Implementing Tensor Methods: Application to Community Detection

Anima Anandkumar

U.C. Irvine
Recap: Basic Tensor Decomposition Method

Toy Example in MATLAB

- Simulated Samples: Exchangeable Model

- Whiten The Samples
  - Second Order Moments
  - Matrix Decomposition

- Orthogonal Tensor Eigen Decomposition
  - Third Order Moments
  - Power Iteration
Simulated Samples: Exchangeable Model

Model Parameters

- **Hidden State:**
  \[ h \in \text{basis } \{e_1, \ldots, e_k\} \]
  \( k = 2 \)

- **Observed States:**
  \[ x_i \in \text{basis } \{e_1, \ldots, e_d\} \]
  \( d = 3 \)

- **Conditional Independency:**
  \( x_1 \perp \perp x_2 \perp \perp x_3 | h \)

- **Transition Matrix:** \( A \)

- **Exchangeability:**
  \( \mathbb{E}[x_i | h] = Ah, \ \forall i \in 1, 2, 3 \)
Simulated Samples: Exchangeable Model

Model Parameters

- **Hidden State:**
  \[ h \in \text{basis} \{e_1, \ldots, e_k\} \]
  \[ k = 2 \]

- **Observed States:**
  \[ x_i \in \text{basis} \{e_1, \ldots, e_d\} \]
  \[ d = 3 \]

- **Conditional Independency:**
  \[ x_1 \perp \perp x_2 \perp \perp x_3|h \]

- **Transition Matrix:** \( A \)

- **Exchangeability:**
  \[ \mathbb{E}[x_i|h] = Ah, \ \forall i \in 1, 2, 3 \]

---

**Generate Samples Snippet**

```matlab
for t = 1 : n
    % generate h for this sample
    h_category=(rand()>0.5) + 1;
    h(t,h_category)=1;
    transition_cum=cumsum(A_true(:,h_category));
    % generate x1 for this sample | h
    x_category=find(transition_cum>rand(),1);
    x1(t,x_category)=1;
    % generate x2 for this sample | h
    x_category=find(transition_cum >rand(),1);
    x2(t,x_category)=1;
    % generate x3 for this sample | h
    x_category=find(transition_cum >rand(),1);
    x3(t,x_category)=1;
end
```
**Whiten The Samples**

**Second Order Moments**
- $M_2 = \frac{1}{n} \sum_t x_1^t \otimes x_2^t$

**Whitening Matrix**
- $W = U_w L_w^{-0.5}$
- $[U_w, L_w] = k$-svd($M_2$)

**Whiten Data**
- $y_1^t = W^\top x_1^t$

**Orthogonal Basis**
- $V = W^\top A \rightarrow V^\top V = I$

**Whitening Snippet**

```matlab
fprintf('The second order moment M2:');
M2 = x1'*x2/n;
[Uw, Lw, Vw] = svd(M2);
fprintf('M2 singular values:'); Lw
W = Uw(:,1:k)* sqrt(pinv(Lw(1:k,1:k)));
y1 = x1 * W; y2 = x2 * W; y3 = x3 * W;
```
Orthogonal Tensor Eigen Decomposition

Third Order Moments

\[ T = \frac{1}{n} \sum_{t \in [n]} y_1^t \otimes y_2^t \otimes y_3^t \approx \sum_{i \in [k]} \lambda_i v_i \otimes v_i \otimes v_i, \quad V^\top V = I \]

Gradient Ascent

\[ T(I, v_1, v_1) = \frac{1}{n} \sum_{t \in [n]} \langle v_1, y_2^t \rangle \langle v_1, y_3^t \rangle y_1^t \approx \sum_{i} \lambda_i \langle v_i, v_1 \rangle^2 v_i = \lambda_1 v_1. \]

• \( v_i \) are eigenvectors of tensor \( T \).
Orthogonal Tensor Eigen Decomposition

\[ T \leftarrow T - \sum_j \lambda_j v_j \otimes^3, \quad v \leftarrow \frac{T(I, v, v)}{\|T(I, v, v)\|} \]

Power Iteration Snippet

```matlab
V = zeros(k,k); Lambda = zeros(k,1);
for i = 1:k
    v_old = rand(k,1); v_old = normc(v_old);
    for iter = 1 : Maxiter
        v_new = (y1' * ((y2*v_old).*y3*v_old))/n;
        if i > 1
            % deflation
            for j = 1: i-1
                v_new = v_new - (V(:,j)*(v_old'*V(:,j))^2)* Lambda(j);
            end
        lambda = norm(v_new);v_new = normc(v_new);
        if norm(v_old - v_new) < TOL
            fprintf('Converged at iteration %d.', iter);
            V(:,i) = v_new; Lambda(i,1) = lambda;
            break;
        end
        v_old = v_new;
    end
end
```
Orthogonal Tensor Eigen Decomposition

\[ T \leftarrow T - \sum_j \lambda_j v^3_j, \quad v \leftarrow \frac{T(I, v, v)}{\|T(I, v, v)\|} \]

