Factorization Models for Recommender Systems and Other Applications

Part II

Lars Schmidt-Thieme, Steffen Rendle

Universität Konstanz

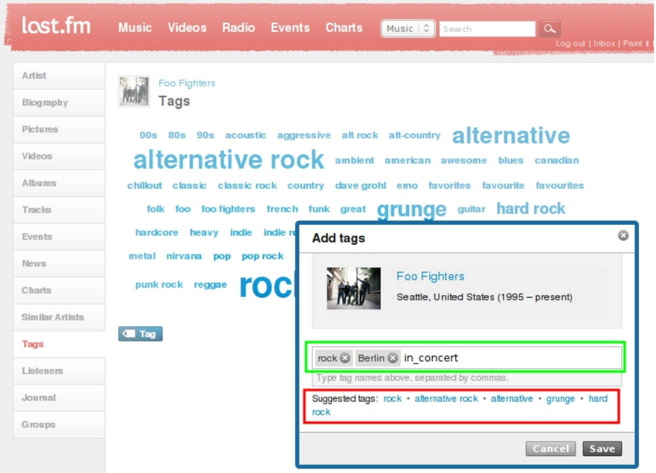
Tutorial at KDD Conference, 12th August 2012, Beijing

Outline

Problem Setting

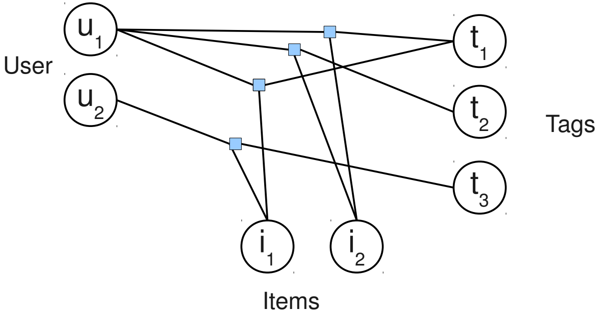
- Predictor variables: m variables of categorical domain $I_1, \ldots, I_m$.
- Target y: Real-valued (regression), binary (classification), scores (ranking).
- Supervised task: set of observations $S = \{(i_1, \ldots, i_m, y), \ldots\}$

Example: Social Tagging[1]



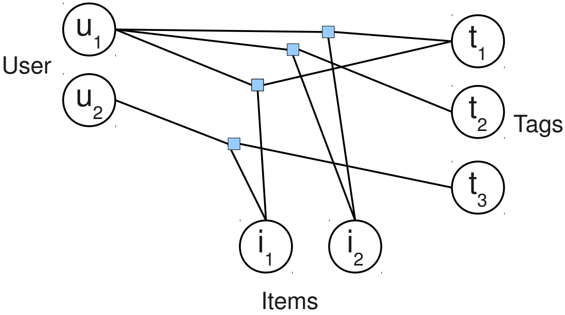
[1] http://last.fm

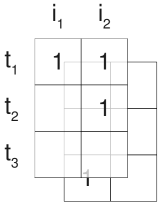
Example: Social Tagging[1]



[1] http://last.fm

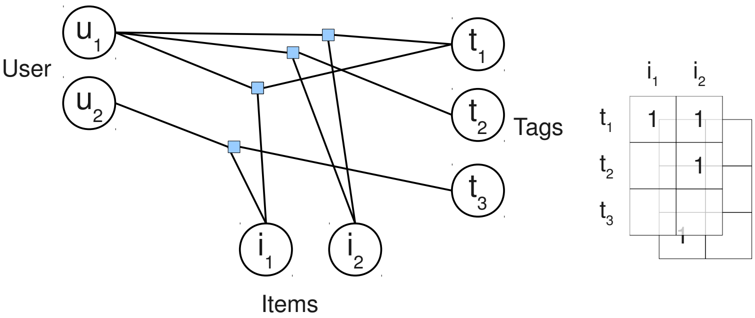
Example: Social Tagging[1]





[1] http://last.fm

Example: Social Tagging[1]

