Non-negative Tensor Decompositions for Unsupervised Learning

2023 SIAM PNW Conference

Western Washington University, Bellingham, WA

Nicholas Richardson

October 14, 2023

THE UNIVERSITY OF BRITISH COLUMBIA

Table of Contents

Overview

- Overview
- Models
- Solving a model
- Special Considerations
- Conclusion

The Setting
Overview

- Do you have a lot of data?

The Setting

Overview

- Do you have a lot of data?
- A function that depends on many independent variables

The Setting

Overview

- Do you have a lot of data?
- A function that depends on many independent variables
- Many n-dimensional samples

The Setting

Overview

- Do you have a lot of data?
- A function that depends on many independent variables
- Many n-dimensional samples
- Want to compress the data

The Setting

Overview

- Do you have a lot of data?
- A function that depends on many independent variables
- Many n-dimensional samples
- Want to compress the data
- Or find relationships between samples/variables

The Setting

Overview

- Do you have a lot of data?
- A function that depends on many independent variables
- Many n-dimensional samples
- Want to compress the data
- Or find relationships between samples/variables

Tensor Decompositions are for you!

What is a Tension Decomposition?

Overview

1. Take your data array

What is a Tensor Decomposition?

Overview

1. Take your data array
2. Select a model

What is a Tensor Decomposition?

Overview

1. Take your data array
2. Select a model
3. Solve a block-convex minimization problem between your data and a sum-product of smaller sized arrays using block coordinate gradient decent but its not converging to reasonable values because you forgot to set a unique scaling for your factors so you settle for a sub-optimal solution by projecting onto a reasonable set

What is a Tensor Decomposition?

Overview

1. Take your data array
2. Select a model
3. Write it as a product of smaller arrays

Benefits

Overview

- Less storage than original data

Benefits

Overview

- Less storage than original data
- Interpretable results

Benefits

Overview

- Less storage than original data
- Interpretable results
- Easy to implement optimization problems often converge to Nash Point

Benefits

Overview

- Less storage than original data
- Interpretable results
- Easy to implement optimization problems often converge to Nash Point
  - Minima $\implies$ critical point $\implies$ (local) Nash point

Benefits

Overview

- Less storage than original data
- Interpretable results
- Easy to implement optimization problems often converge to Nash Point
  - Minima $\implies$ critical point $\implies$ (local) Nash point
  - Some models have: critical point $\iff$ (global) Nash point

Benefits

Overview

- Less storage than original data
- Interpretable results
- Easy to implement optimization problems often converge to Nash Point
  - Minima $\implies$ critical point $\implies$ (local) Nash point
  - Some models have: critical point $\iff$ (global) Nash point
- No "training" required

Benefits

Overview

- Less storage than original data
- Interpretable results
- Easy to implement optimization problems often converge to Nash Point
  - Minima $\implies$ critical point $\implies$ (local) Nash point
  - Some models have: critical point $\iff$ (global) Nash point
- No “training” required
- none or only a few hyperparameters to tune

Table of Contents

Models

▶ Overview

▶ Models

▶ Solving a model

▶ Special Considerations

▶ Conclusion

The Data Tensor
Models

- Order-N tensor

$$\mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$$

The Data Tensor
Models

- Order-N tensor

$$Y \in \mathbb{R}^{I_1 \times \cdots \times I_N}$$

- Often consider additional constraints

The Data Tensor

Models

- Order-N tensor

$$Y \in \mathbb{R}^{I_1 \times \cdots \times I_N}$$

- Often consider additional constraints
  - non-negative: $Y_{i_1 \dots i_N} \in \mathbb{R}_+$

The Data Tensor

Models

- Order-N tensor

$$Y \in \mathbb{R}^{I_1 \times \cdots \times I_N}$$

- Often consider additional constraints
  - non-negative: $Y_{i_1 \dots i_N} \in \mathbb{R}_+$
  - probability values: $0 \leq Y_{i_1 \dots i_N} \leq 1$

The Data Tensor

Models

- Order-N tensor

$$Y \in \mathbb{R}^{I_1 \times \cdots \times I_N}$$

- Often consider additional constraints
  - non-negative: $Y_{i_1 \dots i_N} \in \mathbb{R}_+$
  - probability values: $0 \leq Y_{i_1 \dots i_N} \leq 1$
  - distributions: $1 = \sum_j Y_{i_1 \dots j \dots i_N}$