Power Iteration Snippet

```matlab
V = zeros(k, k); Lambda = zeros(k, 1);
for i = 1:k
    v_old = rand(k, 1); v_old = normc(v_old);
    for iter = 1 : Maxiter
        v_new = (y1' * ((y2 * v_old) .* (y3 * v_old))) / n;
        if i > 1
            % deflation
            for j = 1: i-1
                v_new = v_new - (V(:, j) * (v_old' * V(:, j))^2) * Lambda(j);
            end
        end
        lambda = norm(v_new); v_new = normc(v_new);
        if norm(v_old - v_new) < TOL
            fprintf('Converged at iteration %d.
', iter);
            V(:, i) = v_new; Lambda(i, 1) = lambda;
            break;
        end
    v_old = v_new;
end
```

Green: Groundtruth
Red: Estimation at each iteration
Applications and Challenges

Social Networks
- **Observed**: network of social ties,
- **Hidden**: groups/communities of actors.

Challenges
- Large Scale Networks: $n \sim$ millions or billions
- Latent Communities: $k \sim$ thousands

Topic modeling
- **Observed**: words in corpus,
- **Hidden**: topics.

Challenges
- Large Vocabulary: Words $d \sim 100,000$
- Huge Corpus: Documents $n \sim$ millions
- Latent Topics: $k \sim$ thousands
Resources for this talk

Papers


Code

- GPU and CPU codes: github.com/FurongHuang/Fast-Detection-of-Overlapping-Communities-via-Online-Tensor
- REEF code will be released soon.
Outline

1 Recap: A Toy Example via MATLAB

2 Community Detection Implementation
   - Subgraph Counts as Graph Moments
   - Whitening
   - Tensor Decomposition
   - Code Optimization
   - Experimental Results

3 Implementing In the Cloud

4 Conclusion
Outline

1 Recap: A Toy Example via MATLAB

2 Community Detection Implementation
   • Subgraph Counts as Graph Moments
     • Whitening
     • Tensor Decomposition
     • Code Optimization
     • Experimental Results

3 Implementing In the Cloud

4 Conclusion
Mixed Membership Community Models

Stochastic Block Model

Mixed Membership Model

Goal: Recover communities $\Pi$ given adjacency matrix $G$

$\alpha_0 = 0$

$\alpha_0 = 1$
Subgraph Counts as Graph Moments

3-star counts sufficient for identifiability and learning of MMSB

3-Star Count Tensor

\[
\tilde{M}_3(a, b, c) = \frac{1}{|X|} \# \text{ of common neighbors in } X \\
= \frac{1}{|X|} \sum_{x \in X} G(x, a) G(x, b) G(x, c).
\]

\[
\tilde{M}_3 = \frac{1}{|X|} \sum_{x \in X} \left[ G_{x,A}^\top \otimes G_{x,B}^\top \otimes G_{x,C}^\top \right]
\]
Multi-view Representation

- Conditional independence of the three views
- $\pi_x$: community membership vector of node $x$.

**3-stars**

**Graphical model**

**Linear Multiview Model:**

$$\mathbb{E}[G_{x,A}^\top | \Pi] = \Pi_A^\top P^\top \pi_x = U \pi_x.$$
Subgraph Counts as Graph Moments

Second and Third Order Moments

\[ \hat{M}_2 := \frac{1}{|X|} \sum_x Z_C G_{x,c}^\top G_{x,B} Z_B^\top - \text{shift} \]

\[ \hat{M}_3 := \frac{1}{|X|} \sum_x \left[ G_{x,a}^\top \otimes Z_B G_{x,B}^\top \otimes Z_C G_{x,C}^\top \right] - \text{shift} \]

Symmetrize Transition Matrices

- \( \text{Pairs}_{C,B} := G_{X,C}^\top \otimes G_{X,B}^\top \)
- \( Z_B := \text{Pairs} (A, C) (\text{Pairs} (B, C))^\dagger \)
- \( Z_C := \text{Pairs} (A, B) (\text{Pairs} (C, B))^\dagger \)

Linear Multiview Model: \( \mathbb{E}[G_{x,a}^\top | \Pi] = U \pi_x. \)

\[ \mathbb{E}[\hat{M}_2 | \Pi_{A,B,C}] = \sum_i \frac{\alpha_i}{\alpha_0} u_i \otimes u_i, \quad \mathbb{E}[\hat{M}_3 | \Pi_{A,B,C}] = \sum_i \frac{\alpha_i}{\alpha_0} u_i \otimes u_i \otimes u_i. \]
Overview of Tensor Method

