Tensor Factorization
Models, Algorithms and Applications

Shihua Zhang

Fall 2019

Outline

1. Introduction
2. Low-rank Models
3. Algorithms
4. Applications in machine Learning

Tensors?

[Sidiropoulos et al., 2017]

- A tensor is an array whose elements are indexed by $(i, j, k, ...)$- more than two indexes.
  - Purchase data: indexed by 'user', 'item', 'seller', 'time'
  - Movie rating data: indexed by 'user', 'movie', 'time'
  - MIMO radar: indexed by 'angle', 'range', 'Doppler', 'profiles'
  - ...
  - Tensor can also represent an operator.

Tensors?

[Sidiropoulos et al., 2017]

- A tensor is an array whose elements are indexed by $(i, j, k, ...)$- more than two indexes.
  - Purchase data: indexed by 'user', 'item', 'seller', 'time'
  - Movie rating data: indexed by 'user', 'movie', 'time'
  - MIMO radar: indexed by 'angle', 'range', 'Doppler', 'profiles'
  - ...
  - Tensor can also represent an operator.
- A matrix is a 2nd-order tensor, whose elements are indexed by (i, j).
- Three-way tensors are easy to visualize 'shoe boxes'.

Why Are We Interested in Tensors?

Tensor algebra: similar with matrix algebra, but also some differences:

- Determining matrix rank is easy (SVD), but determining higher-order tensor rank is NP-hard.
- Low-rank matrix decomposition is easy (SVD), but tensor decomposition is NP-hard.
- A matrix can have an infinite number of low-rank decompositions, but tensor decomposition is unique (under mild conditions).

Many Applications

- Early developments: Psychometrics and Chemometrics
- Beginning from the 1990s: Signal processing
  - communications, radar
  - speech and audio
  - biomedical
  - ...
- Starting in mid-2000's: Machine learning and data mining
  - clustering
  - dimensionality reduction
  - latent factor models (topic mining & community detection)
  - structured subspace learning
  - ...

Low-rank Matrix Approximation

- In practice, we observe $X = L + N$.
  - $L = AB^T$ is low-rank.
  - N represents noise and 'unmodeled dynamics'
- In many cases we are interested in the L part:

$$\min_{L|\text{rank}(L)=\ell} \|X - L\|_F^2 \Longleftrightarrow \min_{A\in\mathbb{R}^{I\times\ell}, B\in\mathbb{R}^{J\times\ell}} \|X - AB^T\|_F^2$$

Low-rank Matrix Approximation

- In practice, we observe $X = L + N$.
  - $L = AB^T$ is low-rank.
  - N represents noise and 'unmodeled dynamics'
- In many cases we are interested in the L part:

$$\min_{L|\text{rank}(L)=\ell} \|X - L\|_F^2 \Longleftrightarrow \min_{A\in\mathbb{R}^{I\times\ell}, B\in\mathbb{R}^{J\times\ell}} \|X - AB^T\|_F^2$$

- Solution: truncated SVD of X
  - $X = U\Sigma V^T$,

$$L = U(:, 1:\ell)\Sigma(1:\ell, 1:\ell)(V(:, 1:\ell)^T)$$

  - $A = U(:, 1:\ell)\Sigma(1:\ell, 1:\ell), B = V(:, 1:\ell)$.

Low-rank Matrix Approximation

- In practice, we observe $X = L + N$.
  - $L = AB^T$ is low-rank.
  - N represents noise and 'unmodeled dynamics'
- In many cases we are interested in the L part:

$$\min_{L|\text{rank}(L)=\ell} \|X - L\|_F^2 \iff \min_{A\in\mathbb{R}^{I\times\ell}, B\in\mathbb{R}^{J\times\ell}} \|X - AB^T\|_F^2$$

- Solution: truncated SVD of X
  - $X = U\Sigma V^T$,

$$L = U(:, 1:\ell)\Sigma(1:\ell, 1:\ell)(V(:, 1:\ell)^T)$$

  - $A = U(:, 1:\ell)\Sigma(1:\ell, 1:\ell), B = V(:, 1:\ell)$.
- Even without noise, low-rank decomposition of X is non-unique:

$$AB^T = AMM^{-1}B^T = (AM)(BM^{-}1)^T$$

holds for any non-singular M.

Useful Product 1

- Kronecker product of $A_{I\times K}$ and $B_{J\times L}$ is the $IJ \times KL$ matrix

$$A \otimes B = \begin{bmatrix} BA(1,1) & BA(1,2) & \dots & BA(1,K) \\ BA(2,1) & BA(2,2) & \dots & BA(2,K) \\ \vdots & \vdots & \dots & \vdots \\ BA(I,1) & BA(I,2) & \dots & BA(I,K) \end{bmatrix}$$

- Properties:
  - $b \otimes a = \text{vec}(ab^T)$.
  - $\text{vec}(AMB^T) = (B \otimes A)\text{vec}(M)$.

$$\begin{aligned} \text{vec}(AMB^T) &= \text{vec}\Big(\sum_{k=1}^{K}\sum_{l=1}^{L} A(:,k)M(k,l)(B(:,l))^T\Big) \\ &= \sum_{k=1}^{K}\sum_{l=1}^{L} M(k,l)B(:,l) \otimes A(:,k) \\ &= (B \otimes A)\text{vec}(M) \end{aligned}$$

Useful Product 2

- Khatri-Rao (KR) Product: $A \odot B = [a_1 \otimes b_1, \cdots, a_I \otimes b_I]$.
- Define $D = \text{Diag}(d)$ and $d = \text{diag}(D)$. $\text{vec}(ADB^T) = (B \odot A)d$, which is useful when dealing with the following

$$\min_{D=\text{Diag}(d)} \|X - ADB^T\|_F^2 \Longleftrightarrow \min_d \|\text{vec}(X) - (B \odot A)d\|_2^2$$

- Khatri-Rao product $B \odot A$ is a subset of columns from $B \otimes A$.

Useful Product 3

- Tensor (outer) product of $a_{I\times 1}$ and $b_{J\times 1}$ is defined as the $I \times J$ matrix $a \circledcirc b$ with elements

$$(a \circledcirc b)(i,j) = a(i)b(j)$$

- Note that $a \circledcirc b = ab^T$.
- $a \circledcirc b \circledcirc c$ with elements $(a \circledcirc b \circledcirc c)(i,j,k) = a(i)b(j)c(k)$.
- Naturally generalizes to three- and higher-way cases.

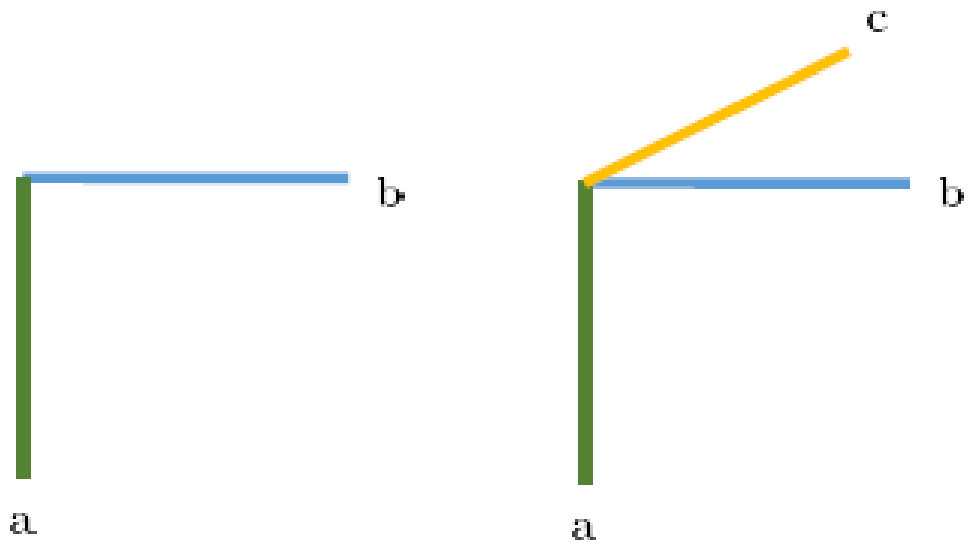
Tensor Rank

- A rank-1 matrix X of size $I \times J$ is an outer product of two vectors: $X(i,j) = a(i)b(j), \forall i \in \{1, \cdots, I\}, j \in \{1, \cdots, J\}$, i.e.,

$$X = a \circledcirc b$$

- A rank-1 third-order tensor X of size $I \times J \times K$ is an outer product of three vectors: $X(i,j,k) = a(i)b(j)c(k)$, i.e.,

$$X = a \circledcirc b \circledcirc c$$



Outline

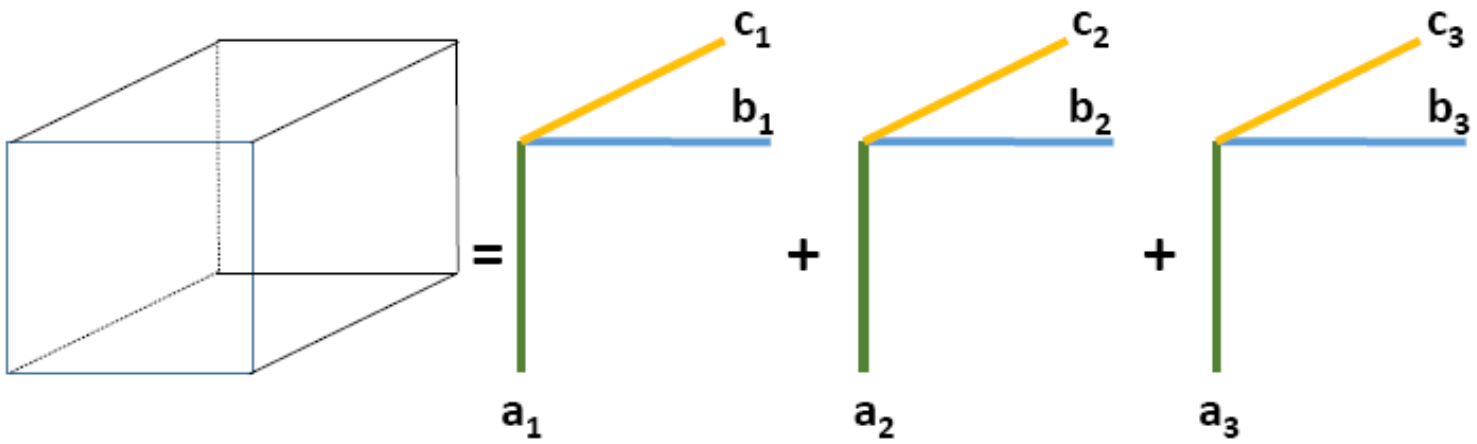
1. Introduction
2. Low-rank Models
   - Canonical Polyadic Decomposition (CPD)
   - Tucker Decomposition
   - Multilinear SVD (MLSVD)
3. Algorithms
4. Applications in machine Learning

Canonical Polyadic Decomposition (CPD)

- CPD or Rank Decomposition: A tensor of rank at most F can be written as

$$X = \sum_{f=1}^{F} a_f \circledcirc b_f \circledcirc c_f \Longleftrightarrow X(i,j,k) = \sum_{f=1}^{F} a_f(i) b_f(j) c_f(k)$$

- Let $A = [a_i, \cdots, a_F]$, $B = [b_1, \cdots, b_F]$, and $C = [c_1, \cdot, c_F] \Rightarrow X(i,j,k) = \sum_{f=1}^{F} A(i,f) B(j,f) C(k,f)$
- For brevity, $X = [A, B, C]$