The Data Tensor
Models

- Order-N tensor

$$Y \in \mathbb{R}^{I_1 \times \cdots \times I_N}$$

- Often consider additional constraints
  - non-negative: $Y_{i_1 \ldots i_N} \in \mathbb{R}_+$
  - probability values: $0 \leq Y_{i_1 \ldots i_N} \leq 1$
  - distributions: $1 = \sum_j Y_{i_1 \ldots j \ldots i_N}$
  - binary: $Y_{i_1 \ldots i_N} \in \{0, 1\}$

The Data Tensor

Models

- Order-N tensor

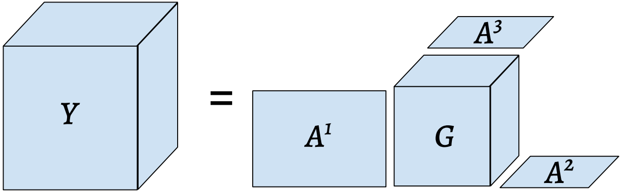
$$Y \in \mathbb{R}^{I_1 \times \cdots \times I_N}$$

- Often consider additional constraints
  - non-negative: $Y_{i_1 \ldots i_N} \in \mathbb{R}_+$
  - probability values: $0 \leq Y_{i_1 \ldots i_N} \leq 1$
  - distributions: $1 = \sum_j Y_{i_1 \ldots j \ldots i_N}$
  - binary: $Y_{i_1 \ldots i_N} \in \{0, 1\}$
- If Y is too big to store, calculate $Y_{i_1 \ldots i_N}$ on-the-fly

Tucker Decomposition

Models

- Factorize Y into a core tensor $G \in \mathbb{R}^{R_1 \times \cdots \times R_N}$ and matrices $A^n \in \mathbb{R}^{I_n \times R_n}$



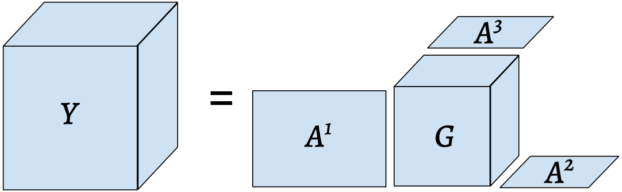
Tucker Decomposition

Models

- Factorize Y into a core tensor $G \in \mathbb{R}^{R_1 \times \cdots \times R_N}$ and matrices $A^n \in \mathbb{R}^{I_n \times R_n}$

$$Y = G \times_1 A^1 \times_2 \cdots \times_N A^N$$

$$Y_{i_1 \dots i_N} = \sum_{r_1 \dots r_N} G_{r_1 \dots r_N} A^1_{i_1 r_1} \cdots A^N_{i_N r_N}$$



Tucker Decomposition

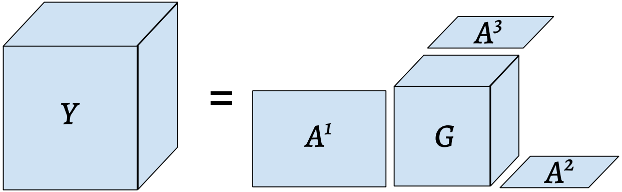
Models

- Factorize Y into a core tensor $G \in \mathbb{R}^{R_1 \times \cdots \times R_N}$ and matrices $A^n \in \mathbb{R}^{I_n \times R_n}$

$$Y = G \times_1 A^1 \times_2 \cdots \times_N A^N$$

$$Y_{i_1 \ldots i_N} = \sum_{r_1 \ldots r_N} G_{r_1 \ldots r_N} A^1_{i_1 r_1} \cdots A^N_{i_N r_N}$$

- Call Y a rank-$(R_1, \ldots, R_N)$ tensor, and R_n the n-rank where $R_n = \mathrm{rank}_n(Y)$



