

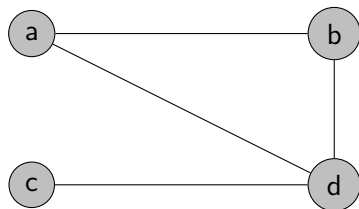
# Tensor Factorization via Matrix Factorization

Volodymyr Kuleshov\*  
**Arun Tejasvi Chaganty\***  
Percy Liang

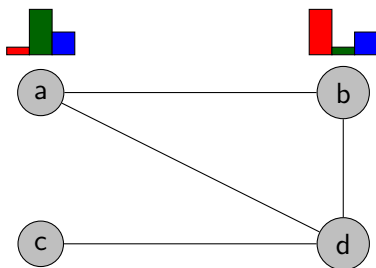
Stanford University

May 11, 2015

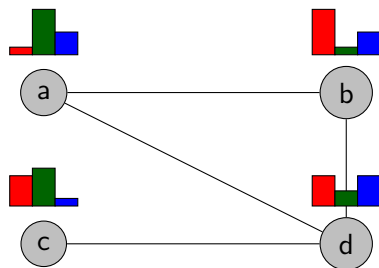
# An application: community detection



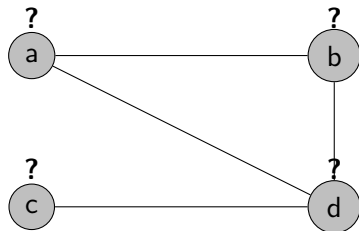
# An application: community detection



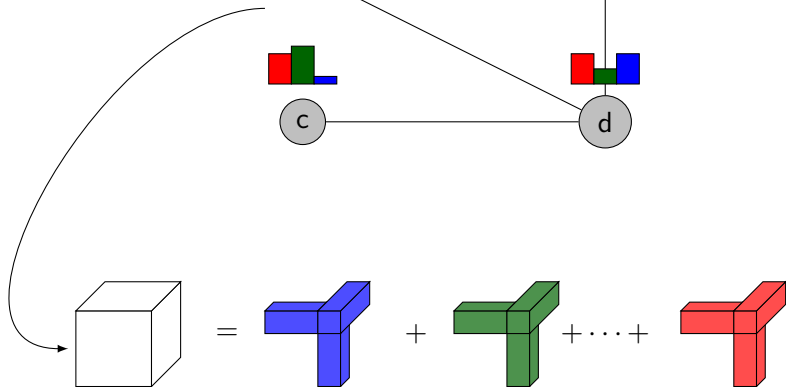
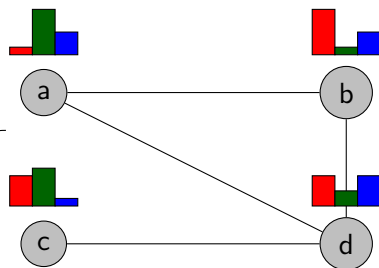
# An application: community detection



# An application: community detection



## An application: community detection

Anandkumar, Ge, Hsu, and  
S. Kakade 2013

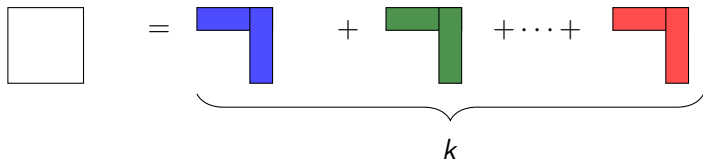
# Applications of tensor factorization

- ▶ **Community detection**
  - ▶ Anandkumar, Ge, Hsu, and S. Kakade 2013
- ▶ **Parsing**
  - ▶ Cohen, Satta, and Collins 2013
- ▶ **Knowledge base completion**
  - ▶ Chang et al. 2014
  - ▶ Singh, Rocktäschel, and Riedel 2015
- ▶ **Topic modelling**
  - ▶ Anandkumar, Foster, et al. 2012
- ▶ **Crowdsourcing**
  - ▶ Zhang et al. 2014
- ▶ **Mixture models**
  - ▶ Anandkumar, Ge, Hsu, S. M. Kakade, et al. 2013
- ▶ **Bottlenecked models**
  - ▶ Chaganty and Liang 2014
- ▶ ...

# What is tensor (CP) factorization?

- ▶ Tensor analogue of matrix eigen-decomposition.

$$M = \sum_{i=1}^k \pi_i u_i \otimes u_i \quad .$$

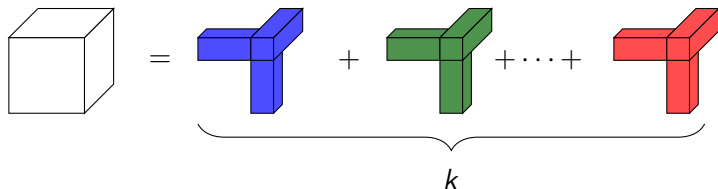




# What is tensor (CP) factorization?

- ▶ Tensor analogue of matrix eigen-decomposition.

$$T = \sum_{i=1}^k \pi_i u_i \otimes u_i \otimes u_i \quad .$$

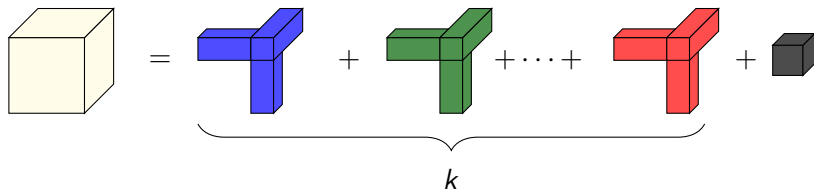


# What is tensor (CP) factorization?

- ▶ 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**  $u_i$ .

