Scaling Up
Graphical Model Inference
Graphical Models

• View observed data and unobserved properties as \textit{random variables}
• Graphical Models: compact graph-based encoding of probability distributions (high dimensional, with complex dependencies)
  
- Bayesian Networks (directed), Markov Random Fields (undirected), hybrids, extensions, etc.  HMM, CRF, RBM, M^3N, HMRF, etc.

• Generative/discriminative/hybrid, un-,semi- and supervised learning
  
  – Bayesian Networks (directed), Markov Random Fields (undirected), hybrids, extensions, etc.  HMM, CRF, RBM, M^3N, HMRF, etc.

• Enormous research area with a number of excellent tutorials
  
  – [J98], [M01], [M04], [W08], [KF10], [S11]
Graphical Model Inference

• Key issues:
  – **Representation**: syntax and semantics (directed/undirected, variables/factors,..)
  – **Inference**: computing probabilities and most likely assignments/explanations
  – **Learning**: of model parameters based on observed data. *Relies on inference!*

• Inference is NP-hard (numerous results, incl. approximation hardness)

• Exact inference: works for very limited subset of models/structures
  – E.g., chains or low-treewidth trees

• Approximate inference: highly computationally intensive
  – Deterministic: variational, *loopy belief propagation*, expectation propagation
  – Numerical sampling (Monte Carlo): *Gibbs sampling*
Inference in Undirected Graphical Models

• Factor graph representation

\[ p(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{x_j \in N(x_i)} \psi_{ij}(x_1, x_2) \]

• Potentials capture compatibility of related observations
  – e.g., \( \psi(x_i, x_j) = \exp(-b|x_i - x_j|) \)

• Loopy belief propagation = message passing
  – iterate (read, update, send)

\[
\begin{align*}
  m_{X_i \rightarrow \psi_A}(x_i) &\propto \prod_{\psi_B \in N[X_i] \setminus \psi_A} m_{\psi_B \rightarrow X_i}(x_i) \\
  m_{\psi_A \rightarrow X_i}(x_i) &\propto \sum_{x_A \setminus x_i} \psi_A(x_A) \prod_{X_k \in N[\psi_A] \setminus X_i} m_{X_k \rightarrow \psi_A}(x_k)
\end{align*}
\]
Synchronous Loopy BP

- Natural parallelization: associate a processor to every node
  - Simultaneous receive, update, send
- Inefficient – e.g., for a linear chain:

\[
\frac{2n^2}{p} \text{ time per iteration} \\
\leq 2n \text{ iterations to converge}
\]

- Naturally Parallel
- Forward-Backward
- Optimal Parallel

[SUML-Ch10]
Optimal Parallel Scheduling

- Partition, local forward-backward for center, then cross-boundary

Partitioning and scheduling components across processors: Processor 1, Processor 2, Processor 3.

Synchronous Schedule

Optimal Schedule

Parallel Component
Sequential Component

Optimal Schedule

\[ O \left( \frac{n \tau_p}{p} + \mathcal{T}_\epsilon \right) \]

Gap
Splash: Generalizing Optimal Chains

1) Select root, grow fixed-size BFS Spanning tree
2) Forward Pass computing all messages at each vertex
3) Backward Pass computing all messages at each vertex

- Parallelization:
  - Partition graph
    - Maximize computation, minimize communication
    - Over-partition and randomly assign
  - Schedule multiple Splashes
    - Priority queue for selecting root
    - Belief residual: cumulative change from inbound messages
    - Dynamic tree pruning
DBRSplash: MLN Inference Experiments

- Experiments: MLN Inference
- 8K variables, 406K factors
- Single-CPU runtime: 1 hour
- Cache efficiency critical

- 1K variables, 27K factors
- Single-CPU runtime: 1.5 minutes
- Network costs limit speedups
Topic Models

- Goal: unsupervised detection of topics in corpora
  - Desired result: topic mixtures, per-word and per-document topic assignments

<table>
<thead>
<tr>
<th>“Arts”</th>
<th>“Budgets”</th>
<th>“Children”</th>
<th>“Education”</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW</td>
<td>MILLION</td>
<td>CHILDREN</td>
<td>SCHOOL</td>
</tr>
<tr>
<td>FILM</td>
<td>TAX</td>
<td>WOMEN</td>
<td>STUDENTS</td>
</tr>
<tr>
<td>SHOW</td>
<td>PROGRAM</td>
<td>PEOPLE</td>
<td>SCHOOLS</td>
</tr>
<tr>
<td>MUSIC</td>
<td>BUDGET</td>
<td>CHILD</td>
<td>EDUCATION</td>
</tr>
<tr>
<td>MOVIE</td>
<td>BILLION</td>
<td>YEARS</td>
<td>TEACHERS</td>
</tr>
<tr>
<td>PLAY</td>
<td>FEDERAL</td>
<td>FAMILIES</td>
<td>HIGH</td>
</tr>
</tbody>
</table>