- Whiten data via SVD of $\hat{M}_2 \in \mathbb{R}^{n \times n}$.
- Estimate the third moment $\hat{M}_3 \in \mathbb{R}^{n \times n \times n}$ and whiten it implicitly to obtain $T$.
- Run power method (gradient ascent) on $T$.
- Apply post-processing to obtain communities.
- Compute error scores and validate with ground truth (if available).
Outline

1 Recap: A Toy Example via MATLAB

2 Community Detection Implementation
   - Subgraph Counts as Graph Moments
   - Whitening
   - Tensor Decomposition
   - Code Optimization
   - Experimental Results

3 Implementing In the Cloud

4 Conclusion
Whitening Matrix Computation

Symmetrization: Finding Second Order Moments $M_2$

\[ \hat{M}_2 = \begin{bmatrix} Z_C \end{bmatrix}_{\text{Pairs}_{C,B}} \begin{bmatrix} Z_B^\top \end{bmatrix}_{\text{Pairs}_{C,B}} - \text{shift} \]

\[ = \begin{bmatrix} (\text{Pairs}_{A,B} \text{Pairs}_{C,B}^\top) \end{bmatrix}_{\text{Pairs}_{C,B}} \begin{bmatrix} (\text{Pairs}_{B,C}^\top) \end{bmatrix}_{\text{Pairs}_{A,C}} \begin{bmatrix} \text{Pairs}_{A,C}^\top \end{bmatrix} - \text{shift} \]

Challenges: $n \times n$ objects, $n \sim$ millions or billions
**Whitening Matrix Computation**

**Symmetrization: Finding Second Order Moments** $M_2$

$$\hat{M}_2 = \begin{bmatrix} Z_C \end{bmatrix}_{Pairs_{C,B}} \begin{bmatrix} Z_B^T \end{bmatrix} - \text{shift}$$

$$= \begin{bmatrix} \left( Pairs_{A,B} Pairs_{C,B}^\dagger \right) \end{bmatrix}_{Pairs_{C,B}} \begin{bmatrix} \left( Pairs_{B,C}^\dagger \right)^T \end{bmatrix}_{Pairs_{A,C}} - \text{shift}$$

**Challenges:** $n \times n$ objects, $n \sim$ millions or billions

**Order Manipulation:** Low Rank Approx. is the key, avoid $n \times n$ objects
Whitening Matrix Computation

Symmetrization: Finding Second Order Moments $M_2$

$$\hat{M}_2 = \begin{bmatrix} Z_C \end{bmatrix}_{Pairs_{C,B}} \begin{bmatrix} Z_B^T \end{bmatrix} - \text{shift}$$

$$= \left[ (Pairs_{A,B} Pairs_{C,B}^\dagger) \right]_{Pairs_{C,B}} \begin{bmatrix} (Pairs_{B,C}^\dagger)^T \end{bmatrix}_{Pairs_{A,C}} - \text{shift}$$

Challenges: $n \times n$ objects, $n \sim$ millions or billions

Order Manipulation: Low Rank Approx. is the key, avoid $n \times n$ objects

$$|A| = \begin{pmatrix} |A| & \dagger \end{pmatrix} \begin{pmatrix} \dagger & T \end{pmatrix}$$

$n=1M, k=5K$: Size($\text{Matrix } n \times n$) = 58TB vs Size($\text{Matrix } n \times k$) = 3.7GB.

Space Complexity $O(nk)$
Whitening Matrix Computation

Symmetrization: Finding Second Order Moments $M_2$

$$\hat{M}_2 = \begin{bmatrix} Z_C \end{bmatrix} \text{Pairs}_{C,B} \begin{bmatrix} Z_B^T \end{bmatrix} - \text{shift}$$

$$= \begin{bmatrix} \text{Pairs}_{A,B} & \text{Pairs}_{C,B}^\dagger \end{bmatrix} \text{Pairs}_{C,B} \begin{bmatrix} \text{Pairs}_{B,C}^\dagger \end{bmatrix}^T \text{Pairs}_{A,C}^T - \text{shift}$$

Challenges: $n \times n$ objects, $n \sim$ millions or billions

Order Manipulation: Low Rank Approx. is the key, avoid $n \times n$ objects

$n=1M, k=5K$: Size(Matrix $n \times n$) = 58TB vs Size(Matrix $n \times k$) = 3.7GB.