Slab Representations

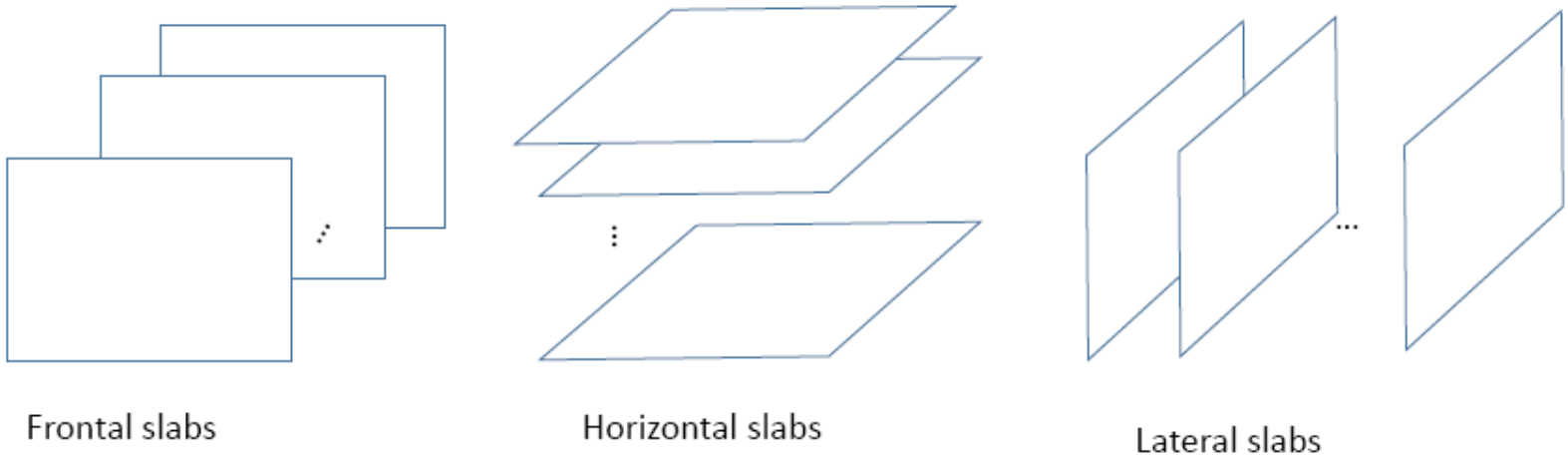


Figure: Slab views of a three-way tensor

Slab Representations-Towards CPD

- Let us look at the frontal slab $X(:,:,1)$ of X:

$$X(i,j,1) = \sum_{f=1}^{F} a_f(i) b_f(j) c_f(1) \Longrightarrow X(:,:,1) = \sum_{f=1}^{F} a_f b_f^T c_f(1) =$$

$$A\,\mathrm{Diag}([c_1(1), c_2(1), \cdots, c_F(1)])B^T = A\,\mathrm{Diag}(C(1,:))B^T$$

- Denote $D_k(C) := \mathrm{Diag}(C(k,:))$ for brevity. Hence, for any k,

$$X(:,:,k) = AD_k(C)B^T, \quad \mathrm{vec}(X(:,:,k)) = (B \odot A)(C(k,:))^T$$

- By parallel stacking, we obtain the matrix unfolding

$$X_3 := [\mathrm{vec}(X(:,:,1)), \mathrm{vec}(X(:,:,2)), \cdots, \mathrm{vec}(X(:,:,K))] \rightarrow$$
$$X_3 = (B \odot A)C^T, \quad (IJ \times K)$$

Slab Representations-Towards CPD

- In the same way, we may consider lateral slabs, e.g.,

$$X(:,j,:) = AD_j(B)C^T \rightarrow \text{vec}(X(:,j,:)) = (C \odot A)(B(j,:))^T$$

Hence

$$X_2 := [\text{vec}(X(:,1,:)), \text{vec}(X(:,2,:)), \cdots, \text{vec}(X(:,J,:))] \rightarrow$$
$$X_2 = (C \odot A)B^T, \quad (IK \times J)$$

- Similarly for the horizontal slabs $X(i,:,:) = BD_i(A)C^T$,

$$X_1 := [\text{vec}(X(1,:,:)), \text{vec}(X(2,:;)), \cdots, \text{vec}(X(I,:,:))] \rightarrow$$
$$X_1 = (C \odot B)A^T, \quad (KJ \times I)$$

Slab Representations

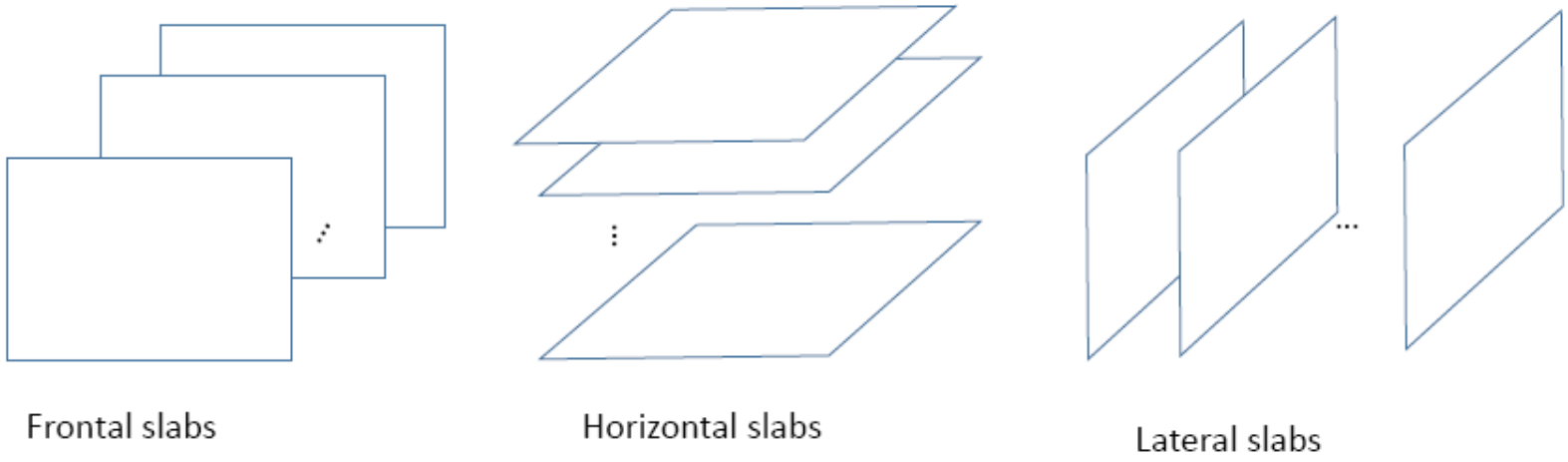


Figure: Slab views of a three-way tensor

- Frontal slabs: $X(:,:,k) = AD_k(C)B^T$.
- Horizontal slabs: $X(i,:,:) = BD_i(A)C^T$.
- Lateral slabs: $X(:,j,:) = AD_j(B)C^T$

Low-rank Tensor Approximation

- Adopting a least squares criterion, the problem is

$$\min_{A,B,C} \left\| X - \sum_{f=1}^{F} a_f \circledcirc b_f \circledcirc c_f \right\|_F^2$$

- Equivalently, we may consider

$$\min_{A,B,C} \left\| X_1 - (C \odot B) A^T \right\|_F^2$$

- Alternating optimization:

$$\begin{aligned} A &\leftarrow \arg\min_A \left\| X_1 - (C \odot B) A^T \right\|_F^2 \\ B &\leftarrow \arg\min_B \left\| X_2 - (C \odot A) B^T \right\|_F^2 \\ C &\leftarrow \arg\min_C \left\| X_3 - (B \odot A) C^T \right\|_F^2 \end{aligned}$$

- It is widely known as Alternating Least Squares (ALS).

Bounds of Tensor Rank

- Denote $X(:,:,k) = A_k B_k^T$.
  - $X(:,:,k)$ is of size $I \times J \Rightarrow A_k$ and B_k have at most $\min(I, J)$ columns.
- Let $A := [A_1, \cdots, A_K]$, $B := [B_1, \cdots, B_K]$ and $C := I_{K\times K} \otimes 1_{1\times \min(I,J)}$, we can synthesize X as $X = [A, B, C]$.
  - The kth frontal slab looks like

$$X(:,:,k) = [A_1, \ldots, A_K] \begin{bmatrix} 0 & \ldots & 0 \\ \vdots & \vdots & \vdots \\ \ldots & I_{\min(I,J)} & \ldots \\ \vdots & \vdots & \vdots \\ 0 & \ldots & 0 \end{bmatrix} [B_1, \ldots, B_K]^T$$

  - Implies at most $\min(IK, JK)$ columns in A, B, C to represent X.
  - Using role symmetry, the rank upper bound is $\min(IK, JK, IJ)$

Bounds of Tensor Rank

- Concatenate the frontal slabs one next to each other

$$[X(:,:,1),\cdots,X(:,:,K)] = A\left[D_1(C)B^T,\cdots,D_k(C)B^\top\right]$$

- Define

$$\begin{aligned} R_1(X) &:= \dim \text{colspan}(X) := \dim \text{span}\{X(:,j,k)\} \quad \forall j,k \\ R_2(X) &:= \dim \text{rowspan}(X) := \dim \text{span}\{X(i,:,k)\} \quad \forall i,k \\ R_3(X) &:= \dim \text{fiberspan}(X) := \dim \text{span}\{X(i,j,:)\} \quad \forall i,j \end{aligned}$$

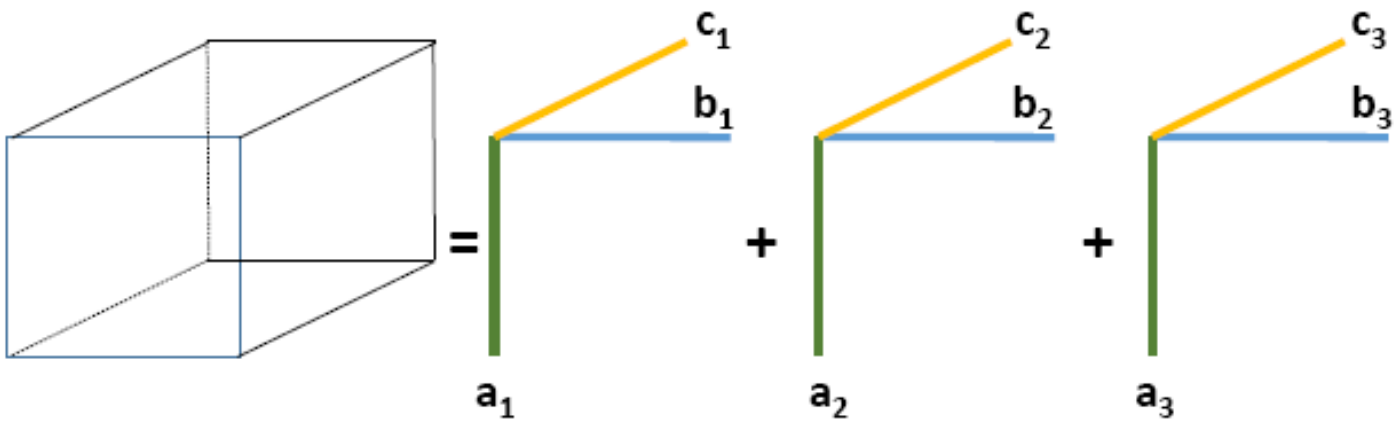
- We have $\max(R_1, R_2, R_3) \leqslant F$.
- Combining with our previous argument on upper bound, we have

$$\max(R_1, R_2, R_3) \leqslant F \leqslant \min(R_1R_2, R_2R_3, R_1R_3)$$

Essential Uniqueness

- If $X = [A, B, C]$, with $A : I \times F$, $B : J \times F$, and $C : K \times F$, then essential uniqueness means that A, B, and C are unique up to a common permutation and scaling of columns.
- Meaning that if $X = [A^{'}, B^{'}, C^{'}]$, for some $A^{'} : I \times F$, $B^{'} : J \times F$, and $C^{'} : K \times F$, then there exists a permutation matrix Π and diagonal scaling matrices $\Lambda_1, \Lambda_2, \Lambda_3$ such that

$$A^{'} = A\Pi\Lambda_1, B^{'} = B\Pi\Lambda_2, C^{'} = C\Pi\Lambda_3, \Lambda_1\Lambda_2\Lambda_3 = \mathbf{I}$$



Essential Uniqueness

- For tensors, low-rank tensor decomposition (essentially) unique for rank $\gg 1$.
- For matrices, only true for rank $= 1$, not interesting in most cases.
- The proof is correlated with Kruskal rank.

