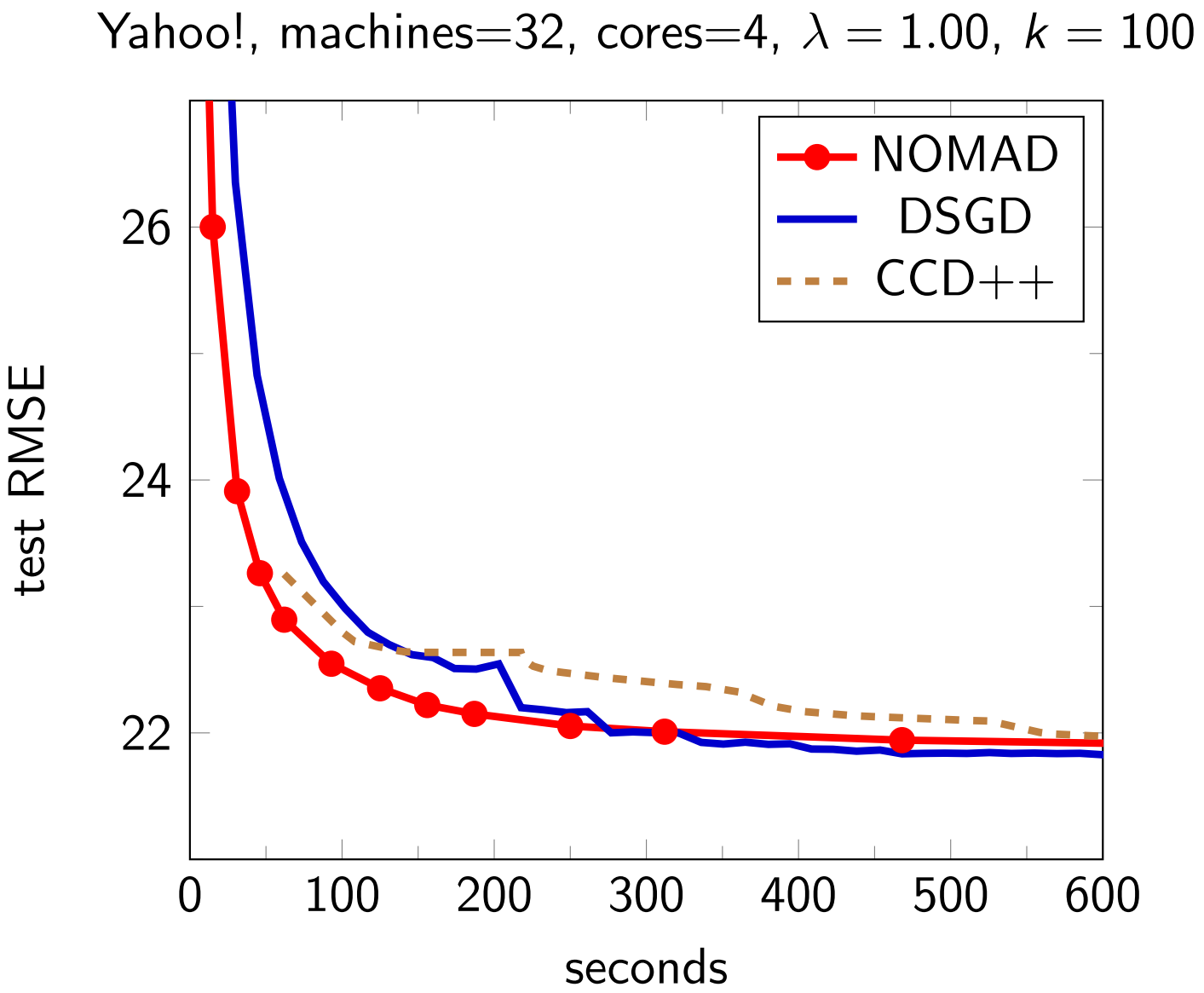
Experiments: Scaling across multiple machines (HPC cluster)