Space Complexity $O(nk)$
Whitening Matrix Computation

Symmetrization: Finding Second Order Moments $M_2$

\[
\hat{M}_2 = Z_C \text{Pairs}_{C,B} Z_B^\top - \text{shift}
\]

\[
= \left( \text{Pairs}_{A,B} \text{Pairs}_{C,B}^\dagger \right) \text{Pairs}_{C,B} \left( \text{Pairs}_{B,C}^\dagger \right)^\top \text{Pairs}_{A,C}^\top - \text{shift}
\]

Challenges: $n \times n$ objects, $n \sim$ millions or billions

Order Manipulation: Low Rank Approx. is the key, avoid $n \times n$ objects

\[
\begin{align*}
\text{Space Complexity} \quad O(nk)
\end{align*}
\]

$n=1M, k=5K$: Size(Matrix $n \times n$) = 58TB vs Size(Matrix $n \times k$) = 3.7GB.
Whitening Matrix Computation

Orthogonalization: Finding Whitening Matrix $W$

$W^T M_2 W = I$ is solved by $\text{k-svd}(M_2)$

Challenges: $n \times n$ Matrix SVDs, $n \sim$ millions or billions
Whitening Matrix Computation

Orthogonalization: Finding Whitening Matrix $W$

$W^T M_2 W = I$ is solved by $\text{k-svd}(M_2)$

Challenges: $n \times n$ Matrix SVDs, $n \sim$ millions or billions

Randomized low rank approx. (GM 13’, CW 13’)

- Random matrix $S \in \mathbb{R}^{n \times \tilde{k}}$ for dense $M_2$
- Column selection matrix: random signs $S \in \{0, 1\}^{n \times \tilde{k}}$ for sparse $M_2$.
- $Q = \text{orth}(M_2 S)$, $Z = (M_2 Q)^\top M_2 Q$
- $[U_z, L_z, V_z] = \text{SVD}(Z)$ \hspace{1cm} $\%$ $Z \in \mathbb{R}^{k \times k}$
- $V_{M_2} = M_2 Q V_z L_z^{-\frac{1}{2}}$, $L_{M_2} = L_z^{\frac{1}{2}}$
Whitening Matrix Computation

Orthogonalization: Finding Whitening Matrix $W$

$W^T M_2 W = I$ is solved by $\text{k-svd}(M_2)$

Challenges: $n \times n$ Matrix SVDs, $n \sim$ millions or billions

Randomized low rank approx. (GM 13’, CW 13’)

- Random matrix $S \in \mathbb{R}^{n \times \tilde{k}}$ for dense $M_2$
- Column selection matrix: random signs $S \in \{0, 1\}^{n \times \tilde{k}}$ for sparse $M_2$
- $Q = \text{orth}(M_2 S)$, $Z = (M_2 Q)^\top M_2 Q$
- $[U_z, L_z, V_z] = \text{SVD}(Z) \% Z \in \mathbb{R}^{k \times k}$
- $V_{M_2} = M_2 Q V_z L_z^{-\frac{1}{2}}$, $L_{M_2} = L_z^{\frac{1}{2}}$

Computational Complexity

- For exact rank-$k$ SVD of $n \times n$ matrix: $O(n^2 k)$.
- For randomized SVD with $c$ cores and sparsity level $s$ per row of $M_2$: $O(nsk/c + k^3)$
Outline

1. Recap: A Toy Example via MATLAB

2. Community Detection Implementation
   - Subgraph Counts as Graph Moments
   - Whitening
   - Tensor Decomposition
   - Code Optimization
   - Experimental Results

3. Implementing In the Cloud

4. Conclusion
Multi-linear transform

- $M_3 \in \mathbb{R}^{n \times n \times n}$ and $T \in \mathbb{R}^{k \times k \times k}$.

- $T = M_3(W, W, W) = \sum_i w_i (W^\top a_i) \otimes^3$.

- $T = \sum_{i \in [k]} w_i \cdot v_i \otimes v_i \otimes v_i$ is orthogonal.

- Dimensionality reduction when $k \ll n$. 

Batch Gradient Descent

Power Iteration with Deflation

\[ T \leftarrow T - \sum_j \lambda_j v_j \otimes^3, \quad v_i \leftarrow \frac{T(I, v_i, v_i)}{\|T(I, v_i, v_i)\|}, j < i \]

Alternating Least Squares

\[
\min_{\sigma, A, B, C} \left\| T - \sum_{i=1}^{k} \lambda_i A(:, i) \otimes B(:, i) \otimes C(:, i) \right\|_F^2
\]

such that \( A^\top A = I, \quad B^\top B = I \) and \( C^\top C = I \).