Definition

The Kruskal rank k_A of an $I \times F$ matrix A is the largest integer k such that any k columns of A are linearly independent.

Essential Uniqueness

Theorem

Given $X = [A, B, C]$, with $A : I \times F$, $B : J \times F$, and $C : K \times F$, if $k_A + k_B + k_C \geqslant 2F + 2$, then $\text{rank}(X) = F$ and the decomposition of X is essentially unique.

- Sharpness: there exist decompositions that are not unique as soon as F goes beyond the bound.
- This does not mean that uniqueness is impossible beyond Kruskal's bound!

Generalization to Any Order

Theorem

Given $X = [A_1, \cdots, A_N]$, with $A_n : I_n \times F$, if $\sum_{n=1}^{N} k_{A_n} \geqslant 2F + N - 1$, then the decomposition of X in terms of $\{A_n\}_{n=1}^{N}$ is essentially unique.

- This condition is sharp in the same sense as the $N = 3$ version.

Outline

1. Introduction
2. Low-rank Models
   - Canonical Polyadic Decomposition (CPD)
   - Tucker Decomposition
   - Multilinear SVD (MLSVD)
3. Algorithms
4. Applications in machine Learning

Tucker Decomposition

- Any $I \times J$ matrix X can be decomposed via SVD as $X = U\Sigma V^T$
- $U := [u_1, \cdot, u_l]$, $V := [v_1, \cdot, v_l]$, $\sigma_f := \Sigma(f, f)$

$$X = U(:, 1:F)\Sigma(1:F, 1:F)(V(:, 1:F))^T = \sum_{f=1}^{F} \sigma_f u_f v_f^T$$

- Can we generalize the SVD to tensors, in a way that retains the many beautiful properties of matrix SVD?

Tucker Decomposition

- Intuitively, introduce orthogonal matrices U, V, W, and a nonnegative $I \times J \times K$ core tensor Σ such that $\Sigma(i, j, k) > 0$ only when $k = j = i$.

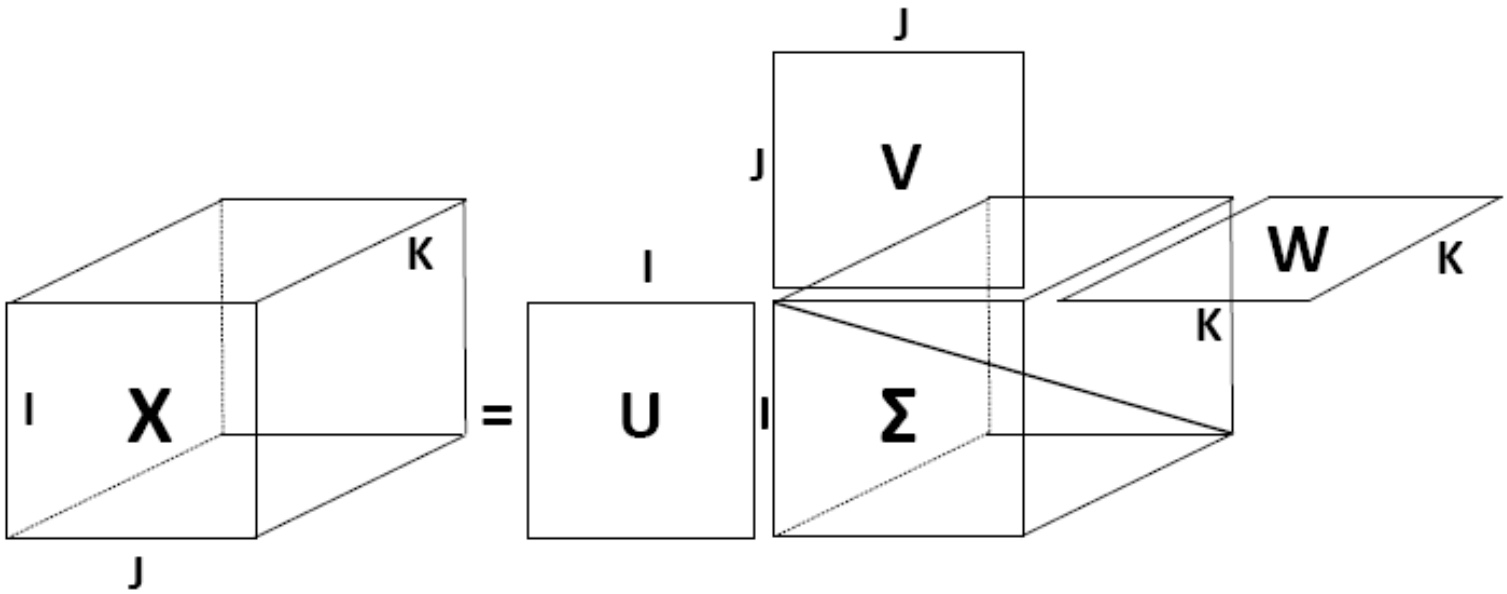


Figure: Diagonal tensor SVD?

Tucker Decomposition

- Can we write an arbitrary tensor X in this way?
- Back-of-the-envelop calculation:
  - DoF in model$= I^2 + J^2 + K^2 + \min(I, J, K)$;
  - $\#\text{equations} = IJK$
  - DoF$<$ #equations
- Contrast: for matrices: $I^2 + J^2 + \min(I, J) > I^2 + J^2 > IJ$, always!
- More formal look: postulated model can be written out as

$$\sigma_1 u_1 \odot v_1 \odot w_1 + \sigma_2 u_2 \odot v_2 \odot w_2 + \cdots + \sigma_m u_m \odot v_m \odot w_m$$

  where $m := \min(I, J, K)$; so, a tensor of rank at most $\min(I, J, K)$, but we know that tensor rank can be much higher than that.
- Hence we certainly have to give up diagonality.

Tucker Decomposition

- Consider instead a full (possibly dense, but ideally sparse) core tensor G

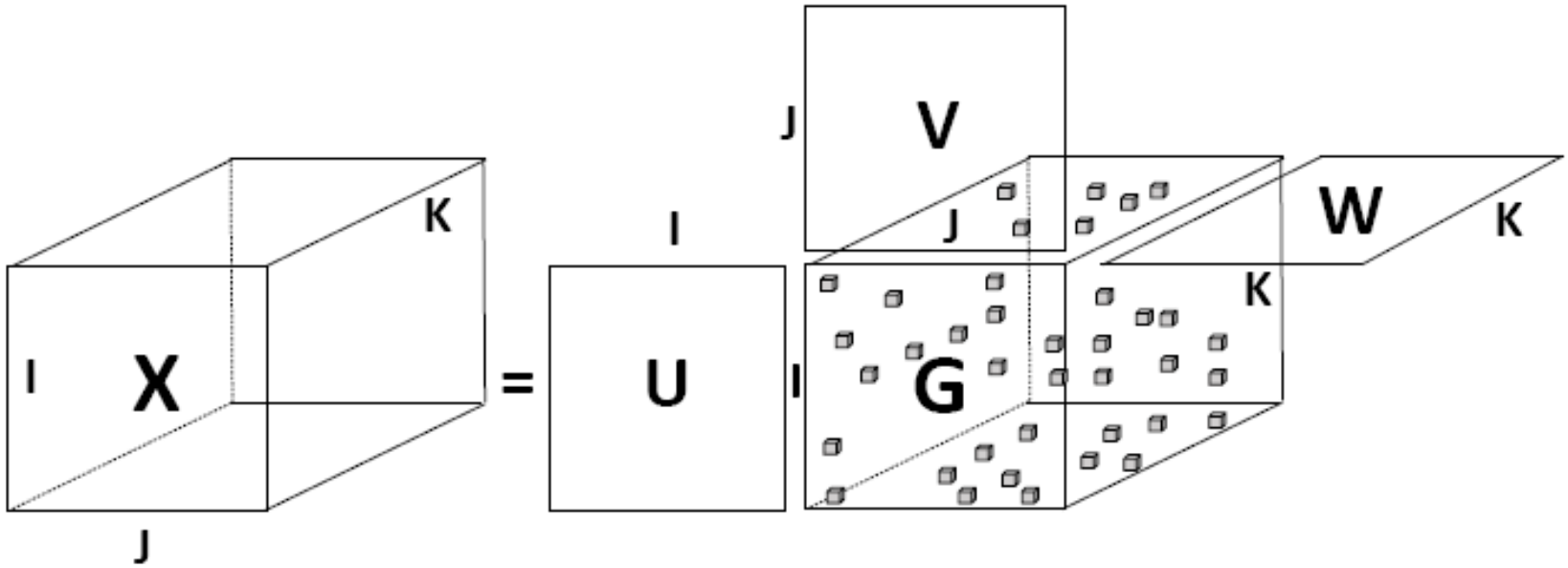


Figure: The Tucker model

Tucker Decomposition

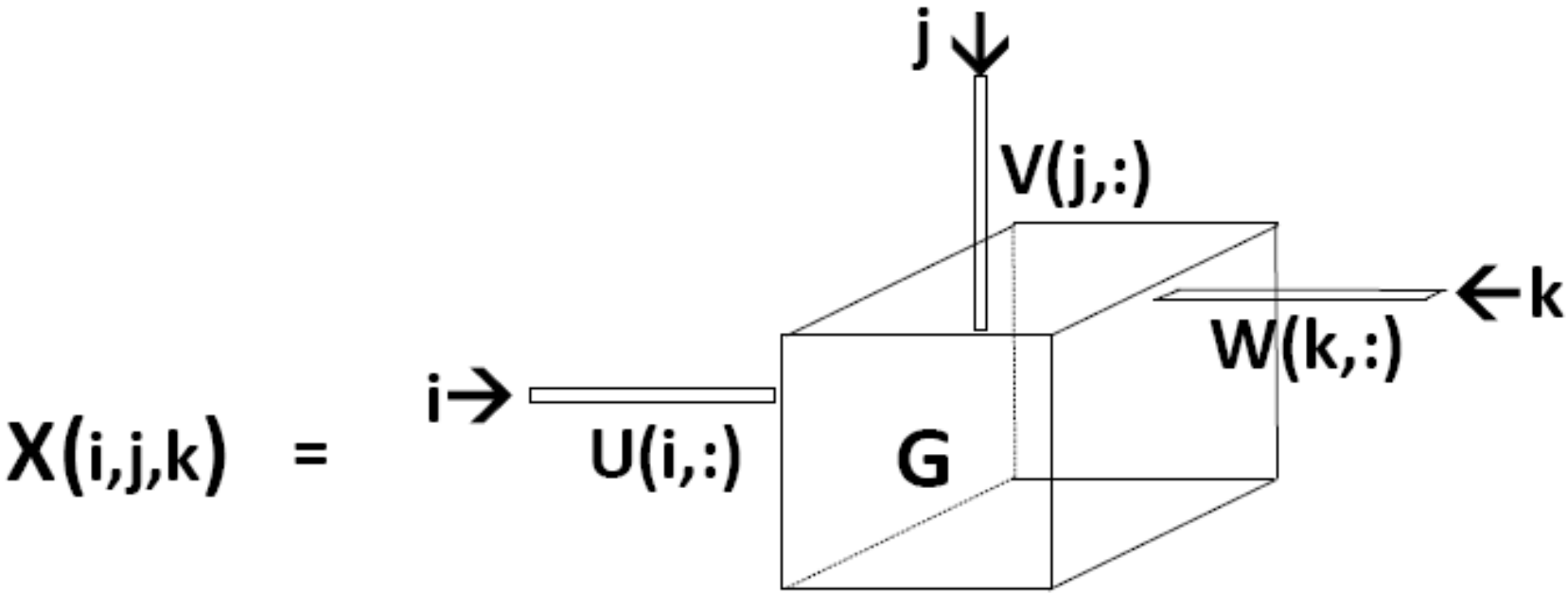


Figure: Element-wise view of the Tucker model

Tucker Decomposition

- From the last figure $\Rightarrow$

$$X(i,j,k) = \sum_{\ell=1}^{I} \sum_{m=1}^{J} \sum_{n=1}^{K} G(\ell,m,n) U(i,\ell) V(j,m) W(k,n)$$

- or, equivalently,

$$X = \sum_{\ell=1}^{I} \sum_{m=1}^{J} \sum_{n=1}^{K} G(\ell,m,n) U(:,\ell) \circledcirc V(:,m) \circledcirc W(:,n)$$

  or $X = \sum_{\ell=1}^{I} \sum_{m=1}^{J} \sum_{n=1}^{K} G(\ell,m,n) u_\ell \circledcirc v_m \circledcirc w_n$

- Here $u_\ell := U(:,\ell)$ and likewise for the v_m, w_n.