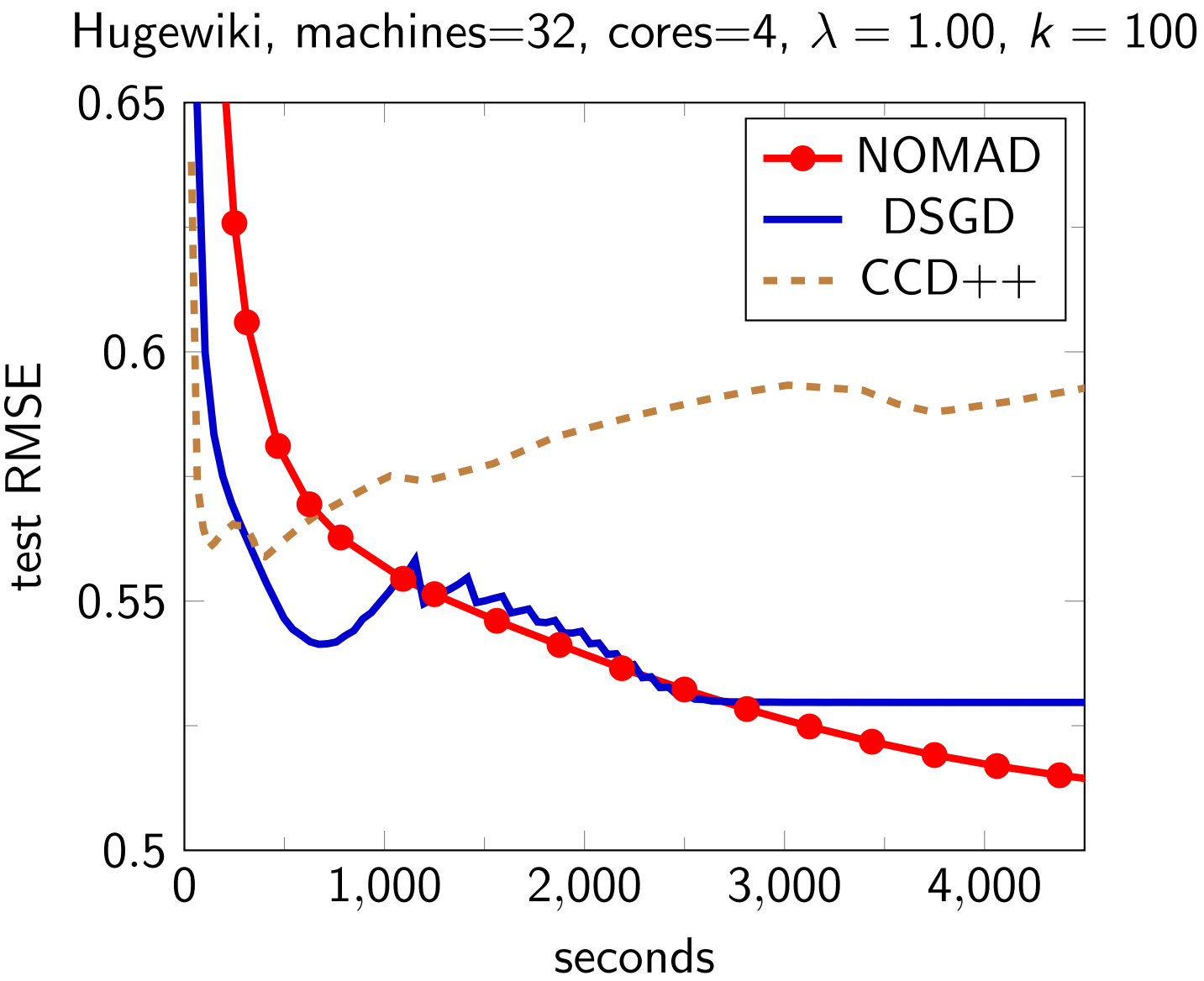
Experiments: Scaling on commodity hardware



Experiments: Scaling on commodity hardware



Experiments: Scaling on commodity hardware



Regularized risk minimization

Machine Learning

- We want to build a model which predicts well on data
- A model's performance is quantified by a loss function
  - a sophisticated discrepancy score
- Our model must generalize to unseen data
- Avoid over-fitting by penalizing *complex* models (Regularization)