Special Cases

Models

- Canonical polyadic decomposition (CP): $G = I$ and $R_1, \ldots, R_N = R$

Special Cases

Models

- Canonical polyadic decomposition (CP): $G = I$ and $R_1, \ldots, R_N = R$
- Smallest R is the tensor rank

$$Y = I \times_1 A^1 \times_2 \cdots \times_N A^N$$

$$Y_{i_1 \ldots i_N} = \sum_r A^1_{i_1 r} \cdots A^N_{i_N r}$$

e.g. low-rank decomp.: $\quad Y = AB^\top$

Special Cases

Models

- Canonical polyadic decomposition (CP): $G = I$ and $R_1, \ldots, R_N = R$
- Smallest R is the tensor rank

$$Y = I \times_1 A^1 \times_2 \cdots \times_N A^N$$

$$Y_{i_1 \ldots i_N} = \sum_r A^1_{i_1 r} \cdots A^N_{i_N r}$$

$$\text{e.g. low-rank decomp.:} \quad Y = AB^\top$$

- Tucker-n: $A^{n+1}, \ldots, A^N = I$ (possibly different sizes!)

$$Y = G \times_1 A^1 \times_2 \cdots \times_n A^n$$

$$\text{e.g. Tucker-1:} \quad Y = G \times_1 A$$

Extensions & Additional Compression

Models

- Factorize the core symmetrically (Extend Tucker)

$$G_{r_1 r_2 r_3} = \sum_{i,j,k=1}^{R} B^1_{r_1 jk} B^2_{i r_2 k} B^3_{ij r_3}$$

Extensions & Additional Compression

Models

- Factorize the core symmetrically (Extend Tucker)

$$G_{r_1 r_2 r_3} = \sum_{i,j,k=1}^{R} B^1_{r_1 jk} B^2_{i r_2 k} B^3_{ij r_3}$$

- Tensor Trains (Extend Tucker-2)

$$Y_{i_1 \ldots i_N} = \sum_{r_1, r_N} A^1_{i_1 r_1} \underbrace{\sum_{j_2 \ldots j_{N-2}} A^2_{r_1 i_2 j_2} A^3_{j_2 i_3 j_3} \ldots A^{n-1}_{j_{N-2} i_{N-1} r_2}}_{G_{r_1 i_2 \ldots i_{N-1} r_N}} A^N_{i_N r_N}$$

Extensions & Additional Compression

Models

- Factorize the core symmetrically (Extend Tucker)

$$G_{r_1 r_2 r_3} = \sum_{i,j,k=1}^{R} B^1_{r_1 jk} B^2_{i r_2 k} B^3_{ij r_3}$$

- Tensor Trains (Extend Tucker-2)

$$Y_{i_1 \ldots i_N} = \sum_{r_1, r_N} A^1_{i_1 r_1} \underbrace{\sum_{j_2 \ldots j_{N-2}} A^2_{r_1 i_2 j_2} A^3_{j_2 i_3 j_3} \ldots A^{n-1}_{j_{N-2} i_{N-1} r_2}}_{G_{r_1 i_2 \ldots i_{N-1} r_N}} A^N_{i_N r_N}$$

- Factorize the matrices (Extend CP)

$$A^n_{i_n r} = \sum_k T_{i_n rk} b_k$$

Table of Contents

Solving a model

- Overview
- Models
- Solving a model
- Special Considerations
- Conclusion

The Optimization Problem

Solving a model

- May try to solve the model exactly

$$Y = f(A^1, \ldots, A^N) \quad \text{(e.g. NMF: } Y = WH\text{)}$$

The Optimization Problem

Solving a model

- May try to solve the model exactly

$$Y = f(A^1, \ldots, A^N) \quad \text{(e.g. NMF: } Y = WH\text{)}$$

- With noise, or imperfect modelling, better to find “closest” fit

$$\min_{A^n \in C^n} D(Y, \hat{Y}) \quad \text{s.t.} \quad \hat{Y} = f(A^1, \ldots, A^N)$$

The Optimization Problem

Solving a model

- May try to solve the model exactly

$$Y = f(A^1, \ldots, A^N) \quad \text{(e.g. NMF: } Y = WH\text{)}$$

- With noise, or imperfect modelling, better to find "closest" fit

$$\min_{A^n \in C^n} D(Y, \hat{Y}) \quad \text{s.t.} \quad \hat{Y} = f(A^1, \ldots, A^N)$$