Challenges:

Requires forming the tensor/passing over data in each iteration
Stochastic (Implicit) Tensor Gradient Descent

Whitened third order moments:

\[ T = M_3(W, W, W). \]

Objective:

\[
\arg \min_v \left\{ \| \theta \sum_{i \in [k]} v_i \otimes^3 - \sum_{t \in X} T^t \|_F^2 \right\},
\]

where \( v_i \) are the unknown tensor eigenvectors, \( T^t = g_A^t \otimes g_B^t \otimes g_C^t \) — shift such that \( g_A^t = W^\top G_{\{x,A\}}, \ldots \)
Stochastic (Implicit) Tensor Gradient Descent

Whitened third order moments:

\[ T = M_3(W, W, W). \]

Objective:

\[
\arg \min_v \left\{ \| \theta \sum_{i \in [k]} v_i^{\otimes 3} - \sum_{t \in X} T_t^t \|_F^2 \right\},
\]

where \( v_i \) are the unknown tensor eigenvectors, \( T_t^t = g_A^t \otimes g_B^t \otimes g_C^t \)—shift such that \( g_A^t = W^\top G_{\{x, A\}}, \ldots \)

Expand the objective:

\[
\theta \left\| \sum_{i \in [k]} v_i^{\otimes 3} \right\|_F^2 - \left\langle \sum_{i \in [k]} v_i^{\otimes 3}, T_t^t \right\rangle
\]

Orthogonality cost vs Correlation Reward
Stochastic (Implicit) Tensor Gradient Descent

Updating Equation

\[ v_{i}^{t+1} \leftarrow v_{i}^{t} - 3\theta \beta^{t} \sum_{j=1}^{k} \left[ \langle v_{j}^{t}, v_{i}^{t} \rangle^{2} v_{j}^{t} \right] + \beta^{t} \langle v_{i}^{t}, g_{A}^{t} \rangle \langle v_{i}^{t}, g_{B}^{t} \rangle g_{C}^{t} + \ldots \]

Orthogonality cost vs Correlation Reward

Never form the tensor explicitly; multilinear operation on implicit tensor.

Space: \( O(k^2) \), Time: \( O(k^3/c) \times \) iterations with \( c \) cores.
Unwhitening

Post Processing for memberships

- $\Lambda$: eigenvalues. $\Phi$: eigenvectors.
- $G$: adjacency matrix, $\gamma$: normalization.
- $W$: Whitening Matrix.

$$\hat{\Pi}_{A^c} = \text{diag}(\gamma)^{1/3} \text{diag}(\Lambda)^{-1} \Phi^\top W^\top G_{A,A^c},$$

where $A^c := X \cup B \cup C$.

- Threshold the values.

Space Complexity $O(nk)$

Time Complexity $O(nsk/c)$ with $c$ cores.
**Computational Complexity** \((k \ll n)\)

- \(n = \# \text{ of nodes}\)
- \(N = \# \text{ of iterations}\)
- \(k = \# \text{ of communities}\)
- \(m = \# \text{ of sampled node pairs (variational)}\)

<table>
<thead>
<tr>
<th>Module</th>
<th>Pre</th>
<th>STGD</th>
<th>Post</th>
<th>Var</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>(O(nk))</td>
<td>(O(k^2))</td>
<td>(O(nk))</td>
<td>(O(nk))</td>
</tr>
<tr>
<td>Time</td>
<td>(O(nsk/c + k^3))</td>
<td>(O(Nk^3/c))</td>
<td>(O(nsk/c))</td>
<td>(O(mkN))</td>
</tr>
</tbody>
</table>

Variational method: \(O(m \times k)\) for each iteration

\(O(n \times k) < O(m \times k) < O(n^2 \times k)\)

Our approach: \(O(nsk/c + k^3)\)
Computational Complexity ($k \ll n$)

- $n =$ number of nodes
- $k =$ number of communities
- $N =$ number of iterations
- $m =$ number of sampled node pairs (variational)

<table>
<thead>
<tr>
<th>Module</th>
<th>Pre</th>
<th>STGD</th>
<th>Post</th>
<th>Var</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>$O(nk)$</td>
<td>$O(k^2)$</td>
<td>$O(nk)$</td>
<td>$O(nk)$</td>
</tr>
<tr>
<td>Time</td>
<td>$O(nsk/c + k^3)$</td>
<td>$O(Nk^3/c)$</td>
<td>$O(nsk/c)$</td>
<td>$O(mkN)$</td>
</tr>
</tbody>
</table>

Variational method: $O(m \times k)$ for each iteration

$O(n \times k) < O(m \times k) < O(n^2 \times k)$

Our approach: $O(nsk/c + k^3)$

In practice STGD is extremely fast and is not the bottleneck
Outline

1 Recap: A Toy Example via MATLAB

2 Community Detection Implementation
   - Subgraph Counts as Graph Moments
   - Whitening
   - Tensor Decomposition
   - Code Optimization
   - Experimental Results

3 Implementing In the Cloud

4 Conclusion
GPU/CPU Implementation

**GPU (SIMD)**
- **GPU**: Hundreds of cores; parallelism for matrix/vector operations
- **Speed-up**: Order of magnitude gains
- **Big data challenges**: GPU memory $\ll$ CPU memory $\ll$ Hard disk

![Storage hierarchy diagram]

![Partitioned matrix diagram]
GPU/CPU Implementation

**GPU (SIMD)**
- **GPU**: Hundreds of cores; parallelism for matrix/vector operations
- **Speed-up**: Order of magnitude gains
- **Big data challenges**: GPU memory ≪ CPU memory ≪ Hard disk