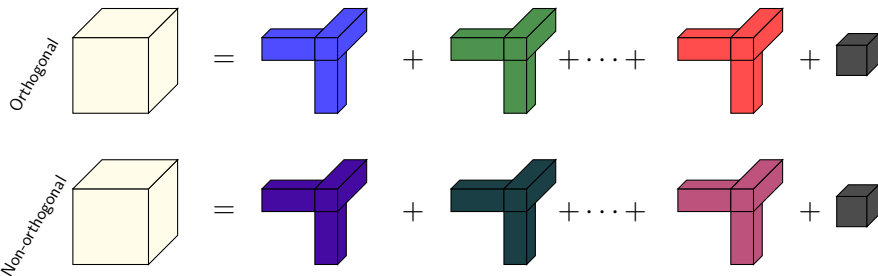


# What is tensor (CP) factorization?

- ▶ 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**  $u_i$ .



# Existing tensor factorization algorithms

- ▶ **Tensor power method** (Anandkumar, Ge, Hsu, S. M. Kakade, et al. 2013)
  - ▶ Analog of matrix power method.
  - ▶ Sensitive to noise.
  - ▶ Restricted to orthogonal tensors.

# Existing tensor factorization algorithms

- ▶ **Tensor power method** (Anandkumar, Ge, Hsu, S. M. Kakade, et al. 2013)
  - ▶ 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, Ge, Hsu, S. M. Kakade, et al. 2013)
  - ▶ 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.

**Our approach:** reduce to existing **fast and robust matrix algorithms**.

# Outline

Introduction: tensor factorization

**Orthogonal Tensor factorization**  
Projections

Non-orthogonal tensor factorization

Related work

Empirical results

Conclusions

## Tensor factorization via single matrix factorization

$$T = \pi_1 u_1^{\otimes 3} + \pi_2 u_2^{\otimes 3} + \pi_3 u_3^{\otimes 3} + \epsilon R$$



## Tensor factorization via single matrix factorization

The diagram shows a white cube labeled  $T$  on the left. To its right is an equals sign, followed by three terms separated by plus signs. Each term consists of a colored L-shaped block (blue, green, and red respectively) and the expression  $u_1^{\otimes 3}$  below it. The blue block is oriented with its horizontal arm pointing left and its vertical stem pointing down. The green block is oriented with its horizontal arm pointing right and its vertical stem pointing down. The red block is oriented with its horizontal arm pointing left and its vertical stem pointing down.

$$T = u_1^{\otimes 3} + u_1^{\otimes 3} + u_1^{\otimes 3}$$

## Tensor factorization via single matrix factorization

The diagram illustrates the factorization of a 3D tensor  $T$  into three rank-1 tensors (blue, green, red) and its corresponding matrix factorization  $T(l, l, w)$  into three rank-1 matrices.

Top row:  $T = u_1^{\otimes 3} + u_1^{\otimes 3} + u_1^{\otimes 3}$  (Note: The image shows three distinct rank-1 tensors in blue, green, and red, each labeled  $u_1^{\otimes 3}$  in blue text).

Bottom row:  $T(l, l, w) = (w^T u_1) u_1^{\otimes 2} + (w^T u_2) u_2^{\otimes 2} + (w^T u_3) u_3^{\otimes 2}$

## Tensor factorization via single matrix factorization

$$\begin{array}{c}
 \text{Cube } T \\
 \downarrow \\
 \text{Matrix } T(l, l, w)
 \end{array}
 =
 \begin{array}{c}
 \text{Blue L-tensor } u_1^{\otimes 3} \\
 \text{Green L-tensor } u_1^{\otimes 3} \\
 \text{Red L-tensor } u_1^{\otimes 3}
 \end{array}
 +
 \begin{array}{c}
 \text{Blue L-tensor } \underbrace{(w^\top u_1) u_1^{\otimes 2}}_{\lambda_1} \\
 \text{Green L-tensor } \underbrace{(w^\top u_2) u_2^{\otimes 2}}_{\lambda_2} \\
 \text{Red L-tensor } \underbrace{(w^\top u_3) u_3^{\otimes 2}}_{\lambda_3}
 \end{array}$$

- ▶ **Proposal:** Eigen-decomposition on the projected matrix.
- ▶ **Return:** recovered eigenvectors,  $u_j$ .

## Sensitivity of single matrix projection

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

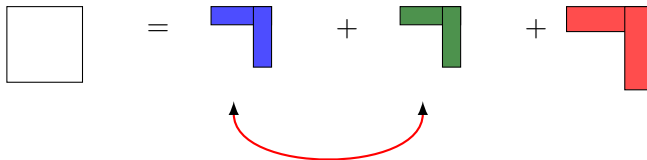
$$\text{error in factors} \propto \frac{1}{\min(\text{difference in eigenvalues})}$$

## Sensitivity of single matrix projection

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

$$\text{error in factors} \propto \frac{1}{\min(\text{difference in eigenvalues})}$$

- ▶ **Intuition:** If two eigenvalues are equal, corresponding eigenvectors are arbitrary.

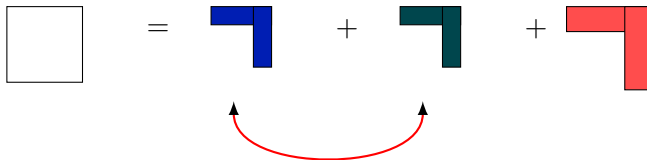


## Sensitivity of single matrix projection

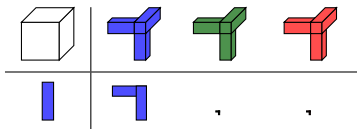
- ▶ **Problem:** Eigendecomposition is very sensitive to the **eigengap**.

$$\text{error in factors} \propto \frac{1}{\min(\text{difference in eigenvalues})}$$

- ▶ **Intuition:** If two eigenvalues are equal, corresponding eigenvectors are arbitrary.



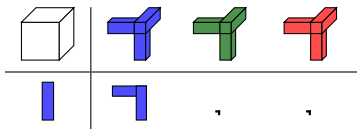
## Sensitivity analysis (contd.)