- Call the objective $D(Y, \hat{Y}) = F(A^1, \ldots, A^N)$

The Optimization Problem

Solving a model

- May try to solve the model exactly

$$Y = f(A^1, \dots, A^N) \quad \text{(e.g. NMF: } Y = WH\text{)}$$

- With noise, or imperfect modelling, better to find "closest" fit

$$\min_{A^n \in C^n} D(Y, \hat{Y}) \quad \text{s.t.} \quad \hat{Y} = f(A^1, \dots, A^N)$$

- Call the objective $D(Y, \hat{Y}) = F(A^1, \dots, A^N)$
  - $\|Y - \hat{Y}\|_F^2$

The Optimization Problem

Solving a model

- May try to solve the model exactly

$$Y = f(A^1, \ldots, A^N) \quad \text{(e.g. NMF: } Y = WH\text{)}$$

- With noise, or imperfect modelling, better to find "closest" fit

$$\min_{A^n \in C^n} D(Y, \hat{Y}) \quad \text{s.t.} \quad \hat{Y} = f(A^1, \ldots, A^N)$$

- Call the objective $D(Y, \hat{Y}) = F(A^1, \ldots, A^N)$
  - $\|Y - \hat{Y}\|_F^2$
  - $\displaystyle\sum_{i_1 \ldots i_N} Y_{i_1 \ldots i_N} \log \left(\frac{Y_{i_1 \ldots i_N}}{\hat{Y}_{i_1 \ldots i_N}} \right)$

Full Gradient Descent, Alternating Least-Squares, and Problems

Solving a model

- (Projected) Full Gradient Descent:

$$\mathcal{A} \leftarrow \arg\min_{\mathcal{B} \in C} \langle \nabla F(\mathcal{A}), \mathcal{B} - \mathcal{A} \rangle + \frac{1}{2\alpha} \|\mathcal{B} - \mathcal{A}\|_F^2$$

$$(A^1, \ldots, A^N) \leftarrow P_C\left((A^1, \ldots, A^N) - \alpha \nabla F(A^1, \ldots, A^N)\right)$$

Full Gradient Descent, Alternating Least-Squares, and Problems

Solving a model

- (Projected) Full Gradient Descent:

$$\mathcal{A} \leftarrow \arg\min_{\mathcal{B}\in C} \langle \nabla F(\mathcal{A}), \mathcal{B} - \mathcal{A} \rangle + \frac{1}{2\alpha} \|\mathcal{B} - \mathcal{A}\|_F^2$$

$$(A^1, \ldots, A^N) \leftarrow P_C\left((A^1, \ldots, A^N) - \alpha \nabla F(A^1, \ldots, A^N)\right)$$

  - Non-expensive updates, but F not convex, not linear

Full Gradient Descent, Alternating Least-Squares, and Problems

Solving a model

- (Projected) Full Gradient Descent:

$$\mathcal{A} \leftarrow \arg\min_{\mathcal{B}\in C} \langle \nabla F(\mathcal{A}), \mathcal{B} - \mathcal{A} \rangle + \frac{1}{2\alpha} \|\mathcal{B} - \mathcal{A}\|_F^2$$

$$(A^1, \ldots, A^N) \leftarrow P_C\left((A^1, \ldots, A^N) - \alpha \nabla F(A^1, \ldots, A^N)\right)$$

  - Non-expensive updates, but F not convex, not linear
- Alternating Least-Squares:

$$A^n \leftarrow \arg\min_{A\in C} \|Y - f(A^1, \ldots, A^{n-1}, A, A^{n+1}, \ldots, A^N)\|_F^2$$

Full Gradient Descent, Alternating Least-Squares, and Problems

Solving a model

- (Projected) Full Gradient Descent:

$$\mathcal{A} \leftarrow \arg\min_{\mathcal{B}\in\mathcal{C}} \langle \nabla F(\mathcal{A}), \mathcal{B} - \mathcal{A} \rangle + \frac{1}{2\alpha}\|\mathcal{B} - \mathcal{A}\|_F^2$$

$$(A^1, \ldots, A^N) \leftarrow P_{\mathcal{C}}\left((A^1, \ldots, A^N) - \alpha \nabla F(A^1, \ldots, A^N)\right)$$