More Formally

- Training data: $\{x_1, \ldots, x_m\}$
- Labels: $\{y_1, \ldots, y_m\}$
- Learn a vector: w

$$\underset{w}{\text{minimize}} \; J(w) := \underbrace{\lambda \sum_{j=1}^{d} \phi_j(w_j)}_{\text{Regularizer}} + \underbrace{\frac{1}{m} \sum_{i=1}^{m} l_i(\langle w, x_i \rangle)}_{\text{Risk } R_{\text{emp}}}$$

Stochastic optimization

$$\underset{w}{\text{minimize}} \;\; J(w) := \lambda \sum_{j=1}^{d} \phi_j(w_j) + \frac{1}{m} \sum_{i=1}^{m} l_i(\langle w, x_i \rangle)$$

Stochastic optimization

$$\underset{w}{\text{minimize}} \;\; J_t(w) := \lambda \sum_{j=1}^{d} \phi_j(w_j) + l_t(\langle w, x_t \rangle)$$

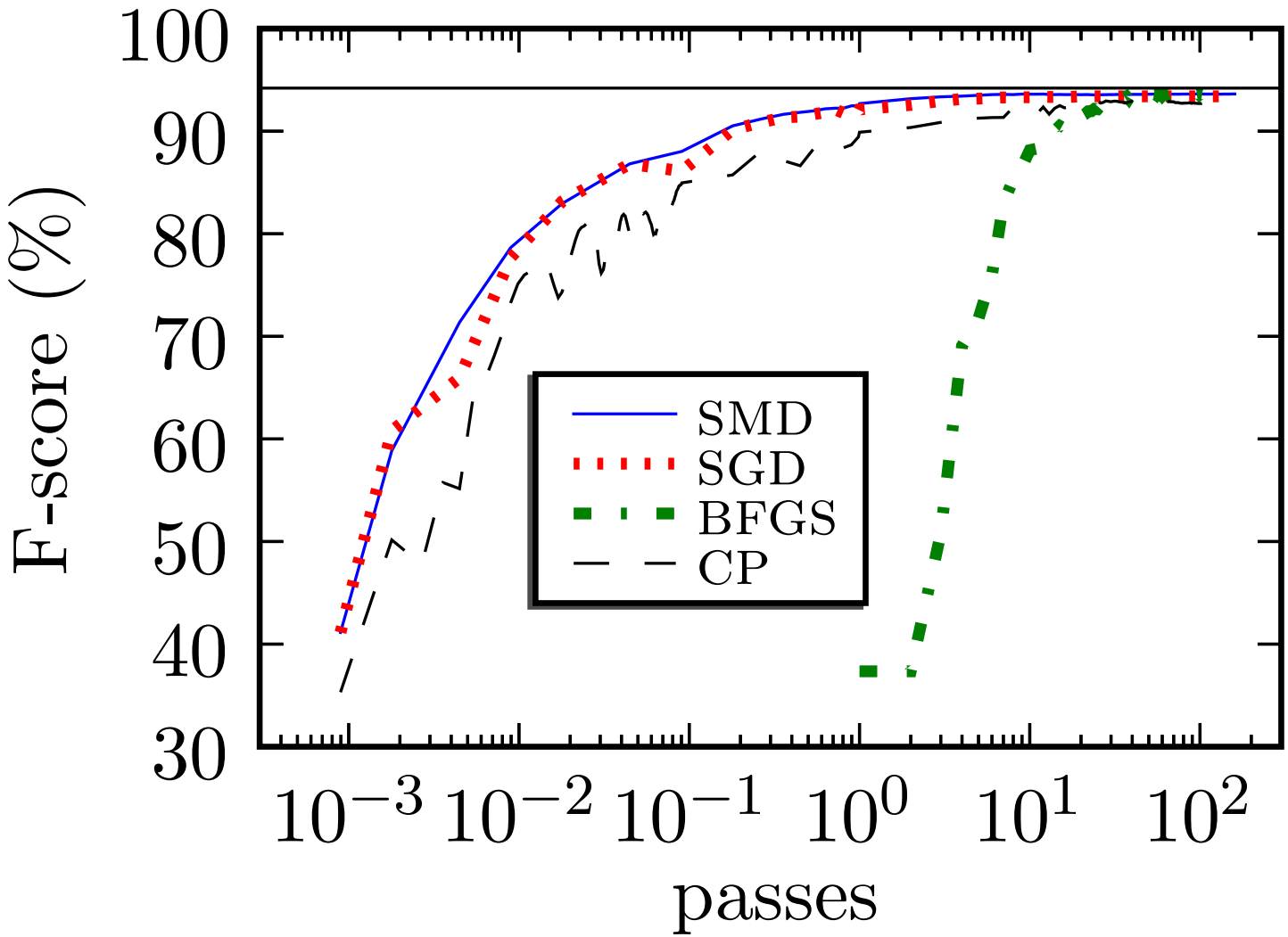
Stochastic optimization

$$J(w) = \mathbb{E}_t \left[J_t(w) \right]$$

Stochastic optimization

$$w_{t+1} \leftarrow w_t - \eta_t \nabla J_t(w_t)$$

Stochastic vs batch optimization



Stochastic optimization

$$w_{t+1} \leftarrow w_t - \eta_t \nabla J_t(w_t)$$

Stochastic optimization

$$w_{t+1} \leftarrow w_t - \eta_t \nabla J_t(w_t)$$

Stochastic optimization

$$w_{t+1} \leftarrow w_t - \eta_t \nabla J_t(w_t)$$

Problem reformulation

$$\min_{w} \quad \lambda \sum_{j=1}^{d} \phi_j(w_j) + \frac{1}{m} \sum_{i=1}^{m} l_i(\langle w, x_i \rangle)$$

Problem reformulation