## Sensitivity analysis (contd.)

- ▶ Single matrix factorization:

$$\text{error in factors} \propto \frac{1}{\text{min diff. in eigenvalues}}.$$

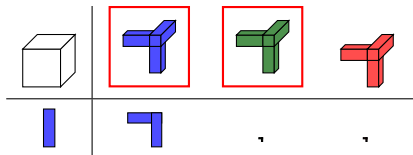




## Sensitivity analysis (contd.)

- ▶ Single matrix factorization:

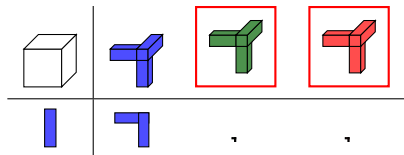
$$\text{error in factors} \propto \frac{1}{\text{min diff. in eigenvalues}}.$$



## Sensitivity analysis (contd.)

- ▶ Single matrix factorization:

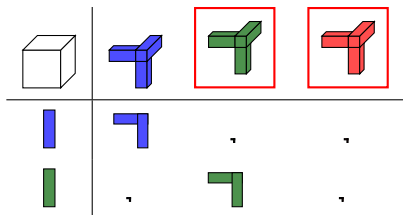
$$\text{error in factors} \propto \frac{1}{\text{min diff. in eigenvalues}}.$$



## Sensitivity analysis (contd.)

- ▶ Single matrix factorization:

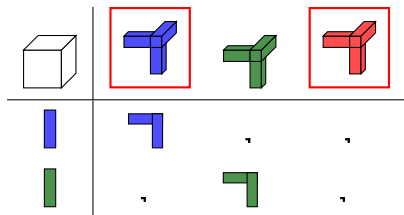
$$\text{error in factors} \propto \frac{1}{\text{min diff. in eigenvalues}}.$$



## Sensitivity analysis (contd.)

- ▶ Single matrix factorization:

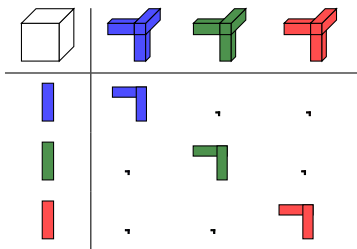
$$\text{error in factors} \propto \frac{1}{\text{min diff. in eigenvalues}}.$$



## Sensitivity analysis (contd.)

- ▶ Single matrix factorization:

$$\text{error in factors} \propto \frac{1}{\text{min diff. in eigenvalues}}.$$



## Sensitivity analysis (contd.)

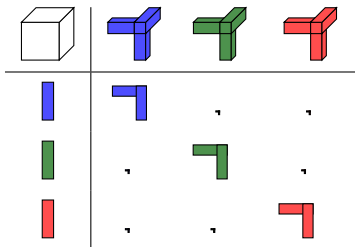
(Cardoso 1994)

- ▶ Single matrix factorization:

$$\text{error in factors} \propto \frac{1}{\text{min diff. in eigenvalues}}.$$

- ▶ Simultaneous matrix factorization:

$$\text{error in factors} \propto \frac{1}{\text{min avg. diff. in eigenvalues}}.$$



## Sensitivity analysis (contd.)

(Cardoso 1994)

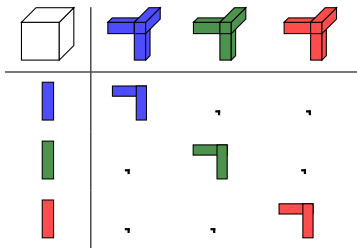
- ▶ Single matrix factorization:

$$\frac{\text{error in factors} \propto 1}{\text{min diff. in eigenvalues}}.$$

- ▶ Simultaneous matrix factorization:

$$\frac{\text{error in factors} \propto 1}{\text{min avg. diff. in eigenvalues}}.$$

**Every coordinate pair needs one good projection** (with a large eigengap).



## Sensitivity analysis (contd.)

(Cardoso 1994)

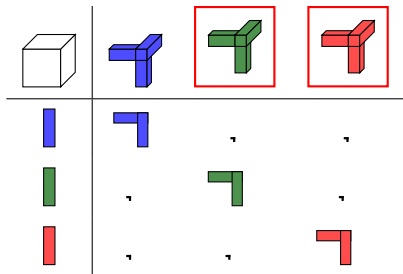
- ▶ Single matrix factorization:

$$\text{error in factors} \propto \frac{1}{\text{min diff. in eigenvalues}}.$$

- ▶ Simultaneous matrix factorization:

$$\text{error in factors} \propto \frac{1}{\text{min avg. diff. in eigenvalues}}.$$

**Every coordinate pair needs one good projection (with a large eigengap).**





## Sensitivity analysis (contd.)

(Cardoso 1994)

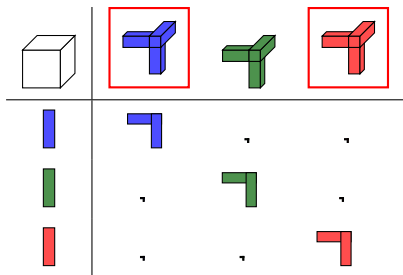
- ▶ Single matrix factorization:

$$\text{error in factors} \propto \frac{1}{\text{min diff. in eigenvalues}}.$$

- ▶ Simultaneous matrix factorization:

$$\text{error in factors} \propto \frac{1}{\text{min avg. diff. in eigenvalues}}.$$

**Every coordinate pair needs one good projection (with a large eigengap).**



## Sensitivity analysis (contd.)

(Cardoso 1994)

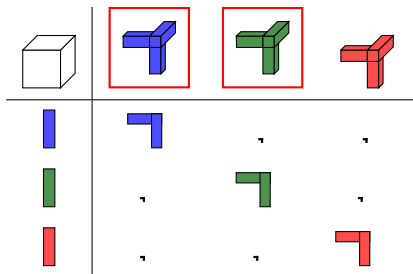
- ▶ Single matrix factorization:

$$\frac{\text{error in factors} \propto 1}{\text{min diff. in eigenvalues}}.$$

- ▶ Simultaneous matrix factorization:

$$\frac{\text{error in factors} \propto 1}{\text{min avg. diff. in eigenvalues}}.$$