  - Non-expensive updates, but F not convex, not linear
- Alternating Least-Squares:

$$A^n \leftarrow \arg\min_{A\in\mathcal{C}} \|Y - f(A^1, \ldots, A^{n-1}, A, A^{n+1}, \ldots, A^N)\|_F^2$$

  - f linear in A^n, but waste time fully optimizing A^n every step

Full Gradient Descent, Alternating Least-Squares, and Problems

Solving a model

- (Projected) Full Gradient Descent:

$$\mathcal{A} \leftarrow \arg\min_{\mathcal{B}\in C} \langle \nabla F(\mathcal{A}), \mathcal{B} - \mathcal{A} \rangle + \frac{1}{2\alpha} \|\mathcal{B} - \mathcal{A}\|_F^2$$

$$(A^1, \ldots, A^N) \leftarrow P_C\left((A^1, \ldots, A^N) - \alpha \nabla F(A^1, \ldots, A^N)\right)$$

  - Non-expensive updates, but F not convex, not linear
- Alternating Least-Squares:

$$A^n \leftarrow \arg\min_{A\in C} \|Y - f(A^1, \ldots, A^{n-1}, A, A^{n+1}, \ldots, A^N)\|_F^2$$

  - f linear in A^n, but waste time fully optimizing A^n every step
  - may not converge to stationary point

Block Coordinate Descent (BCD)

Solving a model

- Combine into BCD:

$$A^n \leftarrow \arg\min_{A \in C} \langle \nabla_{A^n} F, A - A^n \rangle + \frac{1}{2\alpha} \|A - A^n\|_F^2$$

$$A^n \leftarrow P_C \left(A^n - \alpha \nabla_{A^n} F \right)$$

Block Coordinate Descent (BCD)

Solving a model

- Combine into BCD:

$$A^n \leftarrow \arg\min_{A \in C} \langle \nabla_{A^n} F, A - A^n \rangle + \frac{1}{2\alpha} \|A - A^n\|_F^2$$

$$A^n \leftarrow P_C \left(A^n - \alpha \nabla_{A^n} F \right)$$

- Alternately update the factors, but don’t fully optimize each factors

Block Coordinate Descent (BCD)

Solving a model

- Combine into BCD:

$$A^n \leftarrow \arg\min_{A \in C} \langle \nabla_{A^n} F, A - A^n \rangle + \frac{1}{2\alpha} \|A - A^n\|_F^2$$

$$A^n \leftarrow P_C \left(A^n - \alpha \nabla_{A^n} F \right)$$

- Alternately update the factors, but don't fully optimize each factors
- Can choose $\alpha = 1/L_n$

Block Coordinate Descent (BCD)

Solving a model

- Combine into BCD:

$$A^n \leftarrow \arg\min_{A \in C} \langle \nabla_{A^n} F, A - A^n \rangle + \frac{1}{2\alpha} \|A - A^n\|_F^2$$

$$A^n \leftarrow P_C \left(A^n - \alpha \nabla_{A^n} F \right)$$

- Alternately update the factors, but don't fully optimize each factors
- Can choose $\alpha = 1/L_n$
  - L is the Lipshitz constant of $F(A^1, \ldots, A^{n-1}, \cdot, A^{n+1}, \ldots, A^N)$

Block Coordinate Descent (BCD)

Solving a model

- Combine into BCD:

$$A^n \leftarrow \arg\min_{A \in C} \langle \nabla_{A^n} F, A - A^n \rangle + \frac{1}{2\alpha} \|A - A^n\|_F^2$$

$$A^n \leftarrow P_C \left(A^n - \alpha \nabla_{A^n} F \right)$$

- Alternately update the factors, but don't fully optimize each factors
- Can choose $\alpha = 1/L_n$
  - L is the Lipshitz constant of $F(A^1, \ldots, A^{n-1}, \cdot, A^{n+1}, \ldots, A^N)$

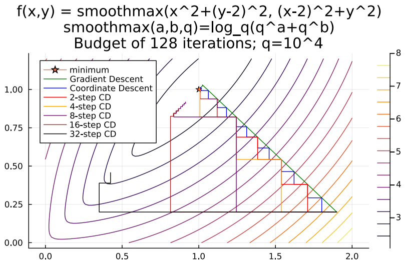


Figure: Full vs Coordinate Descent

Multiplicative Updates

Solving a model

- Alternate to BCD with non-negative constraint:

$$A^n_{i_1 \dots i_n} \leftarrow A^n_{i_1 \dots i_n} \beta^n_{i_1 \dots i_n} \quad \beta \geq 0$$

