Tensor Factorization via Matrix Factorization

Volodymyr Kuleshov*
Arun Tejasvi Chaganty*
Percy Liang

Stanford University

May 8, 2015
What is tensor (CP) factorization?  
(Kolda and Bader 2009)

- Tensor analogue of matrix eigen-decomposition.

\[ M = \sum_{i=1}^{k} \pi_i u_i \otimes u_i. \]

- Goal: Given \( T \) with noise, \( \epsilon \in \mathbb{R} \), recover factors \( u_i \).

\[
\begin{array}{cccc}
\square & = & \text{Tensor} & + \text{Tensor} & + \cdots & + \text{Tensor} \\
\end{array}
\]

\[ k \]
What is tensor (CP) factorization? (Kolda and Bader 2009)

- Tensor analogue of matrix eigen-decomposition.

\[
T = \sum_{i=1}^{k} \pi_i u_i \otimes u_i \otimes u_i .
\]

Kuleshov, Chaganty, Liang (Stanford University) Tensor Factorization May 8, 2015 2 / 27
What is tensor (CP) factorization? (Kolda and Bader 2009)

- Tensor analogue of matrix eigen-decomposition.

\[
\hat{T} = \sum_{i=1}^{k} \pi_i u_i \otimes u_i \otimes u_i + \epsilon R.
\]

- **Goal:** Given \( T \) with noise, \( \epsilon R \), **recover factors** \( u_i \).

\[
\begin{align*}
\text{Cube} & = \text{Tensor} + \text{Tensor} + \cdots + \text{Tensor} + \text{Noise} \\
& = \sum_{i=1}^{k} \hat{T}_i
\end{align*}
\]
What is tensor (CP) factorization?  
(Kolda and Bader 2009)

- Tensor analogue of matrix eigen-decomposition. 

\[
\hat{T} = \sum_{i=1}^{k} \pi_i u_i \otimes u_i \otimes u_i + \epsilon R.
\]

- **Goal:** Given $T$ with noise, $\epsilon R$, **recover factors** $u_i$. 

![Orthogonal and Non-orthogonal Tensor Factorization](image-url)
Why tensor factorization?

- To solve multi-linear algebra problems.
Why tensor factorization?

- To solve multi-linear algebra problems.
- Parsing
  - Cohen, Satta, and Collins 2013
Why tensor factorization?

- To solve multi-linear algebra problems.
- **Parsing**
  - Cohen, Satta, and Collins 2013
- **Knowledge base completion**
  - Chang et al. 2014
  - Singh, Rocktäschel, and Riedel 2015

TODO:
- crowdsourcing
- others
Why tensor factorization?

- To solve multi-linear algebra problems.
- Parsing
  - Cohen, Satta, and Collins 2013
- Knowledge base completion
  - Chang et al. 2014
  - Singh, Rocktäschel, and Riedel 2015
- Topic modelling
  - Anandkumar et al. 2012
Why tensor factorization?

- To solve multi-linear algebra problems.
- **Parsing**
  - Cohen, Satta, and Collins 2013
- **Knowledge base completion**
  - Chang et al. 2014
  - Singh, Rocktäschel, and Riedel 2015
- **Topic modelling**
  - Anandkumar et al. 2012
- **Community detection**
  - Anandkumar et al. 2013a
Why tensor factorization?

- To solve multi-linear algebra problems.
- Parsing
  - Cohen, Satta, and Collins 2013
- Knowledge base completion
  - Chang et al. 2014
  - Singh, Rocktäschel, and Riedel 2015
- Topic modelling
  - Anandkumar et al. 2012
- Community detection
  - Anandkumar et al. 2013a
- Learning latent variable graphical models
  - Anandkumar et al. 2013b
  - TODO: crowdsourcing
  - TODO: others
Existing tensor factorization algorithms

- **Tensor power method** (Anandkumar et al. 2013b)
  - Analog of matrix power method.
  - Sensitive to noise.
  - Restricted to orthogonal tensors.
Existing tensor factorization algorithms

- **Tensor power method** (Anandkumar et al. 2013b)
  - Analog of matrix power method.
  - Sensitive to noise.
  - Restricted to orthogonal tensors.