**Every coordinate pair needs one good projection (with a large eigengap).**



## Reduction to simultaneous diagonalization

$$\underbrace{T(I, I, w_1)}_{M_1} = \underbrace{(w_1^\top u_1)}_{\lambda_{11}} u_1 u_1^\top + \underbrace{(w_1^\top u_2)}_{\lambda_{21}} u_2 u_2^\top + \underbrace{(w_1^\top u_3)}_{\lambda_{31}} u_3 u_3^\top$$

## Reduction to simultaneous diagonalization

$$\begin{array}{c}
 \square \\
 \underbrace{T(I, I, w_1)}_{M_1} \\
 \vdots \\
 \square \\
 \underbrace{T(I, I, w_\ell)}_{M_\ell}
 \end{array}
 =
 \begin{array}{c}
 \color{blue}{\lrcorner} \\
 \underbrace{(w_1^\top u_1)}_{\lambda_{11}} u_1 u_1^\top \\
 \vdots \\
 \color{blue}{\lrcorner} \\
 \underbrace{(w_\ell^\top u_1)}_{\lambda_{1\ell}} u_1 u_1^\top
 \end{array}
 +
 \begin{array}{c}
 \color{green}{\lrcorner} \\
 \underbrace{(w_1^\top u_2)}_{\lambda_{21}} u_2 u_2^\top \\
 \vdots \\
 \color{green}{\lrcorner} \\
 \underbrace{(w_\ell^\top u_2)}_{\lambda_{2\ell}} u_2 u_2^\top
 \end{array}
 +
 \begin{array}{c}
 \color{red}{\lrcorner} \\
 \underbrace{(w_1^\top u_3)}_{\lambda_{31}} u_3 u_3^\top \\
 \vdots \\
 \color{red}{\lrcorner} \\
 \underbrace{(w_\ell^\top u_3)}_{\lambda_{3\ell}} u_3 u_3^\top
 \end{array}$$

## Reduction to simultaneous diagonalization

$$\begin{array}{c}
 \square \\
 \underbrace{T(I, I, w_1)}_{M_1} \\
 \vdots \\
 \square \\
 \underbrace{T(I, I, w_\ell)}_{M_\ell}
 \end{array}
 =
 \begin{array}{c}
 \color{blue}\lrcorner \\
 \underbrace{(w_1^\top u_1)}_{\lambda_{11}} u_1 u_1^\top \\
 \vdots \\
 \color{blue}\lrcorner \\
 \underbrace{(w_\ell^\top u_1)}_{\lambda_{1\ell}} u_1 u_1^\top
 \end{array}
 +
 \begin{array}{c}
 \color{green}\lrcorner \\
 \underbrace{(w_1^\top u_2)}_{\lambda_{21}} u_2 u_2^\top \\
 \vdots \\
 \color{green}\lrcorner \\
 \underbrace{(w_\ell^\top u_2)}_{\lambda_{2\ell}} u_2 u_2^\top
 \end{array}
 +
 \begin{array}{c}
 \color{red}\lrcorner \\
 \underbrace{(w_1^\top u_3)}_{\lambda_{31}} u_3 u_3^\top \\
 \vdots \\
 \color{red}\lrcorner \\
 \underbrace{(w_\ell^\top u_3)}_{\lambda_{3\ell}} u_3 u_3^\top
 \end{array}$$

- **Projections share factors:** can be simultaneously diagonalized.

# Simultaneous diagonalization algorithm

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

$$\hat{U} = \arg \min_{U: UU^T=I} \sum_{\ell=1}^L \text{off}(U^T M_\ell U) \quad \text{off}(A) = \sum_{i \neq j} A_{ij}^2.$$

# Simultaneous diagonalization algorithm

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

$$\hat{U} = \arg \min_{U: UU^T=I} \sum_{\ell=1}^L \text{off}(U^T M_{\ell} U) \quad \text{off}(A) = \sum_{i \neq j} A_{ij}^2.$$

# Simultaneous diagonalization algorithm

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

$$\hat{U} = \arg \min_{U: UU^T=I} \sum_{\ell=1}^L \text{off}(U^T M_{\ell} U) \quad \text{off}(A) = \sum_{i \neq j} A_{ij}^2.$$



# Simultaneous diagonalization algorithm

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

$$\hat{U} = \arg \min_{U: UU^T=I} \sum_{\ell=1}^L \text{off}(U^T M_{\ell} U) \quad \text{off}(A) = \sum_{i \neq j} A_{ij}^2$$

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

# Simultaneous diagonalization algorithm

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

$$\hat{U} = \arg \min_{U: UU^T=I} \sum_{\ell=1}^L \text{off}(U^T M_{\ell} U) \quad \text{off}(A) = \sum_{i \neq j} A_{ij}^2.$$

- ▶ Optimize using Jacobi angles (Cardoso and Souloumiac 1996).
  - ▶ Used widely in the ICA community.
  - ▶ Empirically appears to be generically global convergence.

# Outline

Introduction: tensor factorization

Orthogonal Tensor factorization  
Projections

Non-orthogonal tensor factorization

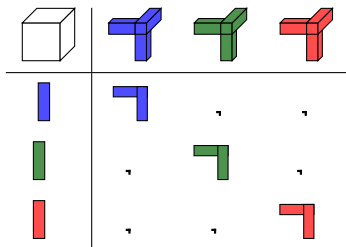
Related work

Empirical results

Conclusions

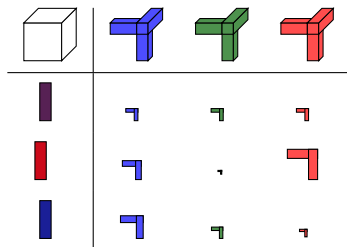
# Oracle and random projections

- ▶ **Hypothetically:** “oracle” projections **along the factors** is good.