Multiplicative Updates

Solving a model

- Alternate to BCD with non-negative constraint:

$$A^n_{i_1 \ldots i_n} \leftarrow A^n_{i_1 \ldots i_n} \beta^n_{i_1 \ldots i_n} \quad \beta \geq 0$$

- Ex: $F(A, B) = \frac{1}{2}\|Y - AB\|_F^2$ use

$$\beta^1 = \frac{YB^\top}{ABB^\top} \quad \& \quad \beta^2 = \frac{A^\top Y}{A^\top AB}$$

Multiplicative Updates

Solving a model

- Alternate to BCD with non-negative constraint:

$$A^n_{i_1 \dots i_n} \leftarrow A^n_{i_1 \dots i_n} \beta^n_{i_1 \dots i_n} \quad \beta \geq 0$$

- Ex: $F(A, B) = \frac{1}{2}\|Y - AB\|_F^2$ use

$$\beta^1 = \frac{YB^\top}{ABB^\top} \quad \& \quad \beta^2 = \frac{A^\top Y}{A^\top AB}$$

- Ensures entries stay positive

Multiplicative Updates

Solving a model

- Alternate to BCD with non-negative constraint:

$$A^n_{i_1\ldots i_n} \leftarrow A^n_{i_1\ldots i_n}\beta^n_{i_1\ldots i_n} \quad \beta \geq 0$$

- Ex: $F(A, B) = \frac{1}{2}\|Y - AB\|_F^2$ use

$$\beta^1 = \frac{YB^\top}{ABB^\top} \quad \& \quad \beta^2 = \frac{A^\top Y}{A^\top AB}$$

- Ensures entries stay positive
- Equivalent to BCD with the stepsizes $\alpha^1_{ij} = \frac{A_{ij}}{(ABB^\top)_{ij}}$ and $\alpha^2_{ij} = \frac{B_{ij}}{(A^\top AB)_{ij}}$

Bells and Whistles

Solving a model

- Momentum:

$$\hat{A}^{(k)} = A^{(k)} + \mu^{(k)} \left(A^{(k)} - A^{(k-1)} \right)$$

Bells and Whistles

Solving a model

- Momentum:

$$\hat{A}^{(k)} = A^{(k)} + \mu^{(k)} \left(A^{(k)} - A^{(k-1)} \right)$$

- Regularizes:

$$\|A^n\|_1,\ \|A^n\|_2^F,\ \text{etc.}$$

Bells and Whistles

Solving a model

- Momentum:

$$\hat{A}^{(k)} = A^{(k)} + \mu^{(k)} \left(A^{(k)} - A^{(k-1)} \right)$$

- Regularizes:

$$\|A^n\|_1, \ \|A^n\|_2^F, \text{ etc.}$$

- Line-search Stepsize:

$$\min_{\alpha > 0} F(A^1, \ldots, A^{n-1}, \underbrace{A^n - \alpha \nabla_{A^n} F}_{\text{line search}}, A^{n+1}, \ldots, A^N)$$

Table of Contents

Special Considerations

- ▶ Overview
- ▶ Models
- ▶ Solving a model
- ▶ Special Considerations
- ▶ Conclusion

- BCD converges to (global) Nash point $(A^1, \dots, A^N)$:

$$A^n = \arg\min_{A \in C^n} F(A^1, \dots, A^{n-1}, A, A^{n+1}, \dots, A^N), \quad n = 1, \dots, N$$

Convergence

Special Considerations

- BCD converges to (global) Nash point $(A^1, \dots, A^N)$:

$$A^n = \arg\min_{A \in C^n} F(A^1, \dots, A^{n-1}, A, A^{n+1}, \dots, A^N), \quad n = 1, \dots, N$$

- "Cannot improve objective by updating one block"

Convergence

Special Considerations

- BCD converges to (global) Nash point $(A^1, \ldots, A^N)$:

$$A^n = \arg\min_{A \in C^n} F(A^1, \ldots, A^{n-1}, A, A^{n+1}, \ldots, A^N), \quad n = 1, \ldots, N$$

