Scalable Gaussian process models on matrices and tensors

Alan Qi
CS & Statistics
Purdue University

Joint work with F. Yan, Z. Xu, S. Zhe, and IBM Research
Models for graph and multiway data

1. STGP: predict who may be your friends on facebook

2. DinTucker: distributed online inference for large graphs and tensors
Outline

- Nonpara. Bayesian: Gaussian processes
- Utilizing Kronecker product
- Tensor data

STGP

DinTucker
Tensor data: multiple aspects

\[ Y_{i,j,k} \]: the value of the k-th biomarker (i.e., cell population) for the j-th patient after taking the i-th medicine

Predict drug response
Matrix data: networks

\[ Y_{i,j} : \begin{cases} 1 & \text{if node } i \text{ is linked to node } j, \\ 0 & \text{otherwise.} \end{cases} \]

Discover communities and predict unknown interactions.
Goals

- Predict unknown elements (e.g., drug response and network interactions)

- Identify latent multi-aspect groups (communities)
Classical Tucker decomposition

Generalization of matrix factorization

3D case:

\[ Y = \mathcal{G} \times_1 U \times_2 Z \times_3 V \]

\[ y_{ijk} = \sum_{r,s,t} g_{rst} u_{ir} z_{js} v_{ks} \]
Assumptions

- Complete
- Continuous
- Multi-linear
Solution

Gaussian process models for tensors
Gaussian processes: prior on scalar $f$

- Powerful Bayesian modeling tool
- Numerous applications: computer vision, sensor networks, computational biology, etc.

$$f \sim \mathcal{N}(0; K_x)$$
GP models on tensor $F$

- GP on a tensor: stochastic process in an infinite tensor space
- Projection of GP on any tensor of finite size follows a tensor-variate Gaussian distribution

$$
\mathcal{N}(F|0; K_u, K_v, K_z) = (2\pi)^{-\frac{3N}{2}} \prod_{s=u,v,z} |K_s|^{-\frac{N^2}{2}} 
\cdot \exp\left\{-\frac{1}{2} \| (F \times_1 K_u^{-1} \times_2 K_v^{-1} \times_3 K_z^{-1}) \circ F \|_2^2 \right\}
$$
Latent sparse GP on tensors

Element \((i, j, k)\) is characterized by

- **\(u_i\):** Sparse loading vector in latent medicine groups
- **\(z_j\):** Sparse loading vector in latent patient groups
- **\(v_k\):** Sparse loading vector in latent biomarker groups
Separate covariance for each dimension

\[ K_u(i, r) = k(u_i, u_r) \]

- Separate covariance/kernel function for each dimension

- The more similar loading vectors, the larger the covariance function value

Nonlinear relationship between medicines \( i \) and \( r \)
Predict unknown tensor elements

1) Based on observed data, estimate loading vectors:

\[
[u_i, z_j, v_k]
\]

2) Compute weights (similarities) between unknown and observed elements:

\[
w(ijk, rst) \equiv w([u_i, z_j, v_k], [u_r, z_s, v_t])
\]

3) Predict the unknown element:

\[
y_{i,j,k} = \sum_{r,s,t} w(ijk, rst) y_{r,s,t}
\]
**Graphical model**

\[ U = [u_1, \ldots, u_N] \]

Sparse loading vectors

\[ \mathbf{F} \sim \mathcal{N}(0; K_u, K_v, K_z) \]

Latent tensor

\[ \mathbf{Y}_O \sim \prod_{(i,j,k) \in O} p(y_{ijk} | f_{ijk}) \]

Observed data

\[ p(y_{ijk} | f_{ijk}) : \text{ Gaussian for continuous data, Probit for binary data, Possion for count data} \]

Unknown data

\[ \mathbf{Y}_U \]

\[ \mathbf{V} \]

\[ \mathbf{Z} \]
Benefits

• Handle binary and missing data
• Discover block/group structures
• Avoid overfitting
• Model prediction uncertainty
• Incorporate additional side information

