NOMAD: Non-locking, stOchastic Multi-machine algorithm for Asynchronous and Decentralized matrix factorization
Scaling by Exploiting Structure

S.V.N. (vishy) Vishwanathan
Purdue University and Amazon
vishy@{purdue.edu,amazon.com}

July 1st, 2013
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: \( w \)

\[
\minimize_w J(w) := \lambda \sum_{j=1}^{d} \phi_j(w_j) + \frac{1}{m} \sum_{i=1}^{m} l(\langle w, x_i \rangle, y_i)
\]

- Regularizer
- Risk \( R_{emp} \)
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: \( w \)

\[
\text{minimize } J(w) := \lambda \sum_{j=1}^{d} \phi_j(w_j) + \frac{1}{m} \sum_{i=1}^{m} l(\langle w, x_i \rangle, y_i)
\]

- Regularizer
- Risk \( R_{\text{emp}} \)
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: \(w\)

\[
\min_w J(w) := \lambda \sum_{j=1}^{d} \phi_j(w_j) + \frac{1}{m} \sum_{i=1}^{m} l(\langle w, x_i \rangle, y_i)
\]

Where \(\phi_j\) is the regularizer and \(l\) is the risk function. This formulation balances the regularizer and the empirical risk.
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: \( w \)

\[
\begin{align*}
\text{minimize } J(w) := & \lambda \sum_{j=1}^{d} \phi_j(w_j) + \frac{1}{m} \sum_{i=1}^{m} l(\langle w, x_i \rangle, y_i) \\
\text{Regularizer} & \quad \text{Risk } R_{\text{emp}}
\end{align*}
\]
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: \( w \)

\[
\text{minimize } J(w) := \lambda \sum_{j=1}^{d} \phi_j(w_j) + \frac{1}{m} \sum_{i=1}^{m} l(\langle w, x_i \rangle, y_i)
\]

- Regularizer
- Risk \( R_{\text{emp}} \)
Outline

1. NOMAD for Matrix Completion

2. NOMAD for Regularized Risk Minimization
Collaborative filtering

<table>
<thead>
<tr>
<th></th>
<th>$M_1$</th>
<th>$M_2$</th>
<th>$M_3$</th>
<th>$M_4$</th>
<th>$M_5$</th>
<th>$M_6$</th>
<th>$M_7$</th>
<th>$M_8$</th>
<th>$M_9$</th>
<th>$M_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>$U_2$</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>$U_3$</td>
<td></td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>$U_4$</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>$U_5$</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>$U_6$</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✗</td>
</tr>
</tbody>
</table>
Matrix completion

\[ A \approx WH \]
Matrix completion

\[
\begin{align*}
\min_{W \in \mathbb{R}^{m \times k}, H \in \mathbb{R}^{n \times k}} f(W, H),
\end{align*}
\]

\[
f(W, H) = \frac{1}{2} \sum_{(i,j) \in \Omega} \left( A_{ij} - w_i^T h_j \right)^2 + \lambda \left( \|w_i\|^2 + \|h_j\|^2 \right).
\]

\[
\begin{array}{c}
\text{loss} \\
\text{Regularizer}
\end{array}
\]
Stochastic approximation

\[ f(W, H) \approx f_n(W, H) = \frac{1}{2} \left\{ (A_{ij} - w_i^T h_j)^2 + \lambda (\|w_i\|^2 + \|h_j\|^2) \right\} \]

\[ \nabla_{w_{i'}} f_n(W, H) = \begin{cases} (A_{ij} - w_i^T h_j) h_j + \lambda w_i, & \text{for } i = i' \\ 0, & \text{otherwise} \end{cases} \]

\[ \nabla_{h_{j'}} f_n(W, H) = \begin{cases} (A_{ij} - w_i^T h_j) w_i + \lambda h_j, & \text{for } j = j' \\ 0, & \text{otherwise} \end{cases} \]
Stochastic updates

\[
\begin{align*}
    w_i & \leftarrow w_i - \eta \left( (A_{ij} - w_i^T h_j) h_j + \lambda w_i \right) \\
    h_j & \leftarrow h_j - \eta \left( (A_{ij} - w_i^T h_j) w_i + \lambda h_j \right)
\end{align*}
\]
Decoupling the updates [Gemulla et al., KDD 2011]
Decoupling the updates [Gemulla et al., KDD 2011]

Synchronize and Communicate
Decoupling the updates [Gemulla et al., KDD 2011]
Synchronize and Communicate
Decoupling the updates [Gemulla et al., KDD 2011]
Decoupling the updates [Gemulla et al., KDD 2011]

Synchronize and Communicate
Decoupling the updates [Gemulla et al., KDD 2011]
Some Observations (also see Gemulla et al, ICDM 2012)

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
Some Observations (also see Gemulla et al, ICDM 2012)

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?
Non-locking, stOchastic Multi-machine algorithm for Asynchronous and Decentralized matrix factorization (NOMAD)
Illustration of NOMAD communication
Illustration of NOMAD communication
Illustration of NOMAD communication

Diagram showing the communication patterns and data exchanges in NOMAD for Matrix Completion.
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: Netflix

- Size: $2649429 \times 17770$, nnz=99 million

![Graph showing speedup vs. number of machines for Netflix experiments](image-url)
Experiments: Netflix