Tucker Decomposition

- Note that each column of U interacts with every column of V and every column of W in this decomposition.
- The strength of this interaction is encoded in the corresponding element of G.
- Different from CPD, which only allows interactions between corresponding columns of A, B, C, i.e., the only outer products that can appear in the CPD are of type $a_f \circledcirc b_f \circledcirc c_f$.

Tucker Decomposition

- Note that any tensor X can be written in Tucker form, and a trivial way of doing so is to take $U = I_{I\times I}$, $V = I_{J\times J}$, $W = I_{K\times K}$, and $G = X$.

- Hence we may seek a possibly sparse G, which could help reveal the underlying "essential" interactions between triples of columns of U, V, W.

- The main interest in Tucker is for finding subspaces and for tensor approximation purposes.

Tucker vs CPD

- CPD appears to be a special case of the Tucker model, corresponding to $G(\ell, m, n) = 0$ for all ℓ, m, n except possibly for $\ell = m = n$.

- However, when U, V, W are all square, such a restricted diagonal Tucker form can only model tensors up to rank $\min(I, J, K)$.

- If we allow "fat" (and therefore, clearly, non-orthogonal) U, V, W in Tucker, it is possible to think of CPD as a special case of such a "blown-up" non-orthogonal Tucker model.

Tucker vs CPD

In short,

- both CPD and Tucker are sum-of-outer-products models;
- one can argue that the most general form of one contains the other;
- what distinguishes the two is uniqueness;
- and modes of usage, which are quite different for the two models, as we will see.

Outline

1. Introduction
2. Low-rank Models
   - Canonical Polyadic Decomposition (CPD)
   - Tucker Decomposition
   - Multilinear SVD (MLSVD)
3. Algorithms
4. Applications in machine Learning

Multilinear SVD (MLSVD)

- Tucker model in the long vector form

$$x := \text{vec}(X) = (U \otimes V \otimes W)g$$

where $g := \text{vec}(G)$.

- From the properties of the Kronecker product, the expression above is the results of vectorization of matrix

$$X_1 = (V \otimes W)G_1 U^T$$

where the $KJ \times I$ matrix X_1 contains all rows (mode-1 vectors) of tensor X, and the $KJ \times I$ matrix G_1 is a likewise reshaped form of the core tensor G.

MLSVD

$$X_1 = (V \otimes W) G_1 U^T$$

- Notice we can linearly transform the columns of U and absorb the inverse transformation in G_1, i.e.,

$$G_1 U^T = G_1 M^{-T} (UM)^T$$

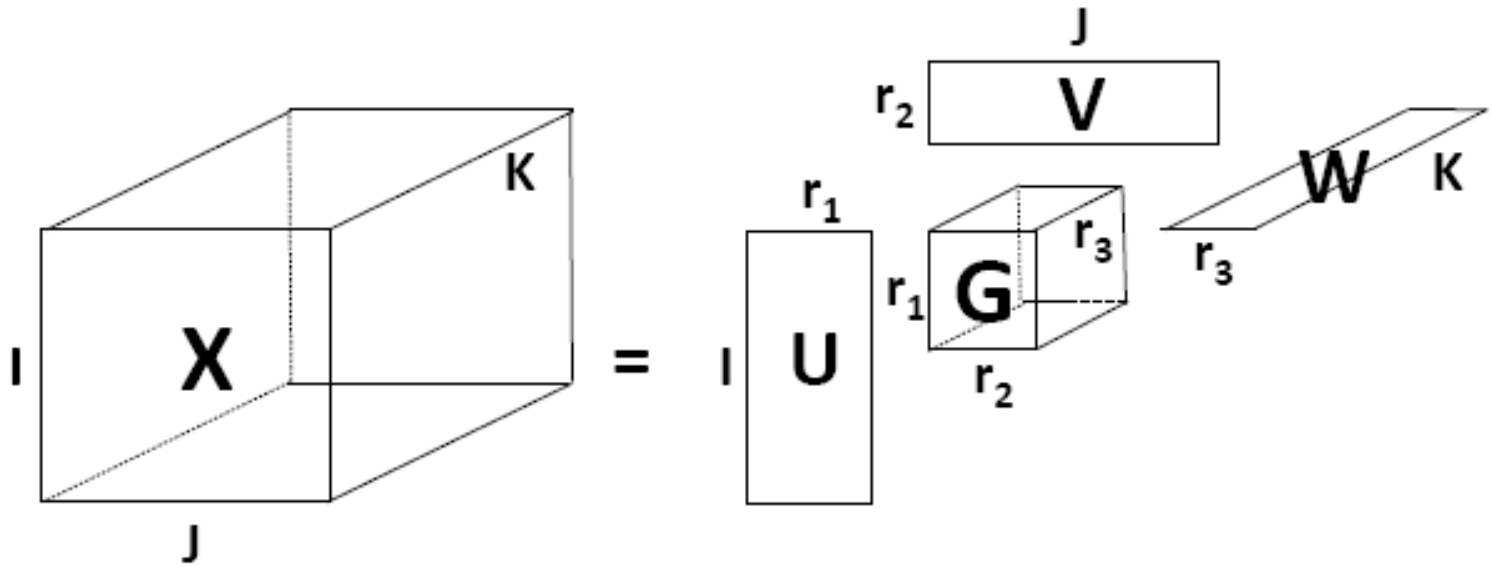
- Hence the Tucker model is not unique.
- X_1 contains all rows of tensor X; let r_1 denote the row-rank (mode-1 rank) of X.
- WLOG can pick U to be an $I \times r_1$ orthogonal basis of the row-span of X, and absorb the linear transformation in G, which is thereby reduced from $I \times J \times K$ to $r_1 \times J \times K$.

MLSVD

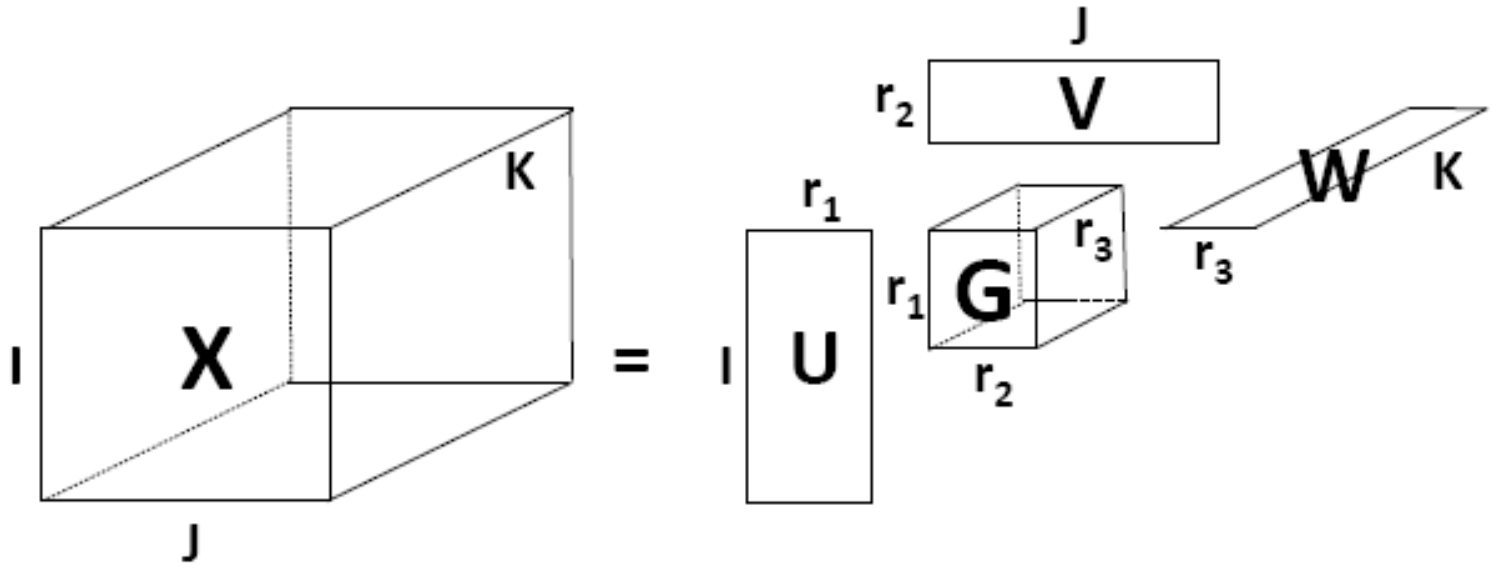
- The Tucker model can be written in compact form as

$$x := \mathrm{vec}(X) = (U_{r_1} \otimes V_{r_2} \otimes W_{r_3})\, g$$

where U_{r_1} is $I \times r_1$, V_{r_2} is $J \times r_2$, W_{r_3} is $K \times r_3$, and $g := \mathrm{vec}(G)$ is $r_1 r_2 r_3 \times 1-$ the vectorization of the $r_1 \times r_2 \times r_3$ reduced-size core tensor G.



MLSVD



- Drop subscripts from U_{r_1}, V_{r_2}, W_{r_3} for brevity.
- MLSVD = Tucker with orthonormal U, V, W chosen as the right singular vectors of the matrix unfoldings X_1, X_2, X_3, resp.

MLSVD

- Orthonormality of the columns of U_{r_1}, V_{r_2}, W_{r_3} implies orthonormality of the columns of their Kronecker product.
- This is because
  $(U_{r_1} \otimes V_{r_2})^T (U_{r_1} \otimes V_{r_2}) = \left(U_{r_1}^T \otimes V_{r_2}^T\right) (U_{r_1} \otimes V_{r_2}) =$
  $\left(U_{r_1}^T U_{r_1}\right) \otimes \left(V_{r_2}^T V_{r_2}\right) = \mathbf{I} \otimes \mathbf{I} = \mathbf{I}$
- Recall that
  $x_1 \perp x_2 \iff x_1^T x_2 = 0 \implies \|x_1 + x_2\|_2^2 = \|x_1\|_2^2 + \|x_2\|_2^2$
- It follows that

$$\|X\|_F^2 := \sum_{\forall i,j,k} |X(i,j,k)|^2 = \|x\|_2^2 = \|g\|_2^2 = \|G\|_F^2$$

where $x = \text{vec}(X)$, and $g = \text{vec}(G)$.

MLSVD

- For matrix SVD, zeroing out small singular values on the diagonal of $\Sigma \longrightarrow$ best low-rank approximation.
- Can we do the same thing for higher-order tensors?
- If we drop certain outer products from the decomposition $x = (U \otimes V \otimes W)g$, i.e., set the corresponding core elements to zero, then, by orthonormality

$$\|X - \widehat{X}\|_F^2 = \sum_{(\ell,m,n)\in\mathcal{D}} |G(\ell, m, n)|^2$$

where $\mathcal{D}$ is the set of dropped core element indices.

MLSVD

- Truncate the core, keeping only its upper-left-front dominant part of size $r_1^{'} \times r_2^{'} \times r_3^{'}$, with $r_1' \leqslant r_1$, $r_2' \leqslant r_2$, and $r_3' \leqslant r_3$.
- The resulting approximation error can be readily bounded as

$$\|X - \widehat{X}\|_F^2 \leqslant \sum_{\ell=r_1'+1}^{r_1} \|G(\ell,:,:)\|_F^2 + \sum_{m=r_2'+1}^{r_2} \|G(:,m,:)\|_F^2 + \sum_{n=r_3'+1}^{r_3} \|G(:,:,n)\|_F^2$$

  where we use $\leqslant$ as opposed to $=$ because dropped elements may be counted up to three times.
- One can of course compute the exact error of such a truncation strategy.