Yan, Xu & Qi, 2011; Xu, Yan & Qi 2011
Two views of the same model

- Infinite latent factor model / nonlinear stochastic blockmodels
- Nonparametric prior on infinite exchangeable (high-dimensional) arrays
Infinite nonlinear tensor decomposition

Nonlinear feature mapping: infinite dimension

\[ f_{ijk} = G \times_1 \Phi(u_i) \times_2 \Phi(v_j) \times_3 \Phi(z_k) \]
\[ G \sim \mathcal{N}(0, I, I, I) \]
\[ y_{ijk} = f_{ijk} + e \]

Integrating out \( G \) gives GP on tensor \( F \)

\[ p(F|U, V, Z) = \int p(F|G, U, V, Z)p(G)dG = \mathcal{N}(0, K_u, K_v, K_z) \]

where \( k(u_i, u_j) = \Phi(u_i)^T \Phi(u_j) \)
Exchangeability and De Finetti theorem

• An exchangeable sequence: its joint distribution is invariant under arbitrary permutation of the indices.

• De Finetti (1931) and Hewitt and Savage (1955) showed:

  If \( X_1, \ldots, X_n, \ldots \) is exchangeable. Then there exists a probability measure \( \mu \) on probability measure, such that

  \[
  p(X_1 \in A_1, \ldots, X_n \in A_n) = \int Q(A_1) \cdots Q(A_n) \mu(dQ)
  \]

• How about priors on exchangeable arrays representing graphs?
Prior for random graphs

- Aldous (1981); Diaconis & Freedman (1981) (See Lauritzen 2007) showed: Any column-row exchangeable binary matrix can be generated as follows:

\[
f \sim p
\]

\[
U_i, V_j \mid W \sim_{\text{IID}} \text{Uniform}(0, 1)
\]

\[
G_{ij} \mid W, U_i, V_j \sim \text{Bernoulli}(f(U_i, V_j))
\]

- Transform \(U_i\) and \(V_j\) to other r.v.s via suitable transformations.
Prior for random graphs

- For weakly exchangeable arrays (shared permutations over rows and columns; for undirected graphs), our model (Yan et al. 2011) is

\[ f \sim \text{GP} \]
\[ U_i, V_j | W \sim_{\text{IID}} \text{Uniform}(0, 1) \]
\[ G_{ij} | W, U_i, V_j \sim \text{Bernoulli}(f(U_i, U_j)) \]

- Replace Bernoulli by Gaussian for continuous values

- Special case of Random function priors (Lloyd et al., 2012)
Related works

- Latent Eigenmodel: (Hoff 2007): parametric models on exchangeable arrays
- GP-LVM (Lawrence 2005): modeling only interactions in rows or columns
- Random function priors (Lloyd et al., 2012)
Algorithm: Variational EM

**Marginal likelihood**

\[
\log p(Y_O|U, V, Z) + \log p(U, V, Z)
\]

Variational approximation
Variational-EM

- **E step:**
  
  Maximize the variational lower bound over $q(F)$ & $q(S)$:
  
  $$\log p(Y_O|U, V, Z) \geq \int q(F)q(S) \log p(Y_O, F, S|U, V, Z)dF + H_q(F) + H_q(S)$$
  
  $$\equiv L(q(F'), q(S))$$
  
  where $S$ are parameters or auxiliary variables of noise model $p(y_{ijk}|f_{ijk})$ and $H_q(F) = -\int q(F) \log q(F)dF$.