1 Processor

Multiple Processors

S.V. N. Vishwanathan (Purdue and Amazon)
Experiments: Netflix

<table>
<thead>
<tr>
<th>Processor</th>
<th>elapsed secs × num processors</th>
<th>test RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Processor</td>
<td>500, 1,000, 1,500</td>
<td>1.1, 1.05, 1.00</td>
</tr>
<tr>
<td>Multiple Processors</td>
<td>500, 1,000, 1,500</td>
<td>1.15, 1.10, 1.05</td>
</tr>
</tbody>
</table>
Experiments: Netflix

1 Processor

Multiple Processors

1 Processor

8 Processors
Experiments: Netflix

1 Processor

<table>
<thead>
<tr>
<th>Elapsed secs × num processors</th>
<th>Test RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>1.1</td>
</tr>
<tr>
<td>1,000</td>
<td>1.05</td>
</tr>
<tr>
<td>1,500</td>
<td>1.01</td>
</tr>
</tbody>
</table>

Multiple Processors

<table>
<thead>
<tr>
<th>Elapsed secs × num processors</th>
<th>Test RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>1.15</td>
</tr>
<tr>
<td>1,000</td>
<td>1.1</td>
</tr>
<tr>
<td>1,500</td>
<td>1.05</td>
</tr>
</tbody>
</table>
Experiments: Netflix

1 Processor

Multiple Processors

1 Processor

30 Processors

elapsed secs × num processors

elapsed secs × num processors
Experiments: Yahoo! music

- Size: $1000990 \times 624961$, nnz=252 million

![Graph showing speedup vs. number of machines for Yahoo! music experiments.](image)
Experiments: Yahoo! music

**1 Processor**

- **1 Processor**
  - Test RMSE vs. elapsed secs × num processors

**Multiple Processors**

- **2 Processors**
  - Test RMSE vs. elapsed secs × num processors
Experiments: Yahoo! music

1 Processor

Multiple Processors
Experiments: Yahoo! music

1 Processor

Multiple Processors
Experiments: Yahoo! music

1 Processor

Multiple Processors

1 Processor

500 1,000 1,500
elapsed secs × num processors

test RMSE

23 24 25

15 Processors

500 1,000 1,500
elapsed secs × num processors

test RMSE

23 24 25 26
Experiments: Yahoo! music

1 Processor

Multiple Processors

S.V. N. Vishwanathan (Purdue and Amazon)  Decentralized Optimization for ML
Experiments: Synthetic Data

- Size: 5,000,000 × 200,000, nnz=270 million

![Graph showing speedup vs. number of machines for synthetic data. The graph shows a linear increase in speedup as the number of machines increases.]
Experiments: Synthetic data

1 Processor

Multiple Processors
Experiments: Synthetic data

1 Processor

Multiple Processors

Test RMSE vs. elapsed secs × num processors for 1 Processor and 4 Processors.
Experiments: Synthetic data

1 Processor

Multiple Processors

S.V. N. Vishwanathan (Purdue and Amazon)
Experiments: Synthetic data

1 Processor

Multiple Processors

elapsed secs × num processors

test RMSE

elapsed secs × num processors

test RMSE
Outline

1. NOMAD for Matrix Completion

2. NOMAD for Regularized Risk Minimization
Problem Formulation

\[ \min_w \lambda \sum_{j=1}^{d} \phi_j(w_j) + \frac{1}{m} \sum_{i=1}^{m} l(\langle w, x_i \rangle, y_i) \]
Problem Formulation

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

subject to  \quad u_i = \langle w, x_i \rangle \quad \{i = 1 \ldots m\}
Problem Formulation

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

\[
\max_{\alpha} \min_{w,u} \quad \lambda \sum_{j=1}^{d} \phi_j(w_j) + \frac{1}{m} \sum_{i=1}^{m} l(u_i) + \frac{1}{m} \sum_{i=1}^{m} \alpha_i (u_i - \langle w, x_i \rangle)
\]
Problem Formulation

\[
\max_{\alpha} \min_w \lambda \sum_{j=1}^{d} \phi_j(w_j) - \frac{1}{m} \sum_{i=1}^{m} \alpha_i \langle w, x_i \rangle + \frac{1}{m} \min_u \sum_{i=1}^{m} (l(u_i) + \alpha_i u_i)
\]
Problem Formulation

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

\[
\max_{\alpha} \min_w \sum_{i=1}^{m} \sum_{j \in \Omega_i} \left( \frac{\lambda}{|\bar{\Omega}_j|} \phi_j(w_j) - \frac{1}{m} \alpha_i w_j x_{ij} + \frac{1}{m|\Omega_i|} I^*(-\alpha_i) \right)
\]
Stochastic Gradients

\[ \partial_{w_j} J(w, \alpha) = |\Omega| \left( \frac{\lambda}{|\Omega_j|} \partial \phi_j(w_j) - \frac{1}{m} \alpha_i x_{ij} \right) \]

\[ \partial_{\alpha_i} J(w, \alpha) = |\Omega| \left( -\frac{1}{m} w_j x_{ij} + \frac{1}{m |\Omega_j|} \partial \alpha_i l^*(-\alpha_i) \right). \]
Decoupling the updates
NOMAD for Regularized Risk Minimization

It Converges!

KDDA Test Dataset

- 10 percent coordinates
- all coordinates

Dual gap vs. number of iterations for KDDA Test Dataset with 10 percent coordinates and all coordinates.
Joint work with

NOMAD for Regularized Risk Minimization

S.V. N. Vishwanathan (Purdue and Amazon)