$$\min_{w,u} \ \lambda \sum_{j=1}^{d} \phi_j(w_j) + \frac{1}{m} \sum_{i=1}^{m} l_i(u_i)$$

$$\text{subject to} \quad u_i = \langle w, x_i \rangle \quad \{i = 1 \dots m\}$$

Problem reformulation

$$\min_{w,u} \max_{\alpha} \; \lambda \sum_{j=1}^{d} \phi_j(w_j) + \frac{1}{m} \sum_{i=1}^{m} l_i(u_i) + \frac{1}{m} \sum_{i=1}^{m} \alpha_i (u_i - \langle w, x_i \rangle)$$

Problem reformulation

$$\max_{\alpha} \min_{w,u} \ \lambda \sum_{j=1}^{d} \phi_j(w_j) + \frac{1}{m} \sum_{i=1}^{m} l_i(u_i) + \frac{1}{m} \sum_{i=1}^{m} \alpha_i (u_i - \langle w, x_i \rangle)$$

Problem reformulation

$$\max_{\alpha} \min_{w} \ \lambda \sum_{j=1}^{d} \phi_j(w_j) - \frac{1}{m} \sum_{i=1}^{m} \alpha_i \langle w, x_i \rangle + \frac{1}{m} \min_{u} \sum_{i=1}^{m} \left(l_i(u_i) + \alpha_i u_i \right)$$

Problem reformulation

$$\max_{\alpha} \min_{w} \ \lambda \sum_{j=1}^{d} \phi_j(w_j) - \left\langle w, \frac{1}{m} \sum_{i=1}^{m} \alpha_i x_i \right\rangle + \frac{1}{m} \sum_{i=1}^{m} l_i^*(-\alpha_i)$$

Problem reformulation

$$\max_{\alpha} \min_{w} \quad f(w,\alpha) := \sum_{i=1}^{m} \sum_{j \in \Omega_i} \underbrace{\left(\frac{\lambda}{|\bar{\Omega}_j|} \phi_j(w_j) - \frac{1}{m} \alpha_i w_j x_{ij} + \frac{1}{m\,|\Omega_i|} l_i^*(-\alpha_i) \right)}_{\frac{1}{|\Omega|} f_{ij}(w_j, \alpha_i)}$$