![Storage hierarchy]  
- Hard disk (expandable)  
- CPU memory (expandable)  
- GPU memory (not expandable)

**CPU**
- **CPU**: Sparse Representation, Expandable Memory
- Randomized Dimensionality Reduction
Scaling Of The Stochastic Iterations

\[ v_{i}^{t+1} \leftarrow v_{i}^{t} - 3\theta \beta^{t} \sum_{j=1}^{k} \left[ \langle v_{j}^{t}, v_{i}^{t} \rangle^2 v_{j}^{t} \right] + \beta^{t} \langle v_{i}^{t}, g_{A}^{t} \rangle \langle v_{i}^{t}, g_{B}^{t} \rangle g_{C}^{t} + \ldots \]

- Parallelize across eigenvectors.

- STGD is iterative: device code reuse buffers for updates.
Scaling Of The Stochastic Iterations

\[ v_{i}^{t+1} \leftarrow v_{i}^{t} - 3\theta\beta^{t} \sum_{j=1}^{k} \left[ \langle v_{j}^{t}, v_{i}^{t} \rangle^{2} v_{j}^{t} \right] + \beta^{t} \langle v_{i}^{t}, g_{A}^{t} \rangle \langle v_{i}^{t}, g_{B}^{t} \rangle g_{C}^{t} + \ldots \]

- Parallelize across eigenvectors.

- STGD is iterative: device code reuse buffers for updates.
Scaling Of The Stochastic Iterations

Running time (secs)

Number of communities $k$

- MATLAB Tensor Toolbox (CPU)
- CULA Standard Interface (GPU)
- CULA Device Interface (GPU)
- Eigen Sparse (CPU)
Validation Metrics

Ground-truth membership available

- Ground-truth membership matrix $\Pi$ vs Estimated membership $\hat{\Pi}$
Validation Metrics

Ground-truth membership available

- Ground-truth membership matrix $\Pi$ vs Estimated membership $\hat{\Pi}$

Problem: How to relate $\Pi$ and $\hat{\Pi}$?
Validation Metrics

Ground-truth membership available

- Ground-truth membership matrix $\Pi$ vs Estimated membership $\hat{\Pi}$

Problem: How to relate $\Pi$ and $\hat{\Pi}$?

Solution: $p$-value test based soft-“pairing”
Validation Metrics

Ground-truth membership available

- Ground-truth membership matrix $\Pi$ vs Estimated membership $\hat{\Pi}$

Problem: How to relate $\Pi$ and $\hat{\Pi}$?

Solution: $p$-value test based soft- “pairing”
Evaluation Metrics

- **Recovery Ratio**: % of ground-truth com recovered
- **Error Score**: $\mathcal{E} := \frac{1}{nk} \sum \{\text{paired membership errors}\}$

  $$= \frac{1}{k} \sum_{(i,j) \in E_{\{p_{val}\}}} \left\{ \frac{1}{n} \sum_{x \in |X|} |\hat{\Pi}_i(x) - \Pi_j(x)| \right\}$$

Insights

- $l_1$ norm error between $\hat{\Pi}_i$ and the corresponding paired $\Pi_j$
- False pairings penalization
  - too many falsely discovered pairings, error $>1$
Outline

1 Recap: A Toy Example via MATLAB

2 Community Detection Implementation
   - Subgraph Counts as Graph Moments
   - Whitening
   - Tensor Decomposition
   - Code Optimization
   - Experimental Results

3 Implementing In the Cloud

4 Conclusion
## Summary of Results

### Graphs
- **Users**: Friend
- **Business**: User, Reviews
- **Author**: Coauthor

### Datasets
- **Facebook**: $n \sim 20k$
- **Yelp**: $n \sim 40k$
- **DBLP(sub)**: $n \sim 1$ million ($\sim 100k$)

### Error ($\mathcal{E}$) and Recovery ratio ($\mathcal{R}$)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$\hat{k}$</th>
<th>Method</th>
<th>Running Time</th>
<th>$\mathcal{E}$</th>
<th>$\mathcal{R}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook (k=360)</td>
<td>500</td>
<td>ours</td>
<td>468</td>
<td>0.0175</td>
<td>100%</td>
</tr>
<tr>
<td>Facebook (k=360)</td>
<td>500</td>
<td>variational</td>
<td>86,808</td>
<td>0.0308</td>
<td>100%</td>
</tr>
<tr>
<td>Yelp (k=159)</td>
<td>100</td>
<td>ours</td>
<td>287</td>
<td>0.046</td>
<td>86%</td>
</tr>
<tr>
<td>Yelp (k=159)</td>
<td>100</td>
<td>variational</td>
<td>N.A.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBLP sub (k=250)</td>
<td>500</td>
<td>ours</td>
<td>10,157</td>
<td>0.139</td>
<td>89%</td>
</tr>
<tr>
<td>DBLP sub (k=250)</td>
<td>500</td>
<td>variational</td>
<td>558,723</td>
<td>16.38</td>
<td>99%</td>
</tr>
<tr>
<td>DBLP (k=6000)</td>
<td>100</td>
<td>ours</td>
<td>5407</td>
<td>0.105</td>
<td>95%</td>
</tr>
</tbody>
</table>