MLSVD

- Truncation in general does not yield the best approximation of X for the given (r_1', r_2', r_3').
- There is no exact equivalent of the Eckart-Young theorem for tensors of order higher than two.
- The best low multilinear rank approximation problem for tensors is NP-hard.
- Despite this "bummer", much of the beauty of matrix SVD remains in MLSVD.

MLSVD

- Simply truncating slabs (or elements) of the full score will not give the best low multilinear rank approximation of X in the case of three- and higher-order tensors.
- Error $\|X - \widehat{X}\|_F^2$ is in fact at most 3 times higher than the minimal error (N times higher in the N-th order case).
- A good approximation in practice as long as we pick high enough (r_1', r_2', r_3').

MLSVD - Optimization Problem

- Let $\left(\hat{U}, \hat{V}, \hat{W}, \hat{G}_1\right)$ be a solution to

$$\begin{array}{ll} \min_{(U,V,W,G_l)} & \left\| X_1 - (V \otimes W) G_1 U^T \right\|_F^2 \\ \text{s.t.} & U : I \times r_1, r_1 \leqslant I, U^T U = \mathbf{I} \\ & V : J \times r_2, r_2 \leqslant J, V^T V = \mathbf{I} \\ & W : K \times r_3, r_3 \leqslant K, W^T W = \mathbf{I} \\ & G_1 : r_3 r_2 \times r_1 \end{array}$$

- Claim: Then $\widehat{G}_1 = (\widehat{V} \otimes \widehat{W})^T X_1 \widehat{U}$
- Substituting the conditionally optimal G_1, problem recast in "concentrated" form

$$\begin{array}{ll} \max_{(U,V,W)} & \left\| (V \otimes W)^T X_1 U \right\|_F^2 \\ \text{s.t.} & \text{the same as above for } U, V, W \end{array}$$

MLSVD

Thus,

- $\widehat{U} =$ dominant $r_1 -$ dim . right subspace of $(\widehat{V} \otimes \widehat{W})^T X_1$;
- $\widehat{V} =$ dominant $r_2 -$ dim . right subspace of $(\widehat{U} \otimes \widehat{W})^T X_2$;
- $\widehat{W} =$ dominant $r_3 -$ dim . right subspace of $(\widehat{U} \otimes \widehat{V})^T X_3$;
- $\widehat{G_1}$ has orthogonal columns;
- $\left\{ \left\| \widehat{G}_1(:, m) \right\|_2^2 \right\}_{m=1}^{r_1} = r_1$ principal sing. vals of $(\widehat{V} \otimes \widehat{W})^T X_1$

MLSVD: Proof

- (Note: each column of $\widehat{G_1}$ is a vectorized slab of the core $\widehat{G}$)
- $\left\|\text{vec}\left(X_1\right) - \left(U \otimes V \otimes W\right)\text{vec}\left(G_1\right)\right\|_2^2 = \left\|X_1 - \left(V \otimes W\right)G_1 U^T\right\|_F^2$
- Conditioned on (orthonormal) U, V, W, the optimal G is given by $\text{vec}\left(\widehat{G}_1\right) = \left(U \otimes V \otimes W\right)^T \text{vec}\left(X_1\right)$
- Therefore $\widehat{G}_1 = \left(V \otimes W\right)^T X_1 U$
- Consider $\left\|X_1 - \left(V \otimes W\right)G_1 U^T\right\|_F^2$, and define $\widetilde{X}_1 := \left(V \otimes W\right)G_1 U^T$

MLSVD: Proof

- Use that

$$\left\|X_1 - \widetilde{X}_1\right\|_F^2 = \mathrm{Tr}\left(\left(X_1 - \widetilde{X}_1\right)^T\left(X_1 - \widetilde{X}_1\right)\right) = \|X_1\|_F^2 + \left\|\widetilde{X}_1\right\|_F^2 - 2\,\mathrm{Tr}\left(X_1^\top \widetilde{X}_1\right)$$

- Orthonormality of $U, V, W \Rightarrow \left\|\widetilde{X}_1\right\|_F^2 = \|G_1\|_F^2$
- Consider

$$-2\,\mathrm{Tr}\left(X_1^T \tilde{X}_1\right) = -2\,\mathrm{Tr}\left(X_1^T (V \otimes W) G_1 U^T\right)$$

- Substitute $G_1 = (V \otimes W)^T X_1 U$ to obtain

$$-2\,\mathrm{Tr}\left(X_1^T (V \otimes W)(V \otimes W)^T X_1 U U^T\right)$$

- Using property of trace to bring rightmost to the left,

$$-2\,\mathrm{Tr}\left(U^T X_1^T (V \otimes W)(V \otimes W)^T X_1 U\right) = -2\,\mathrm{Tr}\left(G_1^T G_1\right) = -2\,\|G_1\|_F^2$$

MLSVD: Proof

- It follows $\left\|X_1 - (V \otimes W) G_1 U^T\right\|_F^2 = \|X_1\|_F^2 - \|G_1\|_F^2 \iff$ maximize $\|G_1\|_F^2 = \left\|(V \otimes W)^T X_1 U\right\|_F^2$
- So $\widehat{U}$ is the dominant right subspace of $(\widehat{V} \otimes \widehat{W})^T X_1$
- take it to be the r_1 principle right singular vectors of $(\hat{V} \otimes \hat{W})^T X_1$
- Corresponding results for $\widehat{V}$ and $\widehat{W}$ by role symmetry.

Outline

1. Introduction
2. Low-rank Models
3. Algorithms
   - Alternating Least Squares (ALS)
   - Gradient Descent Methods
   - Imposing Constraints
4. Applications in machine Learning

Introduction to Alternating Least Squares (ALS)

- ALS is the “workhorse” algorithm for tensor decompositions
- Special case of Block Coordinate Descent (BCD)
- Flexible and easy to derive
- No parameter to tune
- General ALS iteration:
  - Fix all factor except for one
  - Solve the linear LS estimation problem for that factor
  - Cycle over all factors
- Today we will see ALS for CPD and Tucker

ALS for CPD

- Suppose we fix A, B, then update formula for C is :

$$C \leftarrow \arg\min_{C} \left\| X_3 - (B \odot A) C^T \right\|_F^2$$

- This is a simple linear Least Squares problem
- Solution: $C^T \leftarrow \left((B \odot A)^T (B \odot A) \right)^{-1} (B \odot A)^T X_3$

ALS for Tucker

1. Initialize:
   - $U = r_1$ principal right singular vectors of X_1;
   - $V = r_2$ principal right singular vectors of X_2;
   - $W = r_3$ principal right singular vectors of X_3;
2. repeat:
   - $U = r_1$ principal right singular vectors of $(V \otimes W)^T X_1$;
   - $V = r_2$ principal right singular vectors of $(U \otimes W)^T X_2$;
   - $W = r_3$ principal right singular vectors of $(U \otimes V)^T X_3$;
   - negligible change in $\|(V \otimes W)^T X_1 U\|_F^2$
3. $G_1 = (V \otimes W)^T X_1 U$

- Step 1 and 3 correspond to truncated MLSVD
  - Not necessarily optimal, but very good initialization in most cases

Outline

1. Introduction
2. Low-rank Models
3. Algorithms
   - Alternating Least Squares (ALS)
   - Gradient Descent Methods
   - Imposing Constraints
4. Applications in machine Learning

Gradient Descent

Consider the squared loss

$$\mathcal{L}(A,B,C) := \left\|X_1 - (C \odot B)A^T\right\|_F^2 = \operatorname{tr}\left(\left(X_1 - (C \odot B)A^T\right)^T \left(X_1 - (C \odot B)A^T\right)\right) = \|X_1\|_F^2 - 2\operatorname{tr}\left(X_1^T(C \odot B)A^T\right) + \operatorname{tr}\left(A(C \odot B)^T(C \odot B)A^T\right)$$

- Since $(C \odot B)^T(C \odot B) = \left(C^T C\right) * \left(B^T B\right)$
  $*$ denotes the Hadamard product.
- We may equivalently take the gradient of

$$-2\operatorname{tr}\left(X_1^T(C \odot B)A^T\right) + \operatorname{tr}\left(A\left(C^T C\right) * \left(B^T B\right)A^T\right)$$

Gradient Descent

- Arranging the gradient in the same format as A, we have

$$\begin{aligned}\frac{\partial \mathcal{L}(A,B,C)}{\partial A} &= -2X_1^T(C \odot B) + 2A\left[\left(C^TC\right) * \left(B^TB\right)\right] \\ &= -2\left(X_1^T - A(C \odot B)^T\right)(C \odot B)\end{aligned}$$

- Appealing to role symmetry, we likewise obtain $\frac{\partial \mathcal{L}(A,B,C)}{\partial B}$ and $\frac{\partial \mathcal{L}(A,B,C)}{\partial C}$
- Remark Taking a gradient step we needn't invert the $F \times F$ matrix $\left(C^TC\right) * \left(B^TB\right)$ like the LS sub-problem.

Tensor Completion

- Real data are never perfect
- Parts of data mat be missing for various reasons
  - Faulty measurement sensors
  - Errors when storing data
  - Partial observation of the data (e.g., recommendation systems/Netflix prize)
  - ...
- Need to modify algorithms to handle missing values
- In the case of recommendation systems, also need to use decomposition to estimate missing values.

Tensor Completion

- Consider the C-update step in ALS,

$$\min_C \left\| X_3 - (B \odot A) C^T \right\|_F^2$$

- If there are missing elements in X (and so in X_3), define the weight tensor

$$W(i,j,k) = \begin{cases} 1, & X(i,j,k): \text{ available} \\ 0, & \text{otherwise} \end{cases}$$

- We now modify the update step as

$$\min_C \left\| W_3 * \left(X_3 - (B \odot A) C^T \right) \right\|_F^2$$

Handling Missing Values

$$\min_{C} \left\| W_3 * \left(X_3 - (B \odot A) C^T \right) \right\|_F^2$$

- Notice that the Hadamard operation applies to the product $\left((B \odot A) C^T\right)$, not to $(B \odot A)$ –This complicates things
- We perform row-wise updates on C, then we have to deal with minimizing over $C(k,:)$ the squared norm of vector

$$\text{Diag}\left(W_3(:,k)\right) X_3(:,k) - \text{Diag}\left(W_3(:,k)\right) (B \odot A) \left(C(k,:)\right)^T$$

  which is a simple linear least squares problem.
- Utilizing stochastic gradient descent, computes gradient estimates by drawing only from the observed values

Stochastic Gradient Descent (SGD)

- Popular in the machine learning community for many types of convex non-convex problems vfill
- In its simplest form:
  - SGD randomly picks a data point $X(i, j, k)$ from the available ones,
  - takes a gradient step only for those model parameters that have an effect on $X(i, j, k)$; that is, only the i-th row of A, the j-th row of B and the k-th row of C

Stochastic Gradient Descent (SGD)

we have

$$\frac{\partial}{\partial A(i,f)}\left(X(i,j,k)-\sum_{f=1}^{F}A(i,f)B(j,f)C(k,f)\right)^2=$$

$$-2\left(X(i,j,k)-\sum_{f'=1}^{F}A\left(i,f'\right)B\left(j,f'\right)C\left(k,f'\right)\right)\times B(j,f)C(k,f)$$

so that

$$\frac{\partial}{\partial A(i,:)}=-2\left(X(i,j,k)-\sum_{t=1}^{F}A(i,f)B(j,f)C(k,f)\right)\times(B(j,:)*C(k,:))$$

- SGD naturally deals with missing elements
- SGD shows efficiency in very large, sparse data

Outline

1. Introduction
2. Low-rank Models
3. Algorithms
   - Alternating Least Squares (ALS)
   - Gradient Descent Methods
   - Imposing Constraints
4. Applications in machine Learning

Imposing Constraints

Why do we do this? CPD is essentially unique after all!

- Restoring identifiability in otherwise non-identifiable cases
- Improving estimation accuracy in relatively challenging (low-SNR, and/or barely identifiable, and/or numerically illconditioned) cases
- Ensuring interpretability of the results (e.g., power spectra cannot take negative values)
- As a remedy against ill-posedness

Imposing Constraints

Symmetry or Hermitian (conjugate) symmetry:

- Require $B = A$, or $B = A^*$, leading to $X(:,:,k) = AD_k(C)A^T$ or $X(:,:,k) = AD_k(C)A^H$.
- Full symmetry; $A = B = C$.
- Symmetric tensors arise when one considers higher-order statistics.