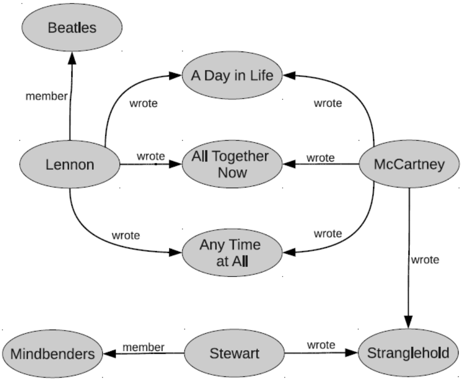


Tagging can be expressed as a function over three categorical domains:

$$y : U \times I \times T \to \{0, 1\}$$

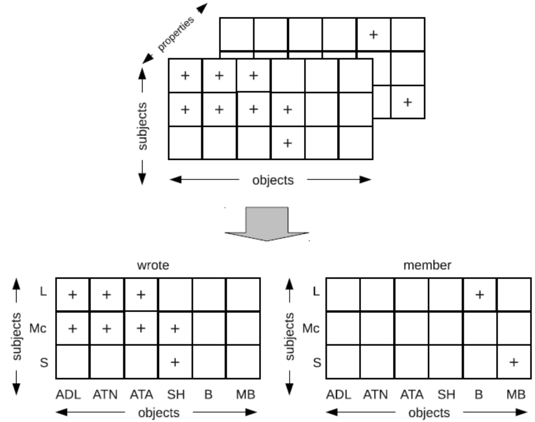
[1] http://last.fm

Example: Querying Incomplete RDF-Graphs



- **Task:** Answer queries about subject-predicate-pairs. E.g. What is McCartney member of?

Example: Querying Incomplete RDF-Graphs



An RDF-Graph can be expressed as a function over three categorical domains: $y : S \times P \times O \rightarrow \{0, 1\}$

Notation: Tensors and Functions

Models in this setting are functions:

$$\hat{y} : I_1 \times \ldots \times I_m \to \mathcal{Y}$$

Notation: Tensors and Functions

Models in this setting are functions:

$$\hat{y} : I_1 \times \ldots \times I_m \to \mathcal{Y}$$

All possible targets and predictions can be written equivalently as a *m-order tensor* / multiway array:

$$Y \in \mathcal{Y}^{|I_1| \times \ldots \times |I_m|}, \quad \hat{Y} \in \mathcal{Y}^{|I_1| \times \ldots \times |I_m|}$$

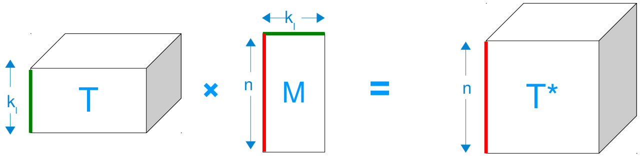
where

$$y(i_1, \ldots, i_m) = y_{i_1,\ldots,i_m}, \quad \hat{y}(i_1, \ldots, i_m) = \hat{y}_{i_1,\ldots,i_m}$$

Notation: Tensor-Matrix product

- Let $T \in \mathbb{R}^{k_1 \times \ldots \times k_m}$ be a m-order tensor and $V \in \mathbb{R}^{n \times k_l}$ be a matrix.
- The *mode-l tensor-matrix product* $\times_l$ is defined as:

$$(T \times_l M)_{i_1,\ldots,i_{l-1},j,i_{l+1},\ldots,i_m} := \sum_{i_l=1}^{k_l} t_{i_1,\ldots,i_m}\, m_{j,i_l}$$

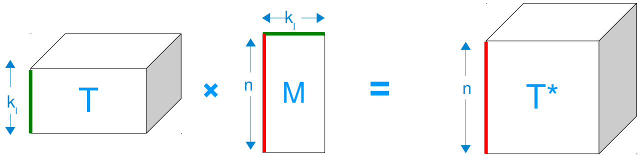


Notation: Tensor-Matrix product

- Let $T \in \mathbb{R}^{k_1 \times \ldots \times k_m}$ be a m-order tensor and $V \in \mathbb{R}^{n \times k_l}$ be a matrix.
- The *mode-l tensor-matrix product* $\times_l$ is defined as:

$$(T \times_l M)_{i_1,\ldots,i_{l-1},j,i_{l+1},\ldots,i_m} := \sum_{i_l=1}^{k_l} t_{i_1,\ldots,i_m}\, m_{j,i_l}$$

- The result is a tensor T^* of dimension $\mathbb{R}^{k_1 \times k_{l-1} \times n \times k_{l+1} \times \ldots \times k_m}$

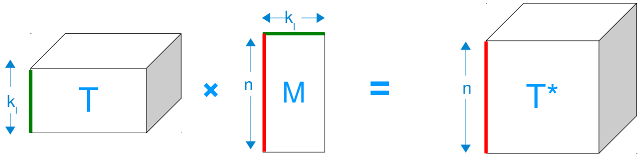


Notation: Tensor-Matrix product

- Let $T \in \mathbb{R}^{k_1 \times \ldots \times k_m}$ be a *m*-order tensor and $V \in \mathbb{R}^{n \times k_l}$ be a matrix.
- The *mode-l tensor-matrix product* $\times_l$ is defined as:

$$(T \times_l M)_{i_1,\ldots,i_{l-1},j,i_{l+1},\ldots,i_m} := \sum_{i_l=1}^{k_l} t_{i_1,\ldots,i_m}\, m_{j,i_l}$$

- The result is a tensor T^* of dimension $\mathbb{R}^{k_1 \times k_{l-1} \times n \times k_{l+1} \times \ldots \times k_m}$
- The size of the l-th mode changes from k_l to n.