- "Cannot improve objective by updating one block"
- For block convex functions and separable constraints:

Convergence

Special Considerations

- BCD converges to (global) Nash point $(A^1, \ldots, A^N)$:

$$A^n = \arg\min_{A \in C^n} F(A^1, \ldots, A^{n-1}, A, A^{n+1}, \ldots, A^N), \quad n = 1, \ldots, N$$

- "Cannot improve objective by updating one block"
- For block convex functions and separable constraints:

Nash point condition

Let $F(\cdot, B), F(A, \cdot)$ be convex, and $(A_0, B_0) \in C = C_A \times C_B$ be point. Then,

$$\mathbf{0} \in \partial(F + \delta_C)(A_0, B_0) \iff \begin{array}{l} F(A_0, B_0) \leq F(A, B_0) \ \forall A \in C_A \\ F(A_0, B_0) \leq F(A_0, B) \ \forall B \in C_B \end{array}.$$

Selecting the Rank(s) R_n

Special Considerations

- Option 1: Use information about your physical system

Selecting the Rank(s) R_n

Special Considerations

- Option 1: Use information about your physical system
  - e.g. decompose piano audio into notes and amplitudes

Selecting the Rank(s) R_n

Special Considerations

- Option 1: Use information about your physical system
  - e.g. decompose piano audio into notes and amplitudes
  - $R = 88$ since there are 88 keys

Selecting the Rank(s) R_n

Special Considerations

- Option 1: Use information about your physical system
  - e.g. decompose piano audio into notes and amplitudes
  - $R = 88$ since there are 88 keys
- Option 2: When there's only 1 unknown rank;

Selecting the Rank(s) R_n

Special Considerations

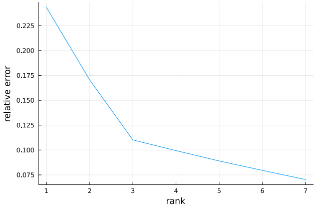
- Option 1: Use information about your physical system
  - e.g. decompose piano audio into notes and amplitudes
  - $R = 88$ since there are 88 keys
- Option 2: When there's only 1 unknown rank;
  - Solve the model for all ranks $R = 1, \ldots, I$

$$R = \arg\max_{r} \kappa(r) := \frac{F''(r)}{(1 + F'(r)^2)^{1.5}}$$

Selecting the Rank(s) R_n

Special Considerations

- Option 1: Use information about your physical system
  - e.g. decompose piano audio into notes and amplitudes
  - $R = 88$ since there are 88 keys
- Option 2: When there's only 1 unknown rank;
  - Solve the model for all ranks $R = 1, \ldots, I$
  - Compute final objective value $F(R)$

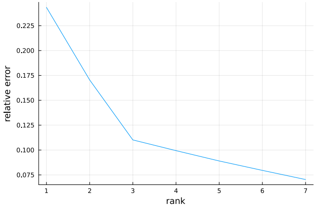


$$R = \arg\max_{r} \kappa(r) := \frac{F''(r)}{(1 + F'(r)^2)^{1.5}}$$

Selecting the Rank(s) R_n

Special Considerations

- Option 1: Use information about your physical system
  - e.g. decompose piano audio into notes and amplitudes
  - $R = 88$ since there are 88 keys
- Option 2: When there's only 1 unknown rank;
  - Solve the model for all ranks $R = 1, \ldots, I$
  - Compute final objective value $F(R)$
  - Select point of maximum curvature



$$R = \arg\max_{r} \kappa(r) := \frac{F''(r)}{(1 + F'(r)^2)^{1.5}}$$

Uniqueness & Scaling

Special Considerations

- These models are not (usually) unique

Uniqueness & Scaling

Special Considerations

- These models are not (usually) unique
- Ex. $Y = AB = (AC)(C^{-1}B)$ for invertable C

Uniqueness & Scaling

Special Considerations

- These models are not (usually) unique
- Ex. $Y = AB = (AC)(C^{-1}B)$ for invertable C
- Fix a scaling on factors: set $\|A^n\| = c_n$ or $\sum_j A_{\dots j \dots} = c_n$ for...