Imposing Constraints

Element-wise non-negativity:

- $A \geqslant 0$ and or $B \geqslant 0$, and or $C \geqslant 0$
- When all three are in effect, the resulting problem is known as non-negative tensor factorization (NTF).
- non-negativity can still ensure essential uniqueness of A, B, C.
- Applications:
  - Modeling power spectra
  - Modeling ”sum-of-parts” representation (generally interpretability)

Imposing Constraints

- Orthogonality: This e.g., may be the result of prewhitening
- Probability simplex constraints: $A(i,:) \geqslant 0, A(i,:)\mathbf{1} = 1, \forall i$ or $A(:,f) \geqslant 0, \mathbf{1}^T A(:,f) = 1, \forall f$, are useful when modeling allocations or probability distributions.
- Linear constraints:More general linear constraints on A, B, C are also broadly used. Can be column-wise, row-wise, or matrix-wise, such as $tr(WA) \leqslant b$.

Imposing Constraints

- Sparsity: In many cases we know (an upper bound on) the number of nonzero elements of A, B, C, or as a whole;
- Smoothness: Smoothness can be measured in different ways, but a simple one is in terms of convex quadratic inequalities such as

$$\left\| \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & & \end{bmatrix} \mathbf{A} \right\|_F^2$$

Imposing Constraints

- Data model constraints: We may also be interested in constraints on the reconstructed model of the data, e.g.,

$$(A \odot B \odot C)\mathbf{1} \geqslant 0, \text{ (element-wise)}, \mathbf{1}^T(A \odot B \odot C)\mathbf{1} = 1$$

  If X models a joint probability distribution, or in $(A \odot B \odot C)\mathbf{1}$ being "smooth" in a suitable sense.
- Parametric constraints: Particularly important in signal processing constraints. Examples include include Vandermonde or Toeplitz structure imposed on A, B, C.

Problem formulation

Starting with constrained matrix factorization, we formulate the problem as

$$\underset{\mathbf{W},\mathbf{H}}{\text{minimize}} \frac{1}{2} \left\| \mathbf{Y} - \mathbf{W}\mathbf{H}^T \right\|_F^2 + r(\mathbf{W}) + r(\mathbf{H})$$

- Easily becomes NP-hard when there are constraints.
- Popular method: alternating optimization (AO).
- Key to accelerate algorithm: solve the sub-problems efficiently. We propose to solve the sub-problems using ADMM.

ADMM: scaled form

Consider a convex optimization problem in the following form.

$$\begin{aligned} \underset{x,z}{\text{minimize}} \quad & f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c \end{aligned}$$

The alternating direction method of multipliers:

$$\begin{aligned} x &\leftarrow \arg\min_x f(x) + (\rho/2)\|Ax + Bz - c + u\|_2^2 \\ z &\leftarrow \arg\min_z g(z) + (\rho/2)\|Ax + Bz - c + u\|_2^2 \\ u &\leftarrow u + (Ax + Bz - c), \quad \text{dual update} \end{aligned}$$

For convex problems, converges to the global solution.

Constrained least squares using ADMM

For 3-way tensor, the update of C becomes

$$\begin{aligned} \underset{C,\tilde{C}}{\text{minimize}} \quad & \tfrac{1}{2}\|X_3 - WC^T\|_F^2 + f_C(\tilde{C}) \\ \text{subject to} \quad & \tilde{C} = C \end{aligned}$$

where $W := (B \odot A)$. The ADMM iterates for this problem are

$$\begin{aligned} C^T &\leftarrow \left(W^T W + \rho I\right)^{-1} \left(W^T X_3 + \rho(\widetilde{C} + U)^T\right) \\ \widetilde{C} &\leftarrow \arg\min_C f_C(\widetilde{C}) + \frac{\rho}{2}\|\widetilde{C} - C + U\|_F^2 \\ U &\leftarrow U + \widetilde{C} - C \end{aligned}$$

Implementation: C

$$C^T \leftarrow \left(W^T W + \rho I\right)^{-1} \left(W^T X_3 + \rho(\widetilde{C} + U)^T\right)$$

- $W^T X_3$ and $W^T W$ only need to be computed once, with complexity $\mathcal{O}(nnz(Y)k)$ and $\mathcal{O}\left(mk^2\right)$ respectively;
- Can cache the Cholesky decomposition $\left(\mathcal{O}\left(k^3\right)\right)$ of $W^T W + \rho I = LL^T$;
- Within one ADMM iteration, $\widetilde{C}$ is obtained from one forward substitution and one backward substitution, with complexity $\mathcal{O}\left(nk^2\right)$

Implementation: $\tilde{C}$

$$\widetilde{C} \leftarrow \arg\min_{C} f_C(\widetilde{C}) + \frac{\rho}{2}\|\widetilde{C} - C + U\|_F^2$$

so-called proximity operator, in a lot of cases can be efficiently evaluated

- non-negative: $\tilde{C} \geqslant 0$;
- l_1 regularization: $\lambda\|\tilde{C}\|_1$ (soft thresholding);
- sum to one: $H\mathbf{1} = \mathbf{1}$ or $H^T\mathbf{1} = \mathbf{1}$;
- smoothness regularization.

All with $\mathcal{O}(nk)$ complexity.

AO-ADMM: Reprise

AO-ADMM can be viewed as a generalization of the workhorse ALS method:

- similar monotonic convergence property (due to AO);
- per-iteration complexity is almost the same;
- any smart implementation of ALS can easily be modified to handle constraints with the corresponding proximity operator: plug-and-play!!
- extended to handle robust tensor factorization problems where some slabs are grossly corrupted.

AO-ADMM: recap

- Efficiency: Uses ADMM to solve the subproblems, with computation caching, warm start, and good choice of ρ;
- Flexibility:
  - Can handle many constraints on the latent factors with element-wise complexity;
  - Can handle non-LS loss: missing values, l_1 fitting, K-L divergence...
- Convergence:
  - Monotonic decrease of the cost function;
  - Adopting block successive upperbound minimization (BSUM) as the AO framework, we can guarantee that every limit point is a stationary point.
- Bottleneck: same as ALS.

Back to ALS

$$C = X_{(3)}(B \odot A)\left(B^\top B * A^\top A\right)^\dagger$$
$$A = X_{(1)}(C \odot B)\left(C^T C * B^T B\right)^\dagger$$
$$B = X_{(2)}(C \odot A)\left(C^T C * A^T A\right)^\dagger$$

- Computation and inversion of $\left(C^T C * B^T B\right)$ relatively easy: relatively small $K \times F$, $J \times F$ matrices, invert $F \times F$ matrix, usually F is small.
- Bottleneck is computing $x_1(C \odot B)$; likewise $x_2(C \odot A)$, $X_{(3)}(B \odot A)$.
- Entire X needs to be accessed for each computation in each ALS iteration, data transport costs.
- Memory access pattern is different for the three computations, making efficient block caching very difficult.
- ”Solution”: replicate data three times in main (fast) memory.

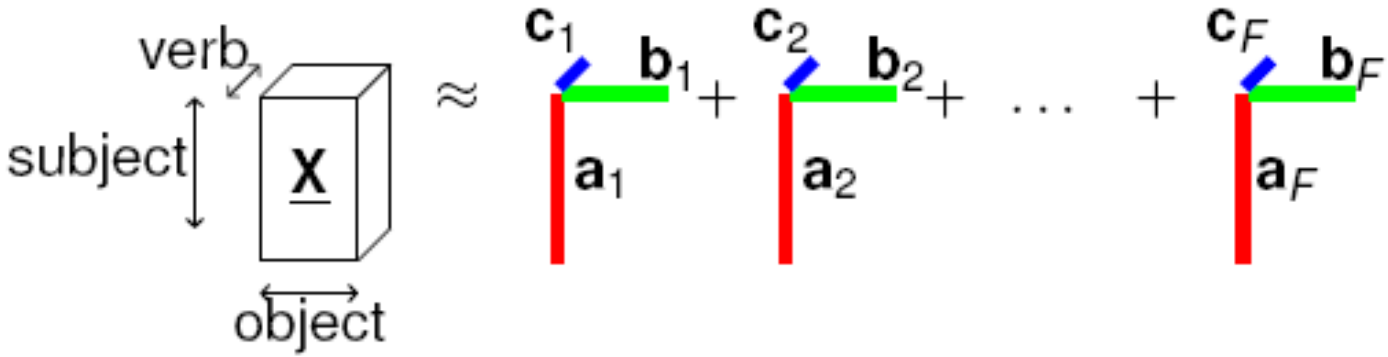
Outline

1. Introduction
2. Low-rank Models
3. Algorithms
4. Applications in machine Learning
   - Knowledge Base Completion
   - Recommender Systems
   - Multilinear Classification
   - Topic Mining
   - Discriminative Subspace Learning

Knowledge Base Completion

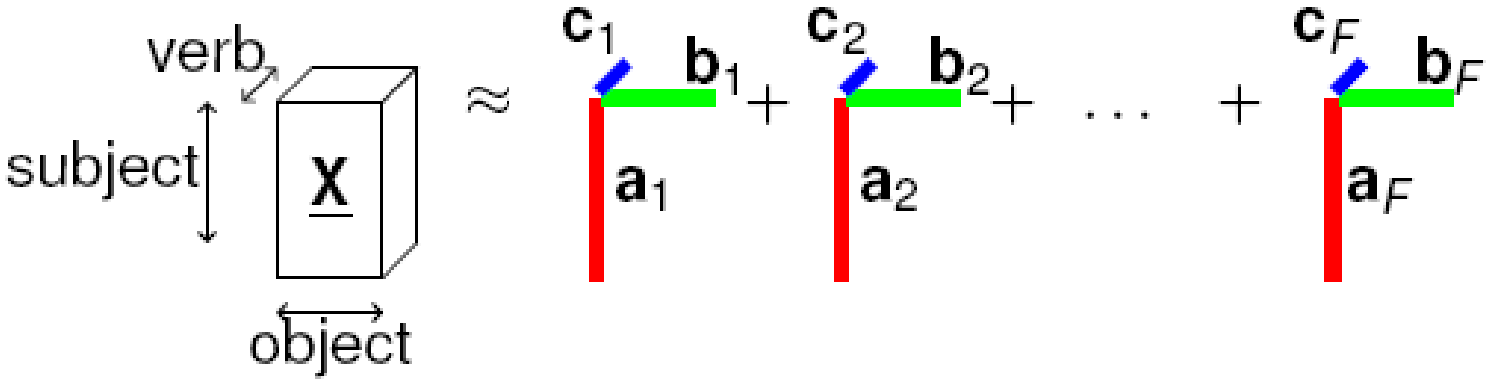
[Chang et al., 2014]

- Given known facts of (subject, verb, object)-type such as (Obama, likes, football), (Donald, likes, football), (Donald, is, American) infer additional knowledge, such as (Obama, is, American).



- You obviously need a model for this...and the most basic algebraic model is low-rank...but does low-rank make sense in this context?

Knowledge Base Completion



- Think ... about matrix case in which you unfold verb-object in one long mode: the long rows for Obama and Donald are similar, so you copy entries from one to the other to make them identical $\leftrightarrow$ matrix completion!
- If you don't unfold $\leftrightarrow$ tensor completion.