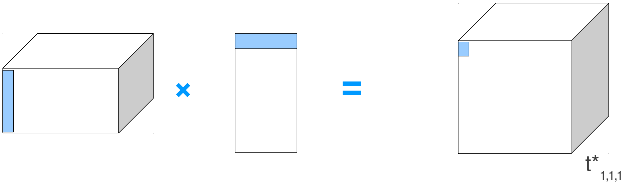


Notation: Tensor-Matrix product

- Let $T \in \mathbb{R}^{k_1 \times \ldots \times k_m}$ be a m-order tensor and $V \in \mathbb{R}^{n \times k_l}$ be a matrix.
- The *mode-l tensor-matrix product* $\times_l$ is defined as:

$$(T \times_l M)_{i_1,\ldots,i_{l-1},j,i_{l+1},\ldots,i_m} := \sum_{i_l=1}^{k_l} t_{i_1,\ldots,i_m} \, m_{j,i_l}$$

- The result is a tensor T^* of dimension $\mathbb{R}^{k_1 \times k_{l-1} \times n \times k_{l+1} \times \ldots \times k_m}$
- The size of the l-th mode changes from k_l to n.

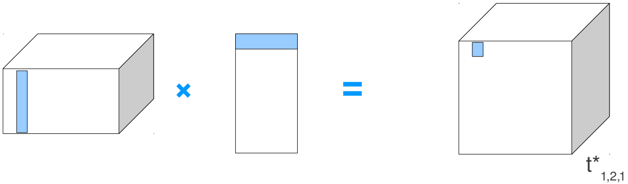


Notation: Tensor-Matrix product

- Let $T \in \mathbb{R}^{k_1 \times \ldots \times k_m}$ be a m-order tensor and $V \in \mathbb{R}^{n \times k_l}$ be a matrix.
- The *mode-l tensor-matrix product* $\times_l$ is defined as:

$$(T \times_l M)_{i_1,\ldots,i_{l-1},j,i_{l+1},\ldots,i_m} := \sum_{i_l=1}^{k_l} t_{i_1,\ldots,i_m} \, m_{j,i_l}$$

- The result is a tensor T^* of dimension $\mathbb{R}^{k_1 \times k_{l-1} \times n \times k_{l+1} \times \ldots \times k_m}$
- The size of the l-th mode changes from k_l to n.

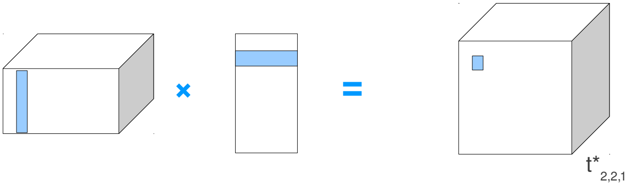


Notation: Tensor-Matrix product

- Let $T \in \mathbb{R}^{k_1 \times \ldots \times k_m}$ be a m-order tensor and $V \in \mathbb{R}^{n \times k_l}$ be a matrix.
- The *mode-l tensor-matrix product* $\times_l$ is defined as:

$$(T \times_l M)_{i_1,\ldots,i_{l-1},j,i_{l+1},\ldots,i_m} := \sum_{i_l=1}^{k_l} t_{i_1,\ldots,i_m} \, m_{j,i_l}$$

- The result is a tensor T^* of dimension $\mathbb{R}^{k_1 \times k_{l-1} \times n \times k_{l+1} \times \ldots \times k_m}$
- The size of the l-th mode changes from k_l to n.