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center’s share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.
Directed Graphical Models: Latent Dirichlet Allocation [B+03, SUML-Ch11]

- Generative model for document collections
  - $K$ topics, topic $k$: Multinomial($\phi_k$) over words
  - $D$ documents, document $j$:
    - Topic distribution $\theta_j \sim$ Dirichlet($\alpha$)
    - $N_j$ words, word $x_{ij}$:
      - Sample topic $z_{ij} \sim$ Multinomial($\theta_j$)
      - Sample word $x_{ij} \sim$ Multinomial($\phi_{z_{ij}}$)
- Goal: infer posterior distributions
  - Topic word mixtures $\{\phi_k\}$
  - Document mixtures $\{\theta_j\}$
  - Word-topic assignments $\{z_{ij}\}$
Gibbs Sampling

• Full joint probability

\[ p(\theta, z, \phi, x | \alpha, \beta) = \prod_{k=1..K} p(\phi_k | \beta) \prod_{j=1..D} p(\theta_j | \alpha) \prod_{j=1..N_j} p(z_{ij} | \theta_j)p(x_{ij} | \phi_{z_{ij}}) \]

• Gibbs sampling: sample \( \phi, \theta, z \) independently

• Problem: slow convergence (a.k.a. mixing)

• Collapsed Gibbs sampling
  – Integrate out \( \phi \) and \( \theta \) analytically

\[ p(z|x, d, \alpha, \beta) \propto \frac{N_{xz} + \beta}{\sum_{x}(N_{xz} + \beta)} \frac{N_{dz} + \alpha}{\sum_{z}(N_{dz} + \alpha)} \]

  – Until convergence:
    • resample \( p(z_{ij} | x_{ij}, \alpha, \beta) \),
    • update counts: \( N_z, N_{zd}, N_{xz} \)
Parallel Collapsed Gibbs Sampling [SUML-Ch11]

• Synchronous version (MPI-based):
  – Distribute documents among $p$ machines
  – Global topic and word-topic counts $N_z, N_{wz}$
  – Local document-topic counts $N_{dz}$
  – After each local iteration, AllReduce $N_z, N_{wz}$

• Asynchronous version: gossip (P2P)
  – Random pairs of processors exchange statistics upon pass completion
  – Approximate global posterior distribution (experimentally not a problem)
  – Additional estimation to properly account for previous counts from neighbor
Parallel Collapsed Gibbs Sampling [SN10,S11]

- Multithreading to maximize concurrency
  - Parallelize both *local* and *global* updates of $N_{xz}$ counts
  - Key trick: $N_z$ and $N_{xz}$ are effectively constant for a given document
    - No need to update continuously: update once per-document *in a separate thread*
    - Enables multithreading the samplers
  - Global updates are asynchronous -> no blocking

---

<table>
<thead>
<tr>
<th></th>
<th>Google LDA</th>
<th>Mallet</th>
<th>Irvine’08</th>
<th>Irvine’09</th>
<th>Yahoo LDA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multicore</strong></td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Cluster</strong></td>
<td>MPI</td>
<td>no</td>
<td>MPI</td>
<td>point 2 point</td>
<td>memcached</td>
</tr>
<tr>
<td><strong>State table</strong></td>
<td>dictionary split</td>
<td>separate sparse</td>
<td>separate</td>
<td>separate</td>
<td>joint sparse</td>
</tr>
<tr>
<td><strong>Schedule</strong></td>
<td>synchronous exact</td>
<td>synchronous exact</td>
<td>synchronous exact</td>
<td>asynchronous approximate messages</td>
<td>asynchronous exact</td>
</tr>
</tbody>
</table>

[S11]
Scaling Up Graphical Models: Conclusions

• Extremely high parallelism is achievable, but variance is high
  – Strongly data dependent
• Network and synchronization costs can be explicitly accounted for in algorithms
• Approximations are essential to removing barriers
• Multi-level parallelism allows maximizing utilization
• Multiple caches allow super-linear speedups
References