- **Alternating least squares** (Comon, Luciani, and Almeida 2009; Anandkumar, Ge, and Janzamin 2014)
  - Sensitive to initialization.
Existing tensor factorization algorithms

- **Tensor power method** (Anandkumar et al. 2013b)
  - Analog of matrix power method.
  - Sensitive to noise.
  - Restricted to orthogonal tensors.

- **Alternating least squares** (Comon, Luciani, and Almeida 2009; Anandkumar, Ge, and Janzamin 2014)
  - Sensitive to initialization.

- **Both operate on the tensor directly.**
Our approach

- **Objective:** a fast robust algorithm.
Our approach

- **Objective:** a fast robust algorithm.
- **Approach:** use existing fast and robust matrix algorithms.
Outline

Tensor factorization

Tensor factorization via matrix factorization
  Single matrix factorizations
    Simultaneous matrix factorizations
    Oracle projections
    Random projections

Non-orthogonal tensor factorization

Empirical results

Conclusions
Tensor factorization via single matrix factorization

\[ T = \pi_1 u_1^3 + \pi_2 u_2^3 + \pi_3 u_3^3 + \epsilon R \]
Tensor factorization via single matrix factorization

\[ T = u_1^3 + u_2^3 + u_3^3 \]
Tensor factorization via single matrix factorization

\[
T = u_1^\otimes^3 + u_2^\otimes^3 + u_3^\otimes^3
\]

\[
T(l, l, w) = (w^\top u_1)u_1^\otimes^2 + (w^\top u_2)u_2^\otimes^2 + (w^\top u_3)u_3^\otimes^3
\]
Tensor factorization via single matrix factorization

\[ T = u_1 \otimes^3 1 + u_2 \otimes^3 2 + u_3 \otimes^3 3 \]

\[ T(l, l, w) = (w^\top u_1) u_1 u_1^\top \lambda_1 + (w^\top u_2) u_2 u_2^\top \lambda_2 + (w^\top u_3) u_3 u_3^\top \lambda_3 \]

**Proposal:** Eigen-decomposition on the projected matrix.
Sensitivity of single matrix projection

If two eigenvalues are equal, corresponding eigenvectors are arbitrary.

Problem: Eigendecomposition is very sensitive to the eigengap.

\[ \text{error in factors} \propto \frac{1}{\min(\text{difference in eigenvalues})} \]
Sensitivity of single matrix projection

If two eigenvalues are equal, corresponding eigenvectors are arbitrary.
Sensitivity of single matrix projection

If two eigenvalues are equal, corresponding eigenvectors are arbitrary.
Sensitivity of single matrix projection

If two eigenvalues are equal, corresponding eigenvectors are arbitrary.

**Problem:** Eigendecomposition is very sensitive to the eigengap.

\[
\text{error in factors } \propto \frac{1}{\min(\text{difference in eigenvalues})}.
\]
Projections matter
Projections matter
Projections matter

▶ How can we leverage multiple projections?
Outline

Tensor factorization

Tensor factorization via matrix factorization
  Single matrix factorizations
  Simultaneous matrix factorizations
  Oracle projections
  Random projections

Non-orthogonal tensor factorization

Empirical results

Conclusions
Enter simultaneous diagonalization

\[
T(I, I, w_1) = \sum_{l=1}^{L} \left( w_l^T u_l \right) u_l u_l^T + \sum_{l=2}^{L} \left( w_l^T u_l \right) u_l u_l^T + \sum_{l=3}^{L} \left( w_l^T u_l \right) u_l u_l^T
\]
Enter simultaneous diagonalization

\[ T(I, I, w_1) = \left( w_1^\top u_1 \right) u_1 u_1^\top + \left( w_1^\top u_2 \right) u_2 u_2^\top + \left( w_1^\top u_3 \right) u_3 u_3^\top \]

\[ T(I, I, w_l) = \left( w_l^\top u_1 \right) u_1 u_1^\top + \left( w_l^\top u_2 \right) u_2 u_2^\top + \left( w_l^\top u_3 \right) u_3 u_3^\top \]
Enter simultaneous diagonalization