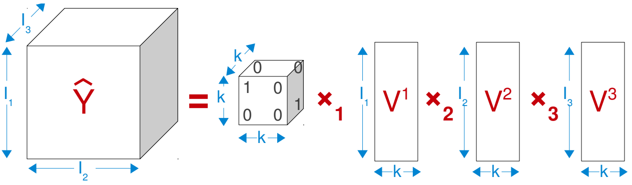
Outline

Tensor Factorization
- Problem Setting
- Models
- Learning
- Examples for Applications
- Summary

Time-aware Factorization Models

Factorization Machines

Parallel Factor Analysis (PARAFAC)



m-order PARAFAC in tensor product notation:

$$\hat{Y} := C \times_1 V^{(1)} \times_2 \ldots \times_m V^{(m)}$$

with model parameters

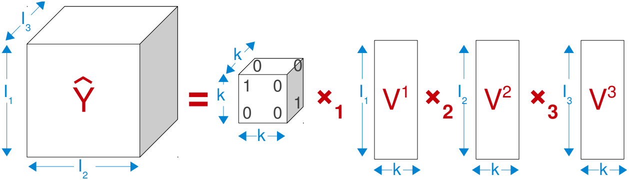
$$V^{(l)} \in \mathbb{R}^{|I_l| \times k}, \quad \forall l \in \{1, \ldots, m\}$$

and where C is the identity tensor:

$$C \in \mathbb{R}^{k \times \ldots \times k}, \quad c_{j_1,\ldots,j_m} := \delta(j_1 = \ldots = j_m)$$

[Harshman 1970, Carroll 1970]

Parallel Factor Analysis (PARAFAC)



m-order PARAFAC in element-wise notation:

$$\hat{y}(i_1, \ldots, i_m) := \sum_{f=1}^{k} v_{i_1,f}^{(1)} \ldots v_{i_m,f}^{(m)} = \sum_{f=1}^{k} \prod_{l=1}^{m} v_{i_l,f}^{(l)}$$

with model parameters

$$V^{(l)} \in \mathbb{R}^{|I_l| \times k}, \quad \forall l \in \{1, \ldots, m\}$$

[Harshman 1970, Carroll 1970]

Parallel Factor Analysis (PARAFAC)

Notes

- For a $m = 2$-order tensor (i.e. a matrix), PARAFAC is the same as matrix factorization.

[e.g. Kolda et al. 2009, Cichocki et al. 2009]

Parallel Factor Analysis (PARAFAC)

Notes

- For a $m = 2$-order tensor (i.e. a matrix), PARAFAC is the same as matrix factorization.
- Sometimes a modified PARAFAC with diagonal core $c_{f,\dots,f} =: \lambda_f$ and factors of unit length is used:

$$\hat{y}(i_1, \dots, i_m) := \sum_{f=1}^{k} \lambda_f \, v^{(1)}_{i_1,f} \dots v^{(m)}_{i_m,f} = \sum_{f=1}^{k} \lambda_f \prod_{l=1}^{m} v^{(l)}_{i_l,f}$$

[e.g. Kolda et al. 2009, Cichocki et al. 2009]

Parallel Factor Analysis (PARAFAC)

Notes

- For a $m = 2$-order tensor (i.e. a matrix), PARAFAC is the same as matrix factorization.
- Sometimes a modified PARAFAC with diagonal core $c_{f,\ldots,f} =: \lambda_f$ and factors of unit length is used:

$$\hat{y}(i_1, \ldots, i_m) := \sum_{f=1}^{k} \lambda_f \, v^{(1)}_{i_1,f} \ldots v^{(m)}_{i_m,f} = \sum_{f=1}^{k} \lambda_f \prod_{l=1}^{m} v^{(l)}_{i_l,f}$$

- Other constraints, e.g. non-negativity or symmetry can be imposed.

[e.g. Kolda et al. 2009, Cichocki et al. 2009]

Parallel Factor Analysis (PARAFAC)

Notes

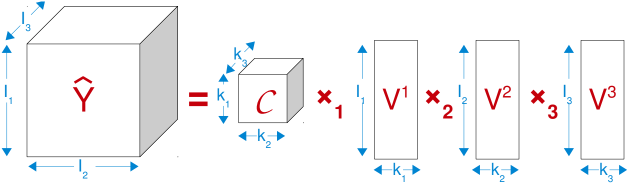
- For a $m = 2$-order tensor (i.e. a matrix), PARAFAC is the same as matrix factorization.
- Sometimes a modified PARAFAC with diagonal core $c_{f,\ldots,f} =: \lambda_f$ and factors of unit length is used:

$$\hat{y}(i_1, \ldots, i_m) := \sum_{f=1}^{k} \lambda_f \, v_{i_1,f}^{(1)} \ldots v_{i_m,f}^{(m)} = \sum_{f=1}^{k} \lambda_f \prod_{l=1}^{m} v_{i_l,f}^{(l)}$$

- Other constraints, e.g. non-negativity or symmetry can be imposed.
- PARAFAC is also called Canonical Decomposition (CANDECOMP).