- **M step:**
  
  Maximize the variational lower bound over $U, V, and Z$. 
Algorithm: explore model structures

Example: \( \text{Trace}\{(I + K_u \otimes K_v \otimes K_z)^{-1}\} \)

Direct computation: Matrix inversion

\( O(N^9) \)

Kronecker product operation:

\[
A \otimes B = \begin{bmatrix}
    a_{11}B & \cdots & a_{1n}B \\
    \vdots & \ddots & \vdots \\
    a_{m1}B & \cdots & a_{mn}B
\end{bmatrix}.
\]
Properties of Kronecker product

Properties: 
\[(AB) \otimes (CD) = (A \otimes B)(C \otimes D)\]
\[(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}\]

Theorem: 
Eigen-decomposition \(K = W \Lambda W^T\)

If \(K = K_u \otimes K_v \otimes K_z\) then

\[W = W_u \otimes W_v \otimes W_z\]
\[\Lambda = \Lambda_u \otimes \Lambda_v \otimes \Lambda_z\]

\(W_u\) : eigenvectors of \(K_u\)
\(\text{diag}\{\Lambda_u\}\) : eigenvalues of \(K_u\)
Reduced computational complexity

Example: \[ \text{Trace}\{(I + K_u \otimes K_v \otimes K_z)^{-1}\} \]

Direct computation

Using this theorem and trace properties

\[ O(N^9) \]

\[
\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \frac{1}{1 + \lambda_i^u \lambda_j^v \lambda_k^z}
\]

\[ O(N^3) \]

\[
= \sum_{i=1}^{M} \sum_{j=1}^{M} \sum_{k=1}^{M} \frac{1}{1 + \lambda_i^u \lambda_j^v \lambda_k^z}
\]

\[ O(N^2 M) \]

\[ M << N \]
2D case: GP stochastic blockmodels

- Undirected networks (friend relationships and protein-protein interactions)
- Represented by symmetric adjacent matrices

Yan, Xu & Qi, 2011; Xu, Yan & Qi 2011
2D: Coauthor networks

Co-authorship dataset: co-authorship links from 100 authors who have the largest number of co-authors from NIPS 1-17.
3D: Enron emails


- 3D tensor representation: Sender-Recipient-Subject

![Graph showing Area Under Curve (AUC) values for different factor numbers and methods, with 'Ours' highlighted in red and compared to other methods like InfTucker, CP, TD, HOSVD, NCP, WCP, and PTD.]
4D: Digg

- Digg dataset: Social news from digg.com
- 4D tensor representation: user-news-keywords-category
### 4D: Digg latent groups

Keyword groups learned by STGP:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>iphone</td>
<td>Olympics</td>
<td>american</td>
<td>journey</td>
</tr>
<tr>
<td>2</td>
<td>apple</td>
<td>games</td>
<td>australian</td>
<td>dream</td>
</tr>
<tr>
<td>3</td>
<td>internet</td>
<td>china</td>
<td>rothschild</td>
<td>action</td>
</tr>
<tr>
<td>4</td>
<td>*seo</td>
<td>beijing</td>
<td>moscow</td>
<td>wisdom</td>
</tr>
<tr>
<td></td>
<td>photosynth</td>
<td>red</td>
<td>century</td>
<td>top</td>
</tr>
</tbody>
</table>

*seo: search engine optimization
Next steps

• Better models for sparse graphs/tensors?

• Distributed inference for big graphs/tensors
Outline

Hierarchical Bayesian

Distributed online inf.

NELL, fMRI,...

STGP

DinTucker
Big data (with big issues)

- High computational cost: $O(d^3)$ where $d$ is the size of the largest mode.
- Not enough memory to store the data in a single machine
- Data are physically stored in different locations
Challenge for GPs on arrays

- Coupled array elements: a huge covariance over all them.

- Obvious solutions, ADMM or stochastic gradient descent, do not apply due to the partition function.
Solution: Hierarchical Bayesian modeling

- Local training over subarrays
- Global information sharing via the parent nodes
Solution: Hierarchical Bayesian modeling

\[ U \]

Global memberships

\[ \tilde{U}_n \]

Local memberships

\[ M_{nt} \]

Latent tensors

\[ Y_{nt} \]

Subarrays

[Diagram showing the hierarchical model with nodes labeled as described in the text.]
Specifically, we first break the observed multidimensional array into subarrays—making distributed inference local. GP enables fast computation over subarrays and the models, and the latent variables for these GP models are assumed to be sampled from many, smaller GP models. This allows us to conduct online learning and distribute the computation over many computational units for a large dimension. However, it still needs to perform eigen-decomposition over the main memory of a single computer; this requirement is a major bottleneck of InfTucker.