Outline

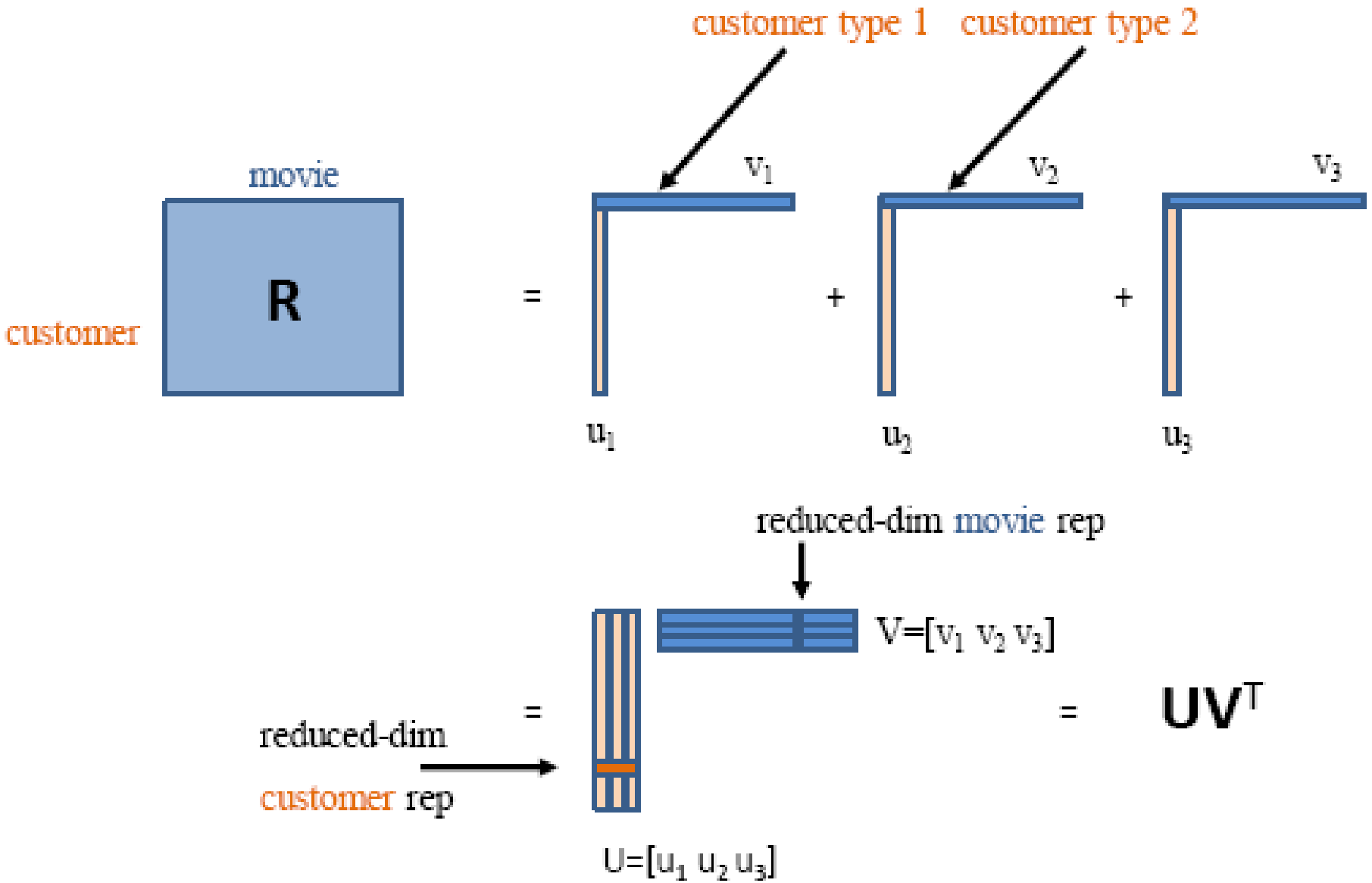
1. Introduction
2. Low-rank Models
3. Algorithms
4. Applications in machine Learning
   - Knowledge Base Completion
   - Recommender Systems
   - Multilinear Classification
   - Topic Mining
   - Discriminative Subspace Learning

Recommender Systems: Collaborative Filtering

[Karatzoglou et al., 2010]

- Given users $\times$ movies (I $\times$ J) ratings matrix R.
- Low-rank approx. $R \approx UV^T$; $U(I \times F)$, $V(J \times F)$, $F \ll \min(I, J)$
- $U(i,:)$: reduced-dim latent description of user i;
- $V(j,:)$: reduced-dim latent description of user j;
- $R \approx UV^T$ implies that user i's rating of movie j is approximated as $R(i,j) \approx U(i,:)(V(j,:))^T$, i.e., the inner product of the latent descriptions of the i-th user and the j-th movie. Intuitive!
- Premise: every user is a linear combination of few (F) user "types" (e.g., sports fan, college student, ... $\leftrightarrow$ rows of V^T / columns of V).
- Every movie is a linear combination of few movie types (e.g., comedy, drama, documentary, ... $\leftrightarrow$ columns of U). Intuitive!

Collaborative Filtering



Recommender Systems: Collaborative Filtering

- Only very small % of the entries of R is available-between 1 per thousand and 1 per 10^5 in practice. Few people provide feedback.
- Goal: predict a user's missing ratings using not only that user's past ratings but also the ratings of all other users - hence the term collaborative filtering.
- If we can find U and V from the available ratings, then we can impute the missing ones using inner products of columns of U and V. This suggests using the following formulation.

$$\min_{U,V} \left\| W * \left(R - UV^T \right) \right\|_F^2$$

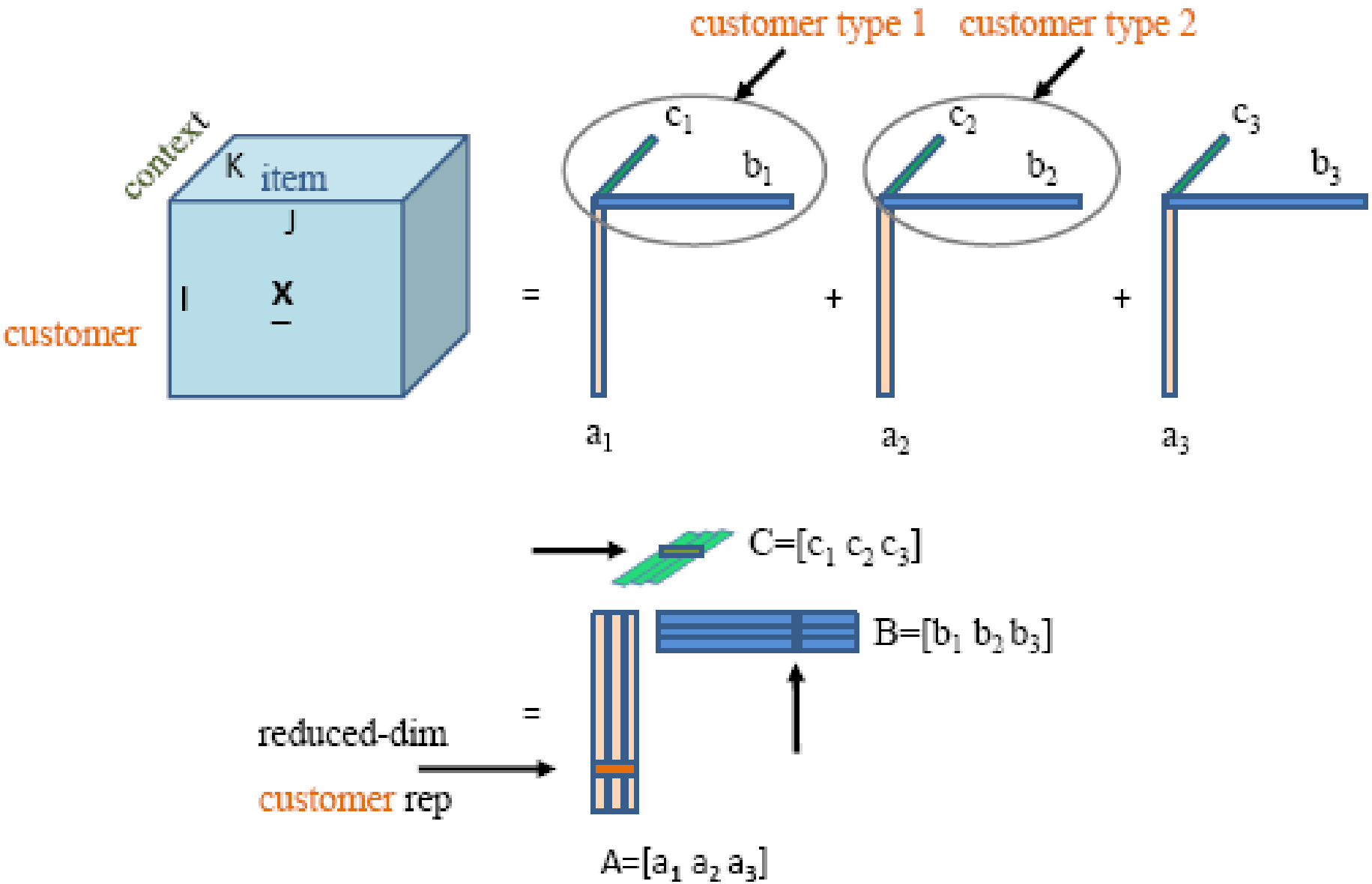
where $W(i,j) = 1$ if $R(i,j)$ is available, 0 otherwise.

Recommender Systems: Collaborative Filtering

- Additional information on the context in which the ratings were given is often available: time, social context, gender, ...
- Every context type +1 mode $\rightarrow$ very sparse higher-order tensors.
- Bummer: approximating very sparse may require very high rank ...
- CPD of user $\times$ movie $\times$ time tensor R w/ elements $R(i,j,k)$

$$\min_{A,B,C} \sum_{k=1}^{K} \left\| W(:,:,k) * \left(R(:,:,k) - AD_k(C)B^T \right) \right\|_F^2$$

Machine learning interpretation of CPD



Outline

1. Introduction
2. Low-rank Models
3. Algorithms
4. Applications in machine Learning
   - Knowledge Base Completion
   - Recommender Systems
   - Multilinear Classification
   - Topic Mining
   - Discriminative Subspace Learning

Multilinear classification

[Rendle, 2010]

- In SVM classification, we use a linear mapping $w^T x = \sum_i w(i)x(i)$ to discriminate between vectors belonging to two different classes: $\text{sign}\left(w^T x - b\right)$
- Augmenting each x with a unit as last element (i.e., replacing x by $[x, 1]^T$) and likewise absorbing $-b$ in $w \rightarrow \text{sign}\left(w^T x\right)$.
- Classifier design: choice of vector w.
- What if classes are not linearly separable?
- Standard ML approach: use kernels and the kernel trick.

Multilinear classification

- What if classes are not linearly separable?
- System approach?
- Bilinear mapping $x^T W x = \sum_{i,j} W(i,j)x(i)x(j) = \text{vec}\left(x^T W x\right) = \left(x^T \otimes x^T\right) \text{vec}(W) = (x \otimes x)^T \text{vec}(W)$
- More generally, multilinear mapping, e.g., $(x \otimes x \otimes x)^T \text{vec}(W) = \sum_{i,j,k} W(i,j,k)x(i)x(j)x(k)$
- Classifier design problem boils down to designing a suitable matrix or tensor W of weights.

Outline

1. Introduction
2. Low-rank Models
3. Algorithms
4. Applications in machine Learning
   - Knowledge Base Completion
   - Recommender Systems
   - Multilinear Classification
   - Topic Mining
   - Discriminative Subspace Learning

Topic Mining

[Anandkumar et al., 2013]

FastAnchor (classic)					AnchorFree (proposed)				
allegations	poll	columbia	gm	bulls	lewinsky	gm	shuttle	bulls	jonesboro
lewinsky	cnnusa	shuttle	motors	jazz	monica	motors	space	jazz	arkansas
clinton	gallup	space	plants	nba	starr	plants	columbia	nba	school
lady	allegations	crew	workers	utah	grand	flint	astronauts	chicago	shooting
white	clinton	astronauts	michigan	finals	white	workers	nasa	game	boys
hillary	presidents	nasa	flint	game	jury	michigan	crew	utah	teacher
monica	rating	experiments	strikes	chicago	house	auto	experiments	finals	students
starr	lewinsky	mission	auto	jordan	clinton	plant	rats	jordan	westside
house	president	stories	plant	series	counsel	strikes	mission	malone	middle
husband	approval	fix	strike	malone	intern	gms	nervous	michael	11year
dissipate	starr	repair	gms	michael	independent	strike	brain	series	fire
president	white	rats	idled	championship	president	union	aboard	championship	girls
intern	monica	unit	production	tonight	investigation	idled	system	karl	mitchell
affair	house	aboard	walkouts	lakers	affair	assembly	weightlessness	pippen	shootings
infidelity	hurting	brain	north	win	lewinskys	production	earth	basketball	suspects
grand	slipping	system	union	karl	relationship	north	mice	win	funerals
jury	americans	broken	assembly	lewinsky	sexual	shut	animals	night	children
sexual	public	nervous	talks	games	ken	talks	fish	sixth	killed
justice	sexual	cleansing	shut	basketball	former	autoworkers	neurological	games	13year
obstruction	affair	dioxide	striking	night	starrs	walkouts	seven	title	johnson

Figure: Mined topics from 5 classes of (1,683) articles of the TDT2 corpus.