# Oracle and random projections

- ▶ **Hypothetically:** “oracle” projections **along the factors** is good.
- ▶ **Practically:** use projections **along random directions.**



# Results: Orthogonal tensor decomposition

$$\hat{T} = \sum_{i=1}^k \pi_i u_i^{\otimes 3} + \epsilon R.$$

# Results: Orthogonal tensor decomposition

$$\hat{T} = \sum_{i=1}^k \pi_i u_i^{\otimes 3} + \epsilon R.$$

## Theorem (Random projections)

Pick  $L = \Omega(k \log k)$  projections **randomly from the unit sphere**. Then, with high probability,

$$\text{error in factors} \leq O \left( \frac{\sqrt{\|\pi\|_1 \pi_{\max}}}{\pi_{\min}^2} + \sqrt{\frac{d}{L}} \right) \epsilon$$

# Results: Orthogonal tensor decomposition

$$\hat{T} = \sum_{i=1}^k \pi_i u_i^{\otimes 3} + \epsilon R.$$

## Theorem (Random projections)

Pick  $L = \Omega(k \log k)$  projections **randomly from the unit sphere**. Then, with high probability,

$$\text{error in factors} \leq O \left( \underbrace{\frac{\sqrt{\|\pi\|_1 \pi_{\max}}}{\pi_{\min}^2}}_{\text{oracle error}} + \sqrt{\frac{d}{L}} \right) \epsilon$$



# Results: Orthogonal tensor decomposition

$$\hat{T} = \sum_{i=1}^k \pi_i u_i^{\otimes 3} + \epsilon R.$$

## Theorem (Random projections)

Pick  $L = \Omega(k \log k)$  projections **randomly from the unit sphere**. Then, with high probability,

$$\text{error in factors} \leq O \left( \underbrace{\frac{\sqrt{\|\pi\|_1 \pi_{\max}}}{\pi_{\min}^2}}_{\text{oracle error}} + \underbrace{\sqrt{\frac{d}{L}}}_{\text{conc. term}} \right) \epsilon$$

# Empirical: Random vs. oracle projections

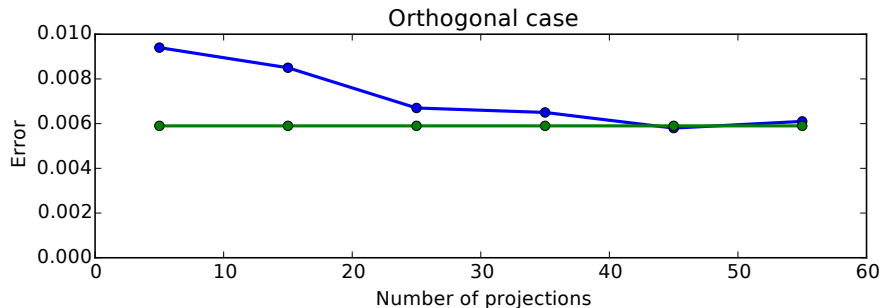


Figure: Comparing random vs. oracle projections ( $d = k = 10$ ,  $\epsilon = 0.05$ )

# Outline

Introduction: tensor factorization

Orthogonal Tensor factorization  
Projections

**Non-orthogonal tensor factorization**

Related work

Empirical results

Conclusions

# Non-orthogonal simultaneous diagonalization

$$\underbrace{T(I, I, w_1)}_{M_1} = \underbrace{(w_1^\top u_1)}_{\lambda_{11}} u_1 u_1^\top + \underbrace{(w_1^\top u_2)}_{\lambda_{21}} u_2 u_2^\top + \underbrace{(w_1^\top u_3)}_{\lambda_{31}} u_3 u_3^\top$$

- ▶ No unique non-orthogonal factorization for a single matrix.

# Non-orthogonal simultaneous diagonalization

$$\begin{array}{c}
 \square \\
 \underbrace{T(I, I, w_1)}_{M_1} \\
 \vdots \\
 \square \\
 \underbrace{T(I, I, w_\ell)}_{M_\ell}
 \end{array}
 =
 \begin{array}{c}
 \underbrace{(w_1^\top u_1)}_{\lambda_{11}} u_1 u_1^\top \\
 \vdots \\
 \underbrace{(w_\ell^\top u_1)}_{\lambda_{1\ell}} u_1 u_1^\top
 \end{array}
 +
 \begin{array}{c}
 \underbrace{(w_1^\top u_2)}_{\lambda_{21}} u_2 u_2^\top \\
 \vdots \\
 \underbrace{(w_\ell^\top u_2)}_{\lambda_{2\ell}} u_2 u_2^\top
 \end{array}
 +
 \begin{array}{c}
 \underbrace{(w_1^\top u_3)}_{\lambda_{31}} u_3 u_3^\top \\
 \vdots \\
 \underbrace{(w_\ell^\top u_3)}_{\lambda_{3\ell}} u_3 u_3^\top
 \end{array}$$

- ▶ No unique non-orthogonal factorization for a single matrix.
- ▶  $\geq 2$  matrices have a unique non-orthogonal factorization.

# Non-orthogonal simultaneous diagonalization

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

$$\hat{U} = \arg \min_U \sum_{\ell=1}^L \text{off}(U^{-1}M_{\ell}U^{-\top}) \quad \text{off}(A) = \sum_{i \neq j} A_{ij}^2.$$

# Non-orthogonal simultaneous diagonalization

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

$$\hat{U} = \arg \min_U \sum_{\ell=1}^L \text{off}(U^{-1} M_{\ell} U^{-T}) \quad \text{off}(A) = \sum_{i \neq j} A_{ij}^2.$$

- ▶  $U$  are not constrained to be orthogonal.

# Non-orthogonal simultaneous diagonalization

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

$$\hat{U} = \arg \min_U \sum_{\ell=1}^L \text{off}(U^{-1} M_{\ell} U^{-\top}) \quad \text{off}(A) = \sum_{i \neq j} A_{ij}^2.$$

- ▶  $U$  are not constrained to be orthogonal.
- ▶ Optimize using the QR1JD algorithm (Afsari 2006).
  - ▶ Only guaranteed to have local convergence.
  - ▶ More stable than ALS in practice.