\[ T(I, I, w_1) = \left( \begin{array}{c} w_1^\top u_1 \\ \lambda_{11} \end{array} \right) u_1 u_1^\top + \left( \begin{array}{c} w_1^\top u_2 \\ \lambda_{21} \end{array} \right) u_2 u_2^\top + \left( \begin{array}{c} w_1^\top u_3 \\ \lambda_{31} \end{array} \right) u_3 u_3^\top \]

\[ T(I, I, w_l) = \left( \begin{array}{c} w_l^\top u_1 \\ \lambda_{11} \end{array} \right) u_1 u_1^\top + \left( \begin{array}{c} w_l^\top u_2 \\ \lambda_{21} \end{array} \right) u_2 u_2^\top + \left( \begin{array}{c} w_l^\top u_3 \\ \lambda_{31} \end{array} \right) u_3 u_3^\top \]

- Projections share factors.
Algorithm

**Algorithm**: Simultaneously diagonalize projected matrices.

\[ \hat{U} = \arg \max_{\hat{U}} \sum_{l=1}^{L} \text{off}(U^T M_l U) \]

\[ \text{off}(A) = \sum_{i \neq j} A_{ij}^2. \]
Algorithm

- **Algorithm**: Simultaneously diagonalize projected matrices.

\[
\hat{U} = \arg \max_{\hat{U}} \sum_{l=1}^{L} \text{off}(U^\top M_l U) \quad \text{off}(A) = \sum_{i \neq j} A_{ij}^2.
\]

- Optimize using the Jacobi angles (Cardoso and Souloumiac 1996).
Algorithm

- **Algorithm**: Simultaneously diagonalize projected matrices.

\[
\hat{U} = \arg \max_{\hat{U}} \sum_{l=1}^{L} \text{off}(U^\top M_l U) \quad \text{off}(A) = \sum_{i \neq j} A_{ij}^2.
\]

- Optimize using the Jacobi angles (Cardoso and Souloumiac 1996).
- Multiple projections proposed in Anandkumar, Hsu, and Kakade 2012, but didn’t use simultaneous diagonalization.
Comparison with single matrix factorization

- Single matrix factorization depends on minimum eigengap.

\[
\text{error in factors } \propto \frac{1}{\min_{i,j} \text{difference in eigenvalues}}.
\]
Comparison with single matrix factorization

- Single matrix factorization depends on **minimum eigengap**.

  \[
  \text{error in factors } \propto \frac{1}{\min_{i,j} \text{difference in eigenvalues}}.
  \]

- Simultaneous matrix factorization depends on **average eigengap**.

  \[
  \text{error in factors } \propto \frac{1}{\min_{i,j} \text{average difference in eigenvalues}}.
  \]
Comparison with single matrix factorization

- Single matrix factorization depends on minimum eigengap.

\[
\text{error in factors} \propto \frac{1}{\min_{i,j} |\lambda_i - \lambda_j|}.
\]

- Simultaneous matrix factorization depends on average eigengap.

\[
\text{error in factors} \propto \frac{1}{\min_{i,j} \sum_{l=1}^L |\lambda_{il} - \lambda_{jl}|}.
\]
Outline

Tensor factorization

Tensor factorization via matrix factorization
  Single matrix factorizations
  Simultaneous matrix factorizations
  Oracle projections
  Random projections

Non-orthogonal tensor factorization

Empirical results

Conclusions
Oracle projections

Theorem

Pick $k$ projections along the factors $(u_i)$. Then,
Oracle projections

Theorem

*Pick k projections along the factors* \((u_i)\). Then,

\[
\text{error in factors} \leq O \left( \sqrt{\frac{\pi_{\text{max}}}{\pi_{\text{min}}^2}} \right) \epsilon.
\]
Oracle projections

Theorem

Pick $k$ projections along the factors $(u_i)$. Then,

$$\text{error in factors} \leq O\left(\frac{\sqrt{\pi_{\text{max}}}}{\pi_{\text{min}}^2}\right) \epsilon.$$
Outline

Tensor factorization

Tensor factorization via matrix factorization
  Single matrix factorizations
  Simultaneous matrix factorizations
  Oracle projections
  Random projections

Non-orthogonal tensor factorization

Empirical results

Conclusions
Theorem

*Pick* $O(k \log k)$ *projections* randomly from the unit sphere.