Stochastic gradients

$$f(w, \alpha) = \mathbb{E}_{ij}\left[f_{ij}(w_j, \alpha_i)\right].$$

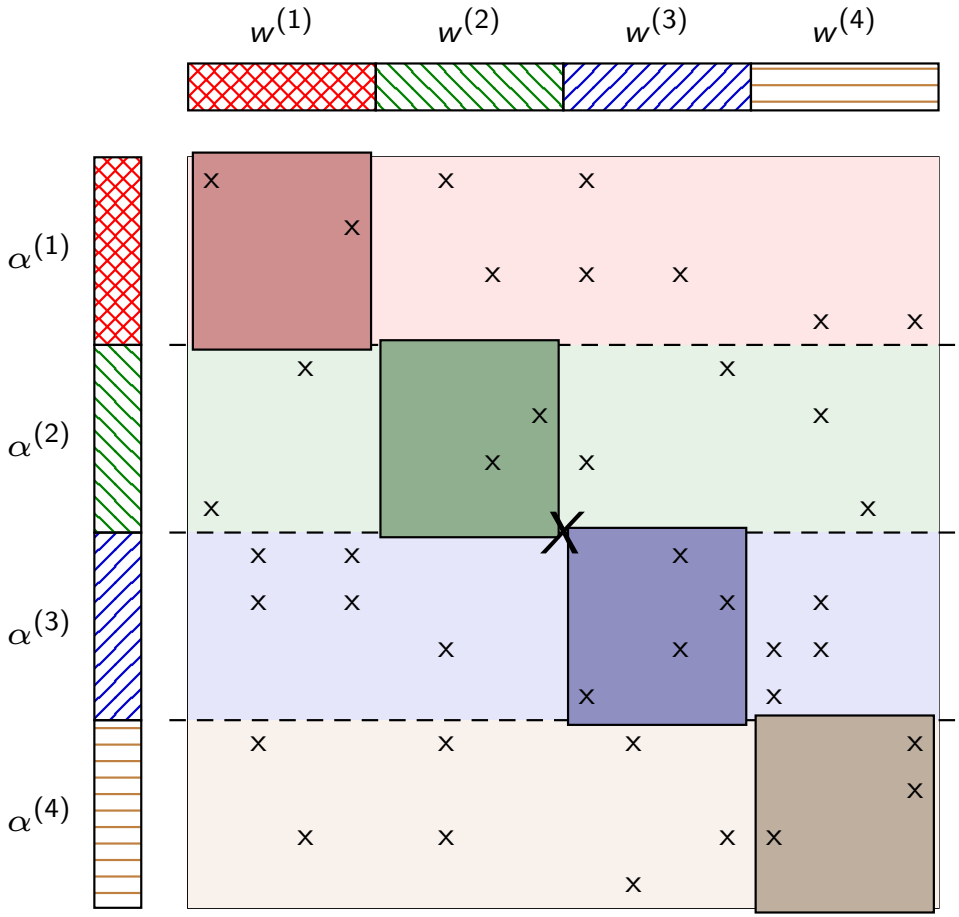
$$\partial_{w_j} f_{ij}(w_j, \alpha_i) = |\Omega| \left(\frac{\lambda}{|\bar{\Omega}_j|} \partial \phi_j(w_j) - \frac{1}{m} \alpha_i x_{ij} \right)$$

$$\partial_{\alpha_i} f_{ij}(w_j, \alpha_i) = |\Omega| \left(-\frac{1}{m} w_j x_{ij} + \frac{1}{m\,|\Omega_i|} \partial_{\alpha_i} l_i^*(-\alpha_i) \right).$$

$$w_j \leftarrow w_j - \eta_w |\Omega| \left(\frac{\lambda}{|\bar{\Omega}_j|} \partial \phi_j(w_j) - \frac{1}{m} \alpha_i x_{ij} \right)$$

$$\alpha_i \leftarrow \alpha_i + \eta_\alpha |\Omega| \left(-\frac{1}{m} w_j x_{ij} + \frac{1}{m\,|\Omega_i|} \partial_{\alpha_i} l_i^*(-\alpha_i) \right).$$

Decoupling the updates



Also see Gemulla et al., 2011 and Re and Recht, 2012.

Convergence: Assumptions

Assumptions

Suppose, DSSO is run with $p \leq \min(m, d)$ processors, and let (w^t, α^t) and $(\tilde{w}^t, \tilde{\alpha}^t) := \left(\frac{1}{t}\sum_{s=1}^{t} w^s, \frac{1}{t}\sum_{s=1}^{t} \alpha^s\right)$ denote the parameter values, and the averaged parameter values respectively after the t-th epoch. Moreover, assume that