# Non-orthogonal simultaneous diagonalization

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

$$\hat{U} = \arg \min_U \sum_{\ell=1}^L \text{off}(U^{-1} M_{\ell} U^{-\top}) \quad \text{off}(A) = \sum_{i \neq j} A_{ij}^2.$$

- ▶  $U$  are not constrained to be orthogonal.
- ▶ Optimize using the QR1JD algorithm (Afsari 2006).
  - ▶ Only guaranteed to have local convergence.
  - ▶ More stable than ALS in practice.
- ▶ Sensitivity analysis due to Afsari 2008

# Results: Non-orthogonal tensor decomposition

## Theorem (Random projections)

Pick  $L = \Omega(k \log k)$  projections **randomly from the unit sphere**. Then, with high probability,

$$\text{error in factors} \leq O \left( \frac{\|U^{-\top}\|_2^2}{1 - \mu^2} \frac{\sqrt{\|\pi\|_1 \pi_{\max}}}{\pi_{\min}^2} \left( 1 + \sqrt{\frac{d}{L}} \right) \epsilon \right)$$

where  $U = [u_1 | \dots | u_k]$ ,  $\mu = \max u_i^\top u_j$  and  $\frac{\|U^{-\top}\|_2^2}{1 - \mu^2}$  measures non-orthogonality.

# Results: Non-orthogonal tensor decomposition

## Theorem (Random projections)

Pick  $L = \Omega(k \log k)$  projections **randomly from the unit sphere**. Then, with high probability,

$$\text{error in factors} \leq O \left( \frac{\|U^{-\top}\|_2^2}{1 - \mu^2} \underbrace{\frac{\sqrt{\|\pi\|_1 \pi_{\max}}}{\pi_{\min}^2} \left(1 + \sqrt{\frac{d}{L}}\right)}_{\text{ortho. cost}} \right) \epsilon$$

where  $U = [u_1 | \dots | u_k]$ ,  $\mu = \max u_i^\top u_j$  and  $\frac{\|U^{-\top}\|_2^2}{1 - \mu^2}$  measures non-orthogonality.

# Results: Non-orthogonal tensor decomposition

## Theorem (Random projections)

Pick  $L = \Omega(k \log k)$  projections **randomly from the unit sphere**. Then, with high probability,

$$\text{error in factors} \leq O \left( \underbrace{\frac{\|U^{-T}\|_2^2}{1 - \mu^2}}_{\text{non-ortho. cost}} \underbrace{\frac{\sqrt{\|\pi\|_1 \pi_{\max}}}{\pi_{\min}^2} \left(1 + \sqrt{\frac{d}{L}}\right)}_{\text{ortho. cost}} \right) \epsilon$$

where  $U = [u_1 | \dots | u_k]$ ,  $\mu = \max_i u_i^\top u_j$  and  $\frac{\|U^{-T}\|_2^2}{1 - \mu^2}$  measures non-orthogonality.

# Outline

Introduction: tensor factorization

Orthogonal Tensor factorization  
Projections

Non-orthogonal tensor factorization

Related work

Empirical results

Conclusions

## Notes and Related work

- ▶ Orthogonal tensor methods can factorize non-orthogonal tensors using a whitening transformation (Anandkumar, Ge, Hsu, S. M. Kakade, et al. 2013).
  - ▶ Is a major source of errors itself (Souloumiac 2009).

## Notes and Related work

- ▶ Orthogonal tensor methods can factorize non-orthogonal tensors using a whitening transformation (Anandkumar, Ge, Hsu, S. M. Kakade, et al. 2013).
  - ▶ Is a major source of errors itself (Souloumiac 2009).
- ▶ Simultaneous diagonalization for tensors proposed by Lathauwer 2006.
  - ▶ Relies on computing the SVD of a  $d^4 \times k^2$  matrix.

## Notes and Related work

- ▶ Orthogonal tensor methods can factorize non-orthogonal tensors using a whitening transformation (Anandkumar, Ge, Hsu, S. M. Kakade, et al. 2013).
  - ▶ Is a major source of errors itself (Souloumiac 2009).
- ▶ Simultaneous diagonalization for tensors proposed by Lathauwer 2006.
  - ▶ Relies on computing the SVD of a  $d^4 \times k^2$  matrix.
- ▶ Simultaneous diagonalizations for multiple projections mentioned in Anandkumar, Ge, Hsu, S. M. Kakade, et al. 2013.
  - ▶ No analysis presented.



# Outline

Introduction: tensor factorization

Orthogonal Tensor factorization  
Projections

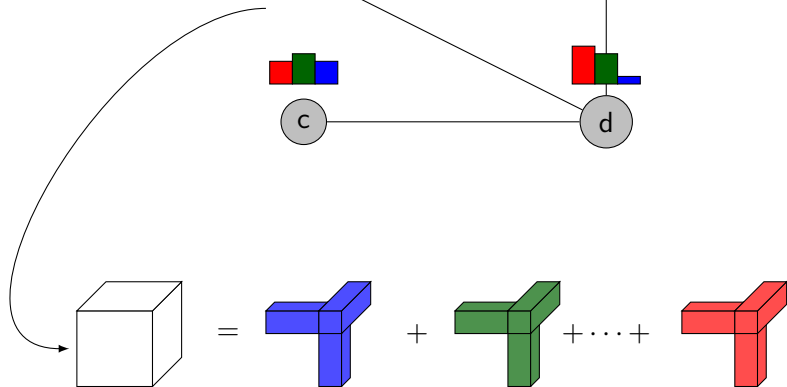
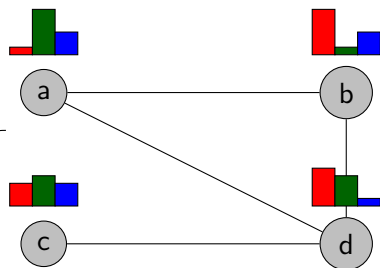
Non-orthogonal tensor factorization