Thanks to Prem Gopalan and David Mimno for providing variational code.
Experimental Results on Yelp

Lowest error business categories & largest weight businesses

<table>
<thead>
<tr>
<th>Rank</th>
<th>Category</th>
<th>Business</th>
<th>Stars</th>
<th>Review Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Latin American</td>
<td>Salvadoreno Restaurant</td>
<td>4.0</td>
<td>36</td>
</tr>
<tr>
<td>2</td>
<td>Gluten Free</td>
<td>P.F. Chang’s China Bistro</td>
<td>3.5</td>
<td>55</td>
</tr>
<tr>
<td>3</td>
<td>Hobby Shops</td>
<td>Make Meaning</td>
<td>4.5</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>Mass Media</td>
<td>KJZZ 91.5FM</td>
<td>4.0</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>Yoga</td>
<td>Sutra Midtown</td>
<td>4.5</td>
<td>31</td>
</tr>
</tbody>
</table>
Experimental Results on Yelp

Lowest error business categories & largest weight businesses

<table>
<thead>
<tr>
<th>Rank</th>
<th>Category</th>
<th>Business</th>
<th>Stars</th>
<th>Review Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Latin American</td>
<td>Salvadoreno Restaurant</td>
<td>4.0</td>
<td>36</td>
</tr>
<tr>
<td>2</td>
<td>Gluten Free</td>
<td>P.F. Chang’s China Bistro</td>
<td>3.5</td>
<td>55</td>
</tr>
<tr>
<td>3</td>
<td>Hobby Shops</td>
<td>Make Meaning</td>
<td>4.5</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>Mass Media</td>
<td>KJZZ 91.5FM</td>
<td>4.0</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>Yoga</td>
<td>Sutra Midtown</td>
<td>4.5</td>
<td>31</td>
</tr>
</tbody>
</table>

Bridgeness: Distance from vector $[1/\hat{k}, \ldots, 1/\hat{k}]^\top$

Top-5 bridging nodes (businesses)

<table>
<thead>
<tr>
<th>Business</th>
<th>Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Four Peaks Brewing</td>
<td>Restaurants, Bars, American, Nightlife, Food, Pubs, Tempe</td>
</tr>
<tr>
<td>Pizzeria Bianco</td>
<td>Restaurants, Pizza, Phoenix</td>
</tr>
<tr>
<td>FEZ</td>
<td>Restaurants, Bars, American, Nightlife, Mediterranean, Lounges, Phoenix</td>
</tr>
<tr>
<td>Matt’s Big Breakfast</td>
<td>Restaurants, Phoenix, Breakfast&amp; Brunch</td>
</tr>
<tr>
<td>Cornish Pasty Co</td>
<td>Restaurants, Bars, Nightlife, Pubs, Tempe</td>
</tr>
</tbody>
</table>
Outline

1 Recap: A Toy Example via MATLAB

2 Community Detection Implementation
   - Subgraph Counts as Graph Moments
   - Whitening
   - Tensor Decomposition
   - Code Optimization
   - Experimental Results

3 Implementing In the Cloud

4 Conclusion
Review of linear algebra

Tensor Modes

- Analogy to Matrix Rows and Matrix Columns.
- For an order-d tensor $A \in \mathbb{R}^{n_1 \times n_2 \cdots n_d}$:
  - mode-1 has dimension $n_1$,
  - mode-2 has dimension $n_2$, and so on.

Tensor Unfolding

In a mode-k unfolding, the mode-k fibers are assembled to produce an $n_k$-by-$N/n_k$ matrix where $N = n_1 \cdots n_d$.

[Diagram of tensor unfolding]

- Mode-1 Unfolding of $A \in \mathbb{R}^{2\times2\times2}$

$$
\begin{bmatrix}
    a_{111} & a_{121} & a_{112} & a_{122} \\
    a_{211} & a_{221} & a_{212} & a_{222}
\end{bmatrix}
$$
Tensor Decomposition In The Cloud

Tensor decomposition is equivalent to

$$\min_{\sigma, A, B, C} \left\| T - \sum_{i=1}^{k} \sigma_i A(:, i) \otimes B(:, i) \otimes C(:, i) \right\|_F^2$$
Tensor Decomposition In The Cloud

- Tensor decomposition is equivalent to

$$\min_{\sigma, A, B, C} \left\| T - \sum_{i=1}^{k} \sigma_i A(:, i) \otimes B(:, i) \otimes C(:, i) \right\|_F^2$$

- Alternating Least Square is the solution:

$$A' \leftarrow T_a f(C, B) \left( C^\top C \ast B^\top B \right)^\dagger$$

$$B' \leftarrow T_b f(C, A') \left( C^\top C \ast A'^\top A' \right)^\dagger$$

$$C' \leftarrow T_c f(B', A') \left( B'^\top B' \ast A'^\top A' \right)^\dagger$$

where $T_a$ is the mode-1 unfolding of $T$, $T_b$ is the mode-2 unfolding of $T$, and $T_c$ is the mode-3 unfolding of $T$.