Convergence: Assumptions

Assumptions

- There exists a partitioning of Ω such that $\left|\Omega^{(q,\sigma_r(q))}\right| \approx \frac{1}{p^2}\left|\Omega\right|$ and $|J_q| \approx \frac{d}{p}$ for all q.
- Performing updates takes constant amount of time denoted by T_{u}.
- Communicating w across the network takes constant amount of time denoted by T_{c}, and communicating a subset of w takes time proportional to its cardinality.
- The diameter of the search space is bounded. In other words, there exists a constant D such that for all (w, α) and (w', α') we have $\|w - w'\|^2 + \|\alpha - \alpha'\|^2 \leq D$.
- The gradients of $f_{i,j}(w_j, \alpha_i)$ are bounded by a constant that does not depend on i or j.

$$\|\nabla_w f_{i,j}(w_j, \alpha_i)\|^2 \leq C_w^2 \text{ and } \|\nabla_\alpha - f_{i,j}(w_j, \alpha_i)\|^2 \leq C_\alpha^2.$$

Convergence: Theorem

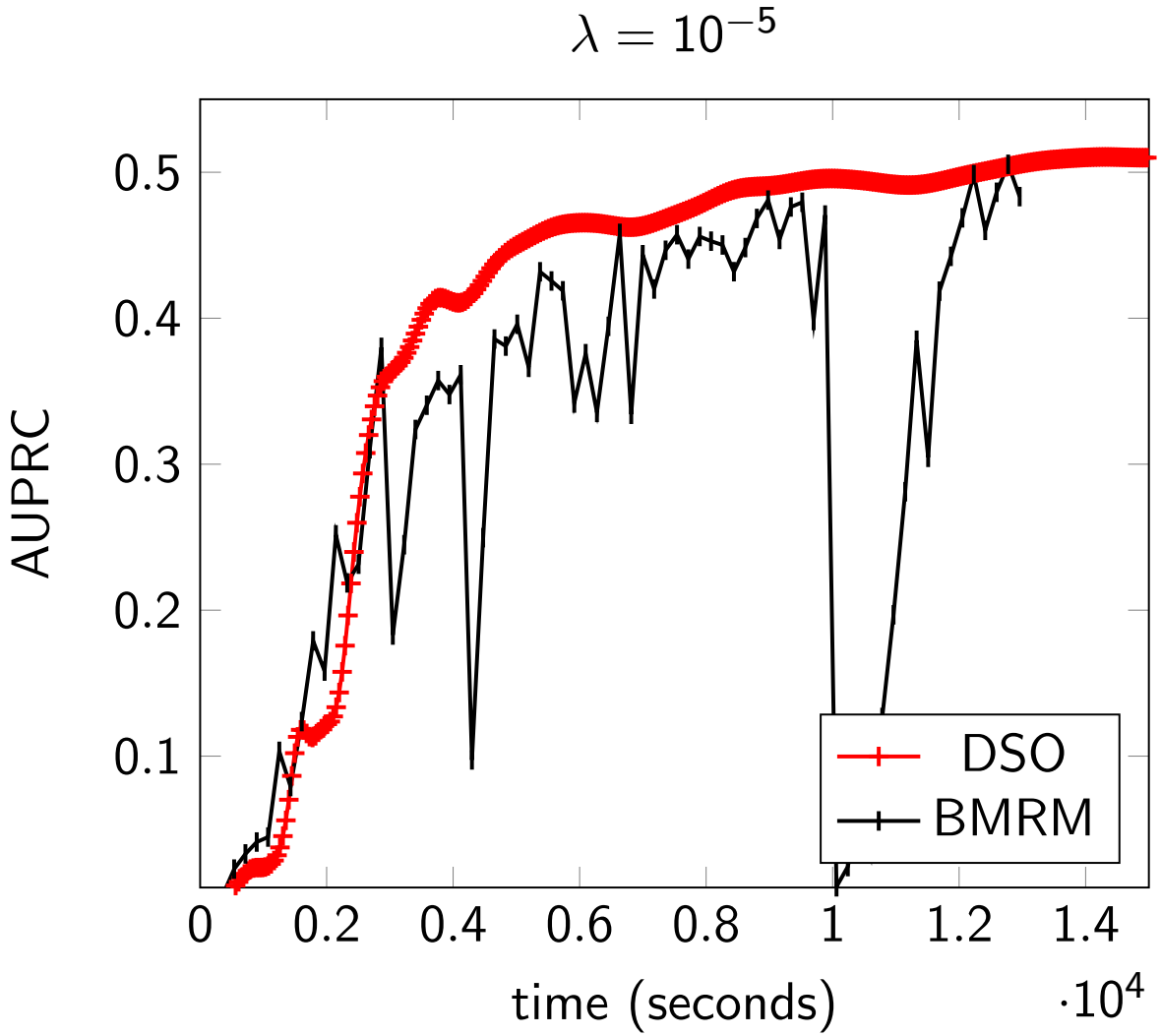
Theorem

There exists a constant C such that after $\left(\frac{|\Omega| T_{\mathrm{u}}}{p} + T_{\mathrm{c}}\right) T$ time, the duality gap is