Related work

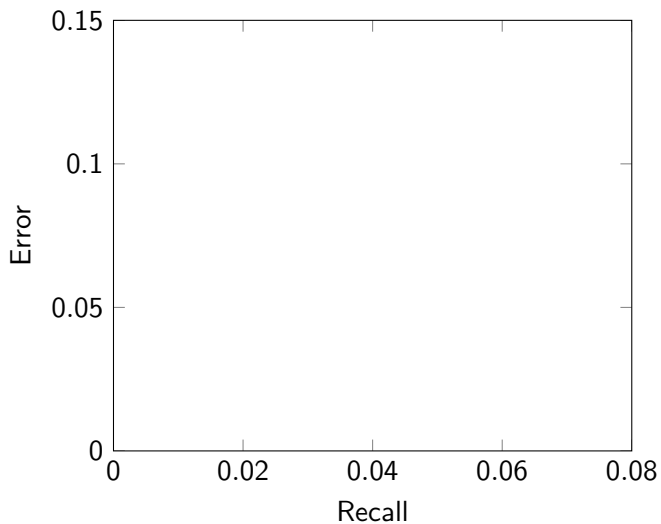
**Empirical results**

Conclusions

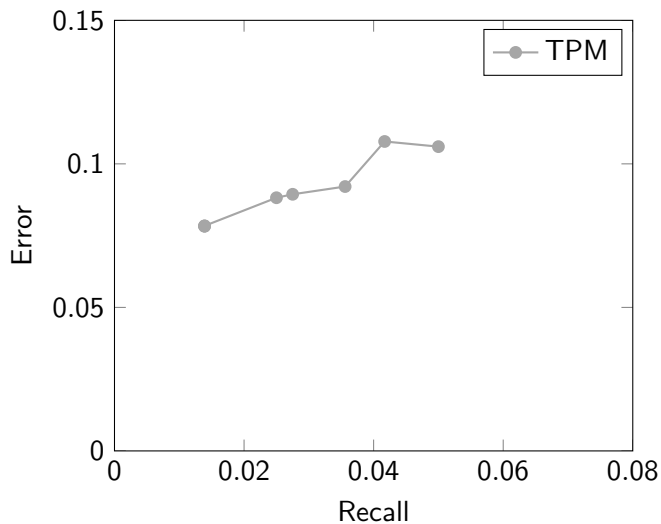
## Community detection

Anandkumar, Ge, Hsu, and  
S. Kakade 2013

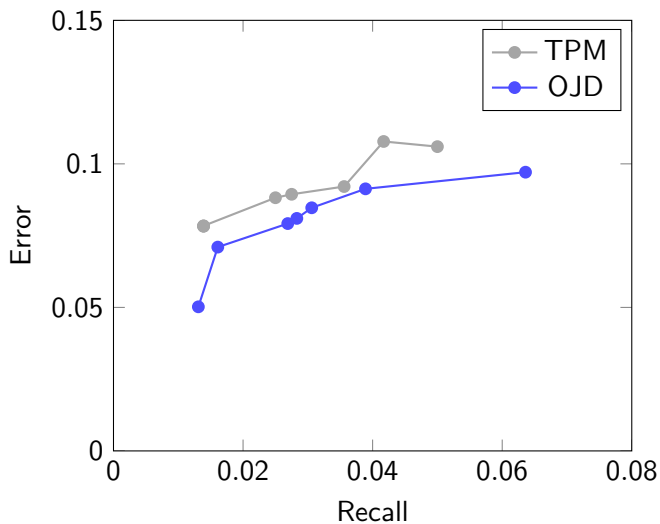
# Community detection



## Community detection

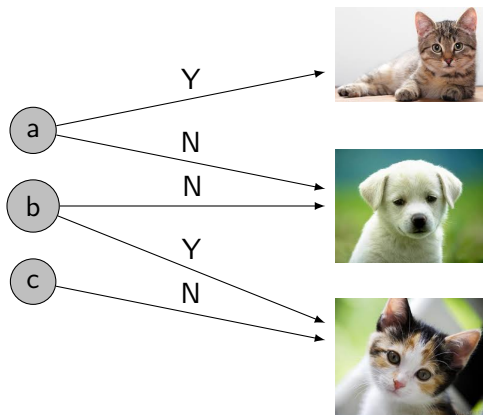


## Community detection



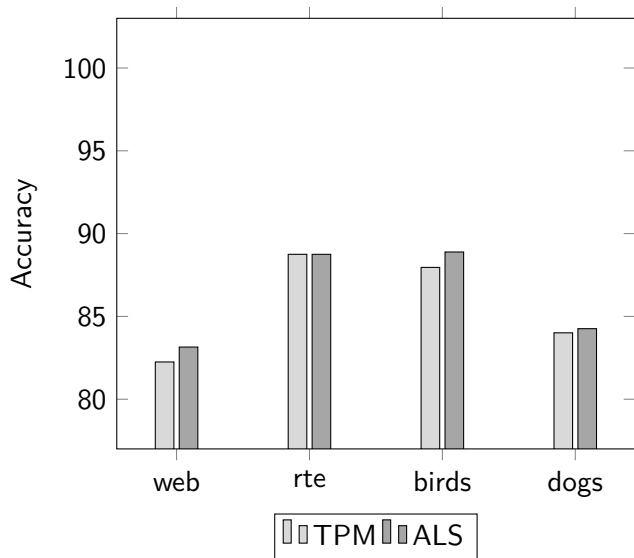
# Crowdsourcing

Zhang et al. 2014



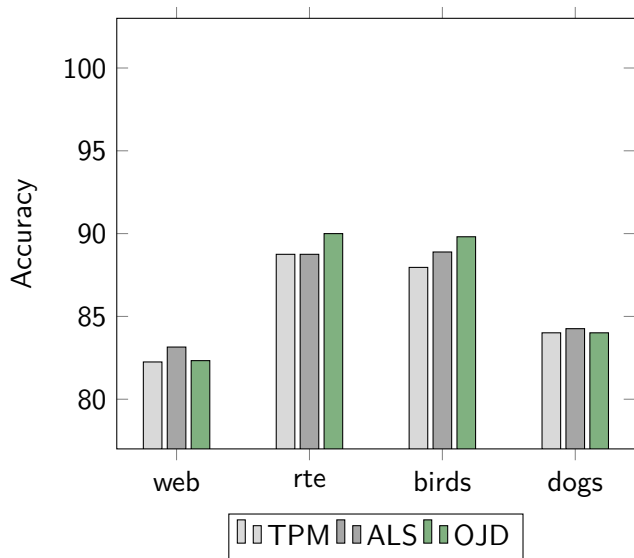
## Crowdsourcing

Zhang et al. 2014



## Crowdsourcing

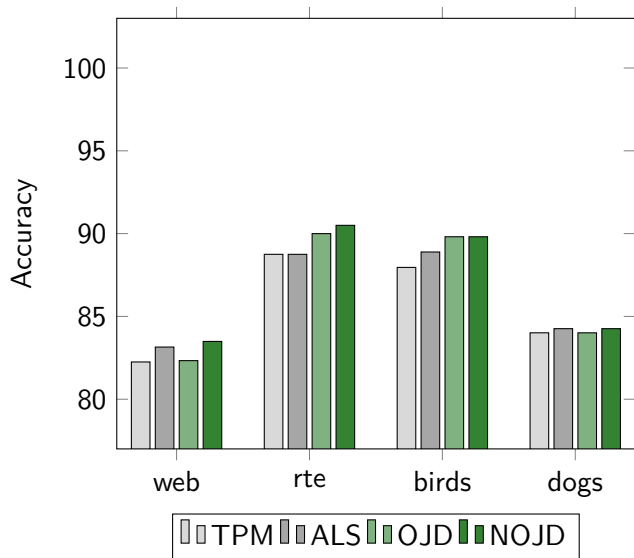
Zhang et al. 2014





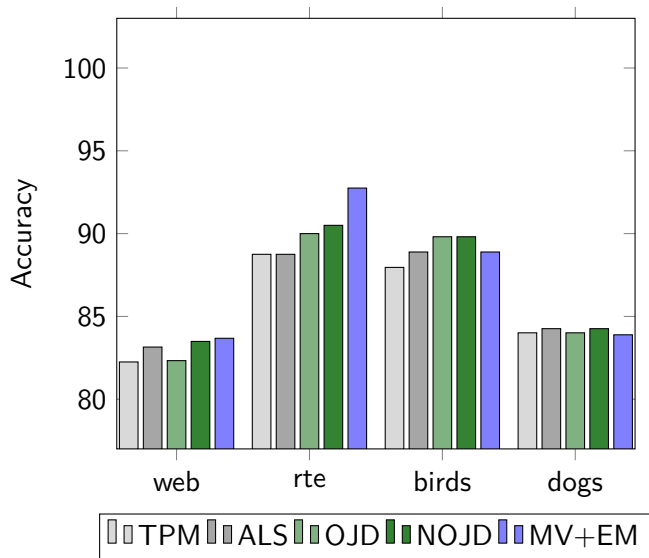
## Crowdsourcing

Zhang et al. 2014

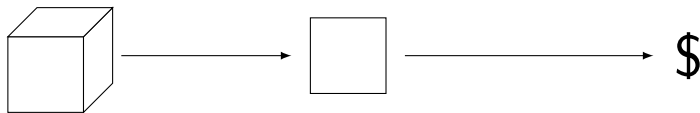


## Crowdsourcing

Zhang et al. 2014

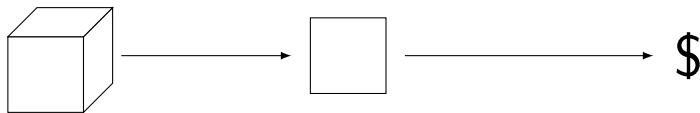


# Conclusions



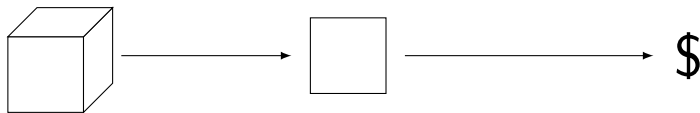
- ▶ **Reduce tensor problems to matrix ones** with random projections.

# Conclusions



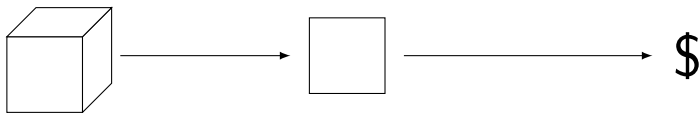
- ▶ **Reduce tensor problems to matrix ones** with random projections.
- ▶ Empirically, **competitive** with state of the art with support for **non-orthogonal, asymmetric tensors of arbitrary order**.

# Conclusions



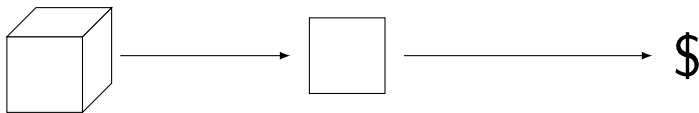