*Then, with probability* $> 1 - \delta$, 

As good as having oracle projections!
Random projections

Theorem

Pick $O(k \log k)$ projections randomly from the unit sphere. Then, with probability $> 1 - \delta$,

\[
\text{error in factors} \leq O \left( \frac{\sqrt{\pi_{\text{max}}}}{\pi_{\text{min}}^2} \right) \epsilon + C(\delta) \epsilon
\]
Theorem

Pick $O(k \log k)$ projections randomly from the unit sphere. Then, with probability $> 1 - \delta$,

$\text{error in factors} \leq O\left(\frac{\sqrt{\pi_{\text{max}}}}{\pi_{\text{min}}^2}\right) \epsilon + C(\delta)\epsilon$

As good as having oracle projections!
Random projections

**Theorem**

*Pick $O(k \log k)$ projections randomly from the unit sphere.*

*Then, with probability $> 1 - \delta$,*

\[
\text{error in factors} \leq O\left(\frac{\sqrt{\pi_{\text{max}}}}{\pi_{\text{min}}^2}\right) \epsilon + C(\delta) \epsilon
\]

- As good as having oracle projections!
Final algorithm

- **Algorithm:**

  1. Project tensor onto $O(k \log k)$ random vectors.
  2. Recover approximate factors $\tilde{u}_i(0)$ through simultaneous diagonalization.
  3. Project tensor onto approximated factors.
  4. Return factors $\tilde{u}_i$ from simultaneous diagonalization.
Final algorithm

- **Algorithm:**
  - Project tensor on to $O(k \log k)$ random vectors.
Final algorithm

- Algorithm:
  - Project tensor on to $O(k \log k)$ random vectors.
  - Recover approximate factors $\tilde{u}_i^{(0)}$ through simultaneous diagonalization.
Final algorithm

- **Algorithm:**
  - Project tensor on to $O(k \log k)$ random vectors.
  - Recover approximate factors $\tilde{u}_i^{(0)}$ through simultaneous diagonalization.
  - Project tensor on to approximated factors.
Final algorithm

Algorithm:

- Project tensor on to $O(k \log k)$ random vectors.
- Recover approximate factors $\tilde{u}_i^{(0)}$ through simultaneous diagonalization.
- Project tensor on to approximated factors.
- Return factors $\tilde{u}_i$ from simultaneous diagonalization.
Outline

Tensor factorization

Tensor factorization via matrix factorization
  Single matrix factorizations
  Simultaneous matrix factorizations
  Oracle projections
  Random projections

Non-orthogonal tensor factorization

Empirical results

Conclusions
Naive approach: whitening non-orthogonal factors

Use a whitening transformation to orthogonalize tensor (Anandkumar et al. 2013b).

Is a major source of errors itself (Souloumiac 2009).
Naive approach: whitening non-orthogonal factors

- Use a whitening transformation to orthogonalize tensor (Anandkumar et al. 2013b).
Naive approach: whitening non-orthogonal factors

- Use a whitening transformation to orthogonalize tensor (Anandkumar et al. 2013b).
  - Is a major source of errors itself (Souloumiac 2009).
Non-orthogonal simultaneous diagonalization

\[
T(I, I, w_1) = M_1 = \begin{pmatrix} w_1^\top & u_1 & u_1^\top \\ \lambda_{11} \\ \vdots \\ \lambda_{31} \end{pmatrix} u_1 u_1^\top + \begin{pmatrix} w_1^\top & u_2 & u_2^\top \\ \lambda_{21} \\ \vdots \\ \lambda_{31} \end{pmatrix} u_2 u_2^\top + \begin{pmatrix} w_1^\top & u_3 & u_3^\top \\ \lambda_{31} \\ \vdots \\ \lambda_{31} \end{pmatrix} u_3 u_3^\top
\]

- No unique non-orthogonal factorization for a single matrix.
Non-orthogonal simultaneous diagonalization

\[
\begin{align*}
T(I, I, w_1) &= \left( w_1^\top u_1 \right) u_1 u_1^\top + \left( w_1^\top u_2 \right) u_2 u_2^\top + \left( w_1^\top u_3 \right) u_3 u_3^\top \\
& \quad \vdots \\
T(I, I, w_l) &= \left( w_l^\top u_1 \right) u_1 u_1^\top + \left( w_l^\top u_2 \right) u_2 u_2^\top + \left( w_l^\top u_3 \right) u_3 u_3^\top
\end{align*}
\]