Topic Mining

- Given a dictionary $\mathcal{D} = \{w_1, \cdots, w_l\}$ comprising l possible words, a topic is a probability mass function (pmf) over $\mathcal{D}$.
- Assume there are F topics overall, let $p_f := \Pr(w_i|f)$ be the pmf associated with topic f, π_f be the probability that one may encounter a document associated with topic f, and $\pi := [\pi_1, \cdots, \pi_f]^T$.
- We begin our discussion of topic modeling by assuming that each document is related to one and only one topic (or, document "type").

Topic Mining

- Consider the following experiment:
  1) Draw a document at random;
  2) Sample m words from it, independently, and at random (with replacement- the order in which words are drawn does not matter);
  3) Repeat (until you collect ”enough samples”-to be qualified later).

Topic Mining

- Assume the number of topics F is known. Goal is to estimate $\{p_f, \pi_f\}_{f=1}^F$
- Clearly, $\Pr(w_i) = \sum_{f=1}^F \Pr(w_i|f)\,\pi_f$
- Furthermore, the word co-occurrence probabilities $\Pr(w_i, w_j) := \Pr(\text{ word } i \text{ and word } j \text{ are drawn from the same document})$ satisfy

$$\Pr(w_i, w_j) = \sum_{f=1}^F \Pr(w_i, w_j|f)\,\pi_f = \sum_{f=1}^F p_f(i)p_f(j)\pi_f$$

since the words are independently drawn from the document.

Topic Mining

- Define the matrix of word co-occurrence probabilities $P^{(2)}$ with elements $P^{(2)}(i,j) := \Pr(w_i, w_j)$, and the matrix of conditional pmfs $C := [p_1, \cdots, p_F]$. Then

$$P^{(2)} = C\,\mathrm{Diag}(\pi)C^T$$

- Next, consider "trigrams" - i.e., probabilities of triples of words being drawn from the same document

$$\Pr(w_i, w_j, w_k) = \sum_{f=1}^{F} \Pr(w_i, w_j, w_k|f)\,\pi_f = \sum_{f=1}^{F} p_f(i)p_f(j)p_f(k)\pi_f$$

- Define tensor $P^{(3)}$ with elements $P^{(3)}(i,j,k) := \Pr(w_i, w_j, w_k)$. Then $P^{(3)}$ admits a symmetric non-negative CPD model of rank (at most) F :

$$P^{(3)} = (C \odot C \odot C)\pi$$

Topic Mining

- We can estimate C and π from the tensor $P^{(3)}$ and the matrix $P^{(2)}$.
- In reality, we will use empirical word co-occurrence counts to estimate $P^{(3)}$ and $P^{(2)}$, and for this we need to sample enough triples ("enough samples").
- Once we have C, we can classify any document by estimating (part of) its conditional word pmf and comparing it to the columns of C.

Topic Mining

- Next, consider the more realistic situation where each document is a mixture of topics, modeled by a pmf $q(F \times 1)$ that is itself drawn from a distribution $\delta(\cdot)$ over the $(F-1)$ -dimensional probability simplex.
- Our working experiment is now modified as follows:
  1) For every document we sample, we draw $\mathbf{q} \sim \delta(\cdot)$
  2) For every word we sample from the given document, we first draw a topic t from $\mathbf{q}-$ i.e., topic f is selected with probability $\mathbf{q}(f)$
  3) Next, we draw a word $\sim \mathbf{p}_t$;
  4) Goto 2, until you have sampled the desired number of words (e.g. three) from the given document;
  5) Goto 1, until you have collected enough samples (e.g., enough triples of words).

Topic Mining

- Then,

$$\Pr(w_i, w_j | t_1, t_2, q) = p_{t_1}(i) p_{t_2}(j) \Longrightarrow$$
$$\Pr(w_i, w_j | q) = \sum_{t_1=1}^{F} \sum_{t=1}^{F} p_{t_1}(i) p_{t_2}(j) q(t_1) q(t_2) \Longrightarrow$$
$$\Pr(w_i, w_j) = \sum_{t_1=1}^{F} \sum_{t_2=1}^{F} p_{t_1}(i) p_{t_2}(j) E[q(t_1) q(t_2)]$$

  where we notice that what comes into play is the second-order statistics $E[q(t_1) q(t_2)]$ (the correlation) of $\delta(\cdot)$

- The $I \times I$ matrix Q with elements $Q(i,j) := \Pr(w_i, w_j)$ admits the decomposition $Q = CEC^T$, where $E(t_1, t_2) := E[q(t_1) q(t_2)]$

Topic Mining

- Likewise, it follows that, for the trigrams $\Pr(w_i, w_j, w_k) = \sum_{t_1=1}^{F} \sum_{t_2=1}^{F} \sum_{t_3=1}^{F} p_{t_1}(i) p_{t_2}(j) p_{t_3}(k) E[q(t_1) q(t_2) q(t_3)]$ which involves the third-order statistics tensor G of $\delta(\cdot)$ with elements $G(t_1, t_2, t_3) := E[q(t_1) q(t_2) q(t_3)]$
- Defining the $I \times I \times I$ tensor P with elements $P(i, j, k) := \Pr(w_i, w_j, w_k)$, it follows that P admits a symmetric Tucker decomposition:

$$P = \text{Tucker}(C, C, C, G), \text{ with } C = [p_1, \cdots, p_F]$$

Topic Mining

$$P = \text{Tucker}\,(C, C, C, G), \text{with } C = [p_1, \cdots, p_F]$$

- Note that C is element-wise non-negative, but in principle G may have negative elements.
- As we know, Tucker models are not identifiable in general- there is linear transformation freedom.
- This can be alleviated when one can assume sparsity in C G, or both (intuitively, this is because linear transformations generally do not preserve sparsity).

Outline

1. Introduction
2. Low-rank Models
3. Algorithms
4. Applications in machine Learning
   - Knowledge Base Completion
   - Recommender Systems
   - Multilinear Classification
   - Topic Mining
   - Discriminative Subspace Learning

Discriminative Subspace Learning

[Lu et al., 2011]

- Given $X = [x_1, \cdots, x_M]$ $(N \times M)$ and associated class labels $z = [z_1, \cdots, z_M]$ $(1 \times M)$ for the columns of X .
- Find a dimensionality-reducing linear transformation U of size $N \times F, F < N$ (usually $F \ll N$) such that

$$\min_{U|U^TU=1} \sum_{m=1}^{M} \left\{ (1-\lambda) \sum_{\ell=1|Z_\ell=Z_m}^{M} \left\| U^T x_m - U^T x_\ell \right\|_2^2 - \lambda \sum_{\ell=1|Z_\ell \neq Z_m}^{M} \left\| U^T x_m - U^T x_\ell \right\|_2^2 \right\} \tag{1}$$

where the first (second) term measures the within-class (across-class) distance in reduced dimension space.

Discriminative Subspace Learning

- Interpretation: find a dimensionality-reducing transformation that will map points close to each other in terms of Euclidean distance if they have the same class label, far otherwise.
- Find a subspace to project onto where we can easily visualize (if $F = 2$ or 3) the point clouds of the different classes.
- Upon defining

$$w_{m,\ell} := (1-\lambda)^{1(z_\ell = z_m)} (-\lambda)^{1-1(z_\ell = Z_m)}$$

where $1(z_\ell = z_m) = 1$ if $z_\ell = z_m$, 0 otherwise, we can compactly write the problem as follows

$$\min_{U \| U^T U = I} \sum_{m=1}^{M} \sum_{\ell=1}^{M} \left\| U^T x_m - U^T x_\ell \right\|_2^2 w_{m,\ell}$$

Discriminative Subspace Learning

- Expanding the squared norm and using properties of $Tr(\cdot)$, we can write the cost function as

$$\sum_{m=1}^{M}\sum_{\ell=1}^{M}\left\|U^T x_m - U^T x_\ell\right\|_2^2 w_{m,\ell} = \mathrm{Tr}\left(UU^T Y\right)$$

where

$$\mathbf{Y} := \sum_{m=1}^{M}\sum_{\ell=1}^{M} w_{m,\ell}\left(x_m - x_\ell\right)\left(x_m - x_\ell\right)^T$$

Notice that $w_{m,\ell} = w_{\ell,m}$ by definition, and Y is symmetric. Let $Y = V\Lambda V^T$ be the eigen decomposition of Y, and note that $\mathrm{Tr}\left(UU^T Y\right) = \mathrm{Tr}\left(U^T YU\right)$. Clearly, $U_{\text{opt}} = F$ minor eigenvectors of Y (columns of V corresponding to the F smallest elements on the diagonal of Λ).

Multilinear Subspace Learning

- Now, suppose that the columns in X are in fact vectorized tensors. As an example, suppose that there exist common bases $U\left(I \times r_1\right)$, $V\left(J \times r_2\right)$, $W\left(K \times r_3\right)$, such that

$$x_m \approx (U \otimes V \otimes W) g_m, \quad \forall m \in \{1, \cdots, M\}$$

  i.e., each x_m can be modeled using a Tucker model with common mode bases, but different cores for different m.
- Think of $(U \otimes V \otimes W)^T \left(r_1 r_2 r_3 \times IJK\right)$ as a (Kronecker) structured dimensionality reducing transformation;
- Think of the vectorized core array g_m as the low-dimensional $\left(r_1 r_2 r_3 \times 1\right)$ representation of x_m.

Multilinear Subspace Learning

- Want to find U, V, W such that the g 's corresponding to x 's in the same (different) class are close (far) from each other.
- Following the same development as before, using

$$\widehat{g}_m = (U \otimes V \otimes W)^T x_m$$

as the projection of $\mathbf{x}_m$ in reduced-dimension space, we arrive at

$$\min_{U,V} \mathrm{Tr}\left((U \otimes V \otimes W)(U \otimes V \otimes W)^T Y\right)$$
$$\text{subject to: } U^T U = \mathbf{I},\ V^T V = \mathbf{I},\ W^T W = \mathbf{I}$$

Multilinear Subspace Learning

- Equivalently,

$$\min_{U,V} \text{Tr}\left((U \otimes V \otimes W)(U \otimes V \otimes W)^T Y\right)$$
$$\text{subject to: } U^T U = \mathbf{I}, V^T V = \mathbf{I}, W^T W = \mathbf{I}$$

- It is now clear that, conditioned on, say, U and V , the update with respect to W boils down to

$$\min_{W | W^T W = 1} \text{Tr}\left(W W^T Z\right)$$

for some matrix Z that depends on the values of U and V .

References I


Anandkumar, A., Hsu, D. J., Janzamin, M., and Kakade, S. M. (2013).
When are overcomplete topic models identifiable? uniqueness of tensor tucker decompositions with structured sparsity.
In *Advances in neural information processing systems*, pages 1986–1994.


Chang, K.-W., Yih, W.-t., Yang, B., and Meek, C. (2014).
Typed tensor decomposition of knowledge bases for relation extraction.
In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1568–1579.


Karatzoglou, A., Amatriain, X., Baltrunas, L., and Oliver, N. (2010).
Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering.
In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM.


Lu, H., Plataniotis, K. N., and Venetsanopoulos, A. N. (2011).
A survey of multilinear subspace learning for tensor data.
Pattern Recognition, 44(7):1540–1551.


Rendle, S. (2010).
Factorization machines.
In *2010 IEEE International Conference on Data Mining*, pages 995–1000. IEEE.

References II

Sidiropoulos, N. D., De Lathauwer, L., Fu, X., Huang, K., Papalexakis, E. E., and Faloutsos, C. (2017).
Tensor decomposition for signal processing and machine learning.
IEEE Transactions on Signal Processing, 65(13):3551–3582.