- ▶ **Reduce tensor problems to matrix ones** with random projections.
- ▶ Empirically, **competitive** with state of the art with support for **non-orthogonal, asymmetric tensors of arbitrary order**.
- ▶ **Open question:** is the Jacobi angles algorithm for orthogonal simultaneous diagonalization generically globally convergent?

# Conclusions



- ▶ **Reduce tensor problems to matrix ones** with random projections.
- ▶ Empirically, **competitive** with state of the art with support for **non-orthogonal, asymmetric tensors of arbitrary order**.
- ▶ **Open question:** is the Jacobi angles algorithm for orthogonal simultaneous diagonalization generically globally convergent?
- ▶ **Github:** <https://github.com/kuleshov/tensor-factorization>
- ▶ **Codalab:** <https://www.codalab.org/worksheets/>

# Conclusions



- ▶ **Reduce tensor problems to matrix ones** with random projections.
- ▶ Empirically, **competitive** with state of the art with support for **non-orthogonal, asymmetric tensors of arbitrary order**.
- ▶ **Open question:** is the Jacobi angles algorithm for orthogonal simultaneous diagonalization generically globally convergent?
- ▶ **Github:** <https://github.com/kuleshov/tensor-factorization>
- ▶ **Codalab:** <https://www.codalab.org/worksheets/>
- ▶ **Thanks! Questions?**