$$\varepsilon(w, \alpha) := \max_{\alpha'} f\left(\tilde{w}^T, \alpha'\right) - \min_{w'} f\left(w', \tilde{\alpha}^T\right) \leq \sqrt{\frac{2DC}{T}}. \tag{1}$$

Experiment

SVM training with 50 million data points and 11.7 million dimensions.



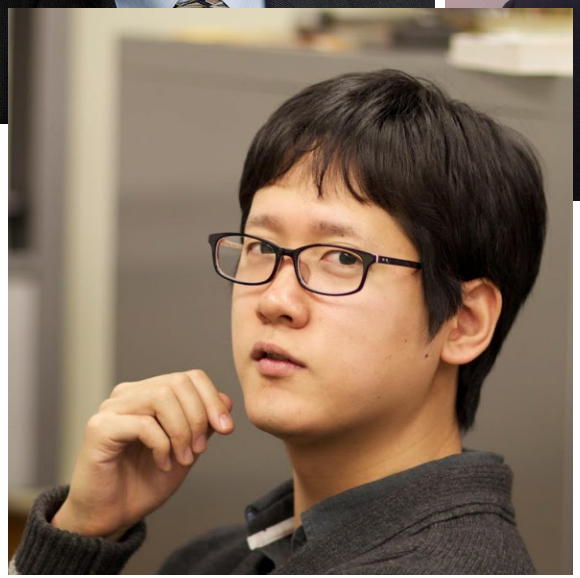
References

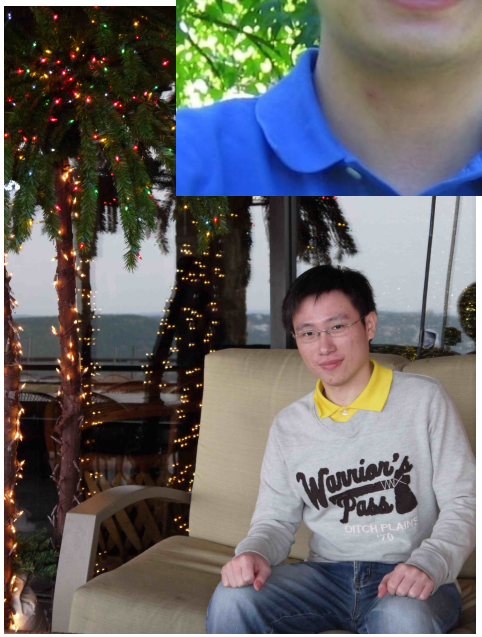
- Shihao Ji, Hyokun Yun, Pinar Yanardag, Shin Matsushima, and S. V. N. Vishwanathan *WordRank:Learning Word Embeddings via Robust Ranking*. Submitted to NIPS 2015.
- Shin Matsushima, Hyokun Yun, and S. V. N. Vishwanathan. *Doubly Separable Stochastic Optimization for Regularized Empirical Risk Minimization*. Submitted to NIPS 2015.
- Hsiang-Fu Yu, Cho-Jui Hsieh, Hyokun Yun, S. V. N. Vishwanathan, and Inderjit Dhillon. *A Scalable Asynchronous Distributed Algorithm for Topic Modeling*. WWW 2015.
- 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*. VLDB 2014.
- Hyokun Yun, Parameshwaran Raman, and S. V. N. Vishwanathan. *Ranking via Robust Binary Classification and Parallel Parameter Estimation in Large-Scale Data*. NIPS 2014.

Collaborators









Questions?