---

Low Rank Structure: Hidden Dimension $<$ Observable Dimension
Challenges I

How to parallelize?

- Observations: \( A'(i,: \leftarrow T_a(i,:) f(C, B)(C^\top C \ast B^\top B)^\dagger \)
- \( T_a \in \mathbb{R}^{k \times k^2}, \quad B \text{ and } C \in \mathbb{R}^{k \times k} \)
Challenges I

How to parallelize?

- Observations: \( A'(i,:) \leftarrow T_a(i,:) f(C, B) (C^\top C \ast B^\top B)\dagger \)
- \( T_a \in \mathbb{R}^{k \times k^2}, B \text{ and } C \in \mathbb{R}^{k \times k} \)

Update Rows Independently

\( k \) tensor slices \( \in \mathbb{R}^{k^2} \)
Challenges II

Communication and System Architecture Overhead

- Map-Reduce Framework

Overhead: Disk reading, Container Allocation, Intense Key/Value Design
Challenges II

Solution: REEF

- Big data framework called REEF (Retainable Evaluator Execution Framework)

- Advantage: Open source distributed system with one time container allocation, keep the tensor in memory
**Correctness**

**Evaluation Score**

\[
\text{perplexity} := \exp \left( - \frac{\sum_i \text{log-likelihood in doc } i}{\sum_i \text{words in doc } i} \right)
\]

**New York Times Corpus**

- Documents \( n = 300,000 \)
- Vocabulary \( d = 100,000 \)
- Topics \( k = 100 \)

<table>
<thead>
<tr>
<th></th>
<th>Stochastic Variational Inference</th>
<th>Tensor Decomposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perplexity</td>
<td>4000</td>
<td>3400</td>
</tr>
</tbody>
</table>

**SVI drawbacks:**

- Hyper parameters
- Learning rate
- Initial points
# Running Time

## Computational Complexity

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Whitening</th>
<th>Tensor Slices $(1, \ldots, k)$</th>
<th>ALS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time</strong></td>
<td>$O(k^3)$</td>
<td>$O(k^2)$ per slice</td>
<td>$O(k^3)$</td>
</tr>
<tr>
<td><strong>Space</strong></td>
<td>$O(kd)$</td>
<td>$O(k^2)$ per slice</td>
<td>$O(k^2)$</td>
</tr>
<tr>
<td><strong>Degree of Parallelism</strong></td>
<td>$\infty$</td>
<td>$\infty$ per slice</td>
<td>$k$</td>
</tr>
<tr>
<td><strong>Communication</strong></td>
<td>$O(kd)$</td>
<td>$O(k^2)$</td>
<td>$O(k^2)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>SVI</th>
<th>1 node Map Red</th>
<th>1 node REEF</th>
<th>4 node REEF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>overall</strong></td>
<td>2 hours</td>
<td>4 hours 31 mins</td>
<td>68 mins</td>
<td>36 mins</td>
</tr>
<tr>
<td><strong>Whiten</strong></td>
<td>16 mins</td>
<td>16 mins</td>
<td>16 mins</td>
<td>16 mins</td>
</tr>
<tr>
<td><strong>Matricize</strong></td>
<td>15 mins</td>
<td>15 mins</td>
<td>4 mins</td>
<td></td>
</tr>
<tr>
<td><strong>ALS</strong></td>
<td>4 hours</td>
<td>37 mins</td>
<td>16 mins</td>
<td></td>
</tr>
</tbody>
</table>
Outline

1 Recap: A Toy Example via MATLAB

2 Community Detection Implementation
   • Subgraph Counts as Graph Moments
   • Whitening
   • Tensor Decomposition
   • Code Optimization
   • Experimental Results

3 Implementing In the Cloud

4 Conclusion
Conclusion

Guaranteed Learning of Latent Variable Models

- Guaranteed to recover correct model
- Efficient sample and computational complexities
- Better performance compared to EM, Variational Bayes etc.

- Tensor approach: mixed membership communities, topic models, latent trees...

In practice

- Scalable and embarrassingly parallel: handle large datasets.
- Efficient performance: perplexity or ground truth validation.

Theoretical guarantees and promising practical performance