- No unique non-orthogonal factorization for a single matrix.
- \( \geq 2 \) matrices have a unique non-orthogonal factorization.
Non-orthogonal simultaneous diagonalization

\[
T(I, I, w_1) = \sum \lambda_i \underbrace{u_i u_i^\top}_{M_i}
\]

\[
T(I, I, w_l) = \sum \lambda_i \underbrace{u_i u_i^\top}_{M_i}
\]

- No unique non-orthogonal factorization for a single matrix.
- \( \geq 2 \) matrices have a unique non-orthogonal factorization.
- **Note:** \( \lambda_i \) are factor weights, not eigenvalues.
Non-orthogonal simultaneous diagonalization

**Algorithm:** Simultaneously diagonalize projected matrices.

\[
\hat{U} = \arg \max_{\hat{U}} \sum_{l=1}^{L} \text{off}(U^{-1} M_l U^{-\top}) \quad \text{off}(A) = \sum_{i \neq j} A_{ij}^2.
\]
Non-orthogonal simultaneous diagonalization

**Algorithm:** Simultaneously diagonalize projected matrices.

\[ \hat{U} = \arg \max_{\hat{U}} \sum_{l=1}^{L} \text{off}(U^{-1} M_l U^{-\top}) \]

\[ \text{off}(A) = \sum_{i \neq j} A_{ij}^2. \]

- \( U \) are not constrained to be orthogonal.
Non-orthogonal simultaneous diagonalization

▶ **Algorithm:** Simultaneously diagonalize projected matrices.

\[
\hat{U} = \arg \max_{\hat{U}} \sum_{l=1}^{L} \text{off}(U^{-1} M_l U^{-\top}) \quad \text{off}(A) = \sum_{i \neq j} A_{ij}^2.
\]

▶ \(U\) are not constrained to be orthogonal.
▶ Optimize using the QR1JD algorithm (Souloumiac 2009).
  ▶ Only guaranteed to have local convergence.
Results: Non-orthogonal simultaneous diagonalization

Theorem (Oracle projections)

Pick $k$ projections along the factors $(u_i)$. Then,

$$\text{error in factors} \leq O \left( \| U^{-T} \|_2^3 \frac{\sqrt{\pi_{\text{max}}}}{\pi_{\text{min}}^2} \right) \epsilon,$$

where $U = [u_1 | \cdots | u_k]$. 
Results: Non-orthogonal simultaneous diagonalization

**Theorem (Oracle projections)**

*Pick k projections along the factors (\(u_i\)). Then,*

\[
\text{error in factors} \leq O \left( \| U^{-T} \|_2^3 \frac{\sqrt{\pi_{\text{max}}}}{\pi_{\text{min}}^2} \right) \epsilon,
\]

*where* \(U = [u_1 | \cdots | u_k]\).
Outline

Tensor factorization

Tensor factorization via matrix factorization
  Single matrix factorizations
  Simultaneous matrix factorizations
  Oracle projections
  Random projections

Non-orthogonal tensor factorization

Empirical results

Conclusions
Outline

Tensor factorization

Tensor factorization via matrix factorization
  Single matrix factorizations
  Simultaneous matrix factorizations
  Oracle projections
  Random projections

Non-orthogonal tensor factorization

Empirical results

Conclusions
Reduce tensor problems to matrix ones with $\tilde{O}(k)$ random projections.
Conclusions

- Reduce tensor problems to matrix ones with $\tilde{O}(k)$ random projections.
- **Robust** to noise with general support for non-orthogonal factors.
Conclusions

- Reduce tensor problems to matrix ones with $\tilde{O}(k)$ random projections.
- Robust to noise with general support for non-orthogonal factors.
- Competitive empirical performance.
Conclusions

- Reduce tensor problems to matrix ones with $\tilde{O}(k)$ random projections.
- Robust to noise with general support for non-orthogonal factors.
- Competitive empirical performance.
- Questions?