Uniqueness & Scaling

Special Considerations

- These models are not (usually) unique
- Ex. $Y = AB = (AC)(C^{-1}B)$ for invertable C
- Fix a scaling on factors: set $\|A^n\| = c_n$ or $\sum_j A_{\dots j \dots} = c_n$ for…

$\dots$ all but one factor

$$\text{e.g.: } Y_{ij} = \sum_r A_{ir} B_{rj} \text{ s.t. } \|B_{r:}\|_2 = 1$$

Uniqueness & Scaling

Special Considerations

- These models are not (usually) unique
- Ex. $Y = AB = (AC)(C^{-1}B)$ for invertable C
- Fix a scaling on factors: set $\|A^n\| = c_n$ or $\sum_j A_{\ldots j \ldots} = c_n$ for…

… all but one factor

$$\text{e.g.: } Y_{ij} = \sum_r A_{ir} B_{rj} \text{ s.t. } \|B_{r:}\|_2 = 1$$

… all factors & add scaling parameter λ

$$\text{e.g.: } Y_{ij} = \sum_r \lambda_r A_{ir} B_{rj} \text{ s.t. } \|A_{:r}\|_2, \|B_{r:}\|_2 = 1$$

Uniqueness & Scaling
Special Considerations

- These models are not (usually) unique
- Ex. $Y = AB = (AC)(C^{-1}B)$ for invertable C
- Fix a scaling on factors: set $\|A^n\| = c_n$ or $\sum_j A_{\ldots j \ldots} = c_n$ for...

$\ldots$ all but one factor

$$\text{e.g.: } Y_{ij} = \sum_r A_{ir} B_{rj} \text{ s.t. } \|B_{r:}\|_2 = 1$$

$\ldots$ all factors & add scaling parameter λ

$$\text{e.g.: } Y_{ij} = \sum_r \lambda_r A_{ir} B_{rj} \text{ s.t. } \|A_{:r}\|_2, \|B_{r:}\|_2 = 1$$

- Enforce through a constraint (projection) or rescale at the end/each iteration

Uniqueness & Scaling

Special Considerations

- These models are not (usually) unique
- Ex. $Y = AB = (AC)(C^{-1}B)$ for invertable C
- Fix a scaling on factors: set $\|A^n\| = c_n$ or $\sum_j A_{\dots j \dots} = c_n$ for…

… all but one factor

$$\text{e.g.: } Y_{ij} = \sum_r A_{ir} B_{rj} \text{ s.t. } \|B_{r:}\|_2 = 1$$

… all factors & add scaling parameter λ

$$\text{e.g.: } Y_{ij} = \sum_r \lambda_r A_{ir} B_{rj} \text{ s.t. } \|A_{:r}\|_2, \|B_{r:}\|_2 = 1$$

- Enforce through a constraint (projection) or rescale at the end/each iteration
- Still only unique up to permutations of rows/columns/fibres

Table of Contents

Conclusion

- Overview
- Models
- Solving a model
- Special Considerations
- Conclusion

Summary
Conclusion

- Looked at various tensor decomposition models

Summary

Conclusion

- Looked at various tensor decomposition models
- Optimization methods to solve them

Summary

Conclusion

- Looked at various tensor decomposition models
- Optimization methods to solve them
- Practical considerations

References

Conclusion

- W. Austin, T. G. Kolda, and T. Plantenga, “Tensor Rank Prediction via Cross Validation,” Aug. 2014.
- T. G. Kolda and B. W. Bader, “Tensor Decompositions and Applications,” *SIAM Review*, vol. 51, pp. 455–500, Aug. 2009.
- D. Lee and H. S. Seung, “Algorithms for Non-negative Matrix Factorization,” in *Advances in Neural Information Processing Systems*, vol. 13, MIT Press, 2000.
- L. Qi, Y. Chen, M. Bakshi, and X. Zhang, “Triple Decomposition and Tensor Recovery of Third Order Tensors,” Mar. 2020.
- V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, “Finding a ”Kneedle” in a Haystack: Detecting Knee Points in System Behavior,” in *2011 31st International Conference on Distributed Computing Systems Workshops*, pp. 166–171, June 2011.
- Y. Xu and W. Yin, “A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion,” *SIAM Journal on Imaging Sciences*, vol. 6, pp. 1758–1789, Jan. 2013.

Thank you for listening!
Any questions?