A solution is to use local GP models for each subarray, which allows us to compute the distribution of the global latent factors. Given an observed array $Y$, we can use probit models for binary observations. We allow the subarrays from different computational units to couple these distributions, and we use stochastic gradient descent (SGD) to optimize the core tensor model interpretation. Given a subarray $Y_{nt}$, we can set $U_{nt}$ as the corresponding local GP. Then we sample the noisy $Y_{nt}$ from the corresponding local GP. We also assume that we do not need to explicitly compute the distribution for the global tensor $Y$.
Hierarchical Bayesian on MapReduce

\[ U \quad \tilde{U}_n \quad M_{nt} \quad Y_{nt} \quad t=1...T_n \quad n=1...N \]

- Global memberships
- Local memberships
- Latent tensors
- Subarrays

Global sharing: Reducer
Local VB inf.: Mapper

Figure 1. Hierarchical Bayesian on MapReduce
Subarray sampling strategies

- Uniform
- Weighted
- Grid
Three questions

• How does the new distributed online inference compare with the sequential inference?

• How does it scale in terms of number of CPUs?

• How does it compare with the state-of-the-art large-scale tensor decomposition method?
To examine the scalability and predictive performance of accuracy, all versions of \( D \) in Figure 5. We tuned the learning rate used the three strategies described in Section 0.5. Furthermore, \( D \) Out\), the start time and end time of the action. We information such as user id, target resource (i.e., file system in a large company. The log provides various ACC: Access logs from a source code version control frequent entities.

DinTucker: Scaling up Gaussian process models on multidimensional arrays with billions of elements

6.2. Scalability with regard to the number of machines in Figure 3. To generate triples (user, action, resource) for analysis. We compared \( D \) and \( IN \) with regard to the number of machines on the NELL dataset. Note that the running time


<table>
<thead>
<tr>
<th>Number of Factors</th>
<th>Data</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>Number of entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.5</td>
<td>0.7</td>
<td>0.8</td>
<td>0.9</td>
<td>1.0</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>0.7</td>
<td>0.8</td>
<td>0.9</td>
<td>1.0</td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>0.7</td>
<td>0.8</td>
<td>0.9</td>
<td>1.0</td>
</tr>
<tr>
<td>15</td>
<td>0.5</td>
<td>0.7</td>
<td>0.8</td>
<td>0.9</td>
<td>1.0</td>
</tr>
<tr>
<td>20</td>
<td>0.5</td>
<td>0.7</td>
<td>0.8</td>
<td>0.9</td>
<td>1.0</td>
</tr>
</tbody>
</table>

\( \text{GIN} \) and \( \text{IN} \) are similar to InfTucker on the NELL dataset. This dataset is downloaded from Carlson et al. (\( \text{ACP} \)). The start time and end time of the action. We used the original GigaTensor implementation of Tikachov et al. (\( \text{AP} \)). We used anoth-}

Same accuracy as sequential inference

DinTucker with a specific sampling strategy

(b) digg2

(c) enron
Nearly linear scalability
More accurate, less time

(a) NELL: running time

(b) NELL: prediction
More accurate, less time

(c) ACC: running time

(d) ACC: prediction
Conclusions
Scalable nonparametric Bayesian methods for graphs

STGP: predict who may be your friends on Facebook

DinTucker: distributed online inference for large graphs and tensors
Acknowledgment

- **Funding agencies:**
  - NSF: CAREER, Robust Intelligence, CDI, STC
  - NIH: R01
  - Microsoft
  - IBM
  - Showalter foundation
  - Indiana Clinical & Translational Science Institute
  - Purdue