[e.g. Kolda et al. 2009, Cichocki et al. 2009]

Tucker Decomposition (TD)



m-order Tucker Decomposition in tensor product notation:

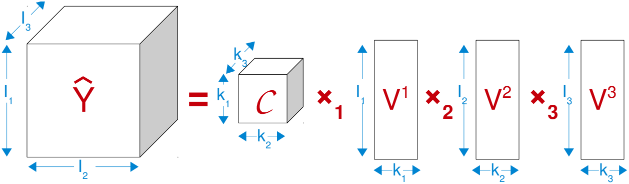
$$\hat{Y} := C \times_1 V^{(1)} \times_2 \ldots \times_m V^{(m)}$$

where V **and** C are model parameters:

$$C \in \mathbb{R}^{k_1 \times \ldots \times k_m}, \quad V^{(l)} \in \mathbb{R}^{|I_l| \times k_l}, \quad \forall l \in \{1, \ldots, m\}$$

[Tucker 1966]

Tucker Decomposition (TD)



m-order Tucker Decomposition in element-wise notation:

$$\hat{y}(i_1, \ldots, i_m) := \sum_{f_1=1}^{k_1} \ldots \sum_{f_m=1}^{k_m} c_{f_1,\ldots,f_m} \, v_{i_1,f_1}^{(1)} \ldots v_{i_m,f_m}^{(m)} = \sum_{f_1=1}^{k_1} \ldots \sum_{f_m=1}^{k_m} c_{f_1,\ldots,f_m} \prod_{l=1}^{m} v_{i_l,f_l}^{(l)}$$

with model parameters:

$$C \in \mathbb{R}^{k_1 \times \ldots \times k_m}, \quad V^{(l)} \in \mathbb{R}^{|I_l| \times k_l}, \quad \forall l \in \{1, \ldots, m\}$$

[Tucker 1966]

Tucker Decomposition (TD)

Notes

- For a $m = 2$-order tensor (i.e. a matrix), TD is different from matrix factorization:

$$\hat{y}^{\mathrm{TD}}(i_1, i_2) = \sum_{f_1=1}^{k_1} \sum_{f_2=1}^{k_2} c_{f_1,f_2} v_{i_1,f_1}^{(1)} v_{i_2,f_2}^{(2)} \neq \sum_{f=1}^{k} v_{i_1,f}^{(1)} v_{i_2,f}^{(2)} = \hat{y}^{\mathrm{MF}}(i_1, i_2)$$

[e.g. Kolda et al. 2009, Cichocki et al. 2009]

Tucker Decomposition (TD)

Notes

- For a $m = 2$-order tensor (i.e. a matrix), TD is different from matrix factorization:

$$\hat{y}^{\mathrm{TD}}(i_1, i_2) = \sum_{f_1=1}^{k_1} \sum_{f_2=1}^{k_2} c_{f_1,f_2} v^{(1)}_{i_1,f_1} v^{(2)}_{i_2,f_2} \neq \sum_{f=1}^{k} v^{(1)}_{i_1,f} v^{(2)}_{i_2,f} = \hat{y}^{\mathrm{MF}}(i_1, i_2)$$

- Sometimes orthogonality constraints on V are imposed.

[e.g. Kolda et al. 2009, Cichocki et al. 2009]

Tucker Decomposition (TD)

Notes

- For a $m = 2$-order tensor (i.e. a matrix), TD is different from matrix factorization:

$$\hat{y}^{\mathrm{TD}}(i_1, i_2) = \sum_{f_1=1}^{k_1} \sum_{f_2=1}^{k_2} c_{f_1,f_2} v^{(1)}_{i_1,f_1} v^{(2)}_{i_2,f_2} \neq \sum_{f=1}^{k} v^{(1)}_{i_1,f} v^{(2)}_{i_2,f} = \hat{y}^{\mathrm{MF}}(i_1, i_2)$$

- Sometimes orthogonality constraints on V are imposed.
- Other constraints, e.g. non-negativity or symmetry can be imposed.

[e.g. Kolda et al. 2009, Cichocki et al. 2009]

PARAFAC vs. TD

- PARAFAC

$$\hat{y}(i_1, \ldots, i_m) := \sum_{f=1}^{k} v_{i_1,f}^{(1)} \ldots v_{i_m,f}^{(m)} = \sum_{f=1}^{k} \prod_{l=1}^{m} v_{i_l,f}^{(l)}$$

- TD

$$\hat{y}(i_1, \ldots, i_m) := \sum_{f_1=1}^{k_1} \ldots \sum_{f_m=1}^{k_m} c_{f_1,\ldots,f_m}\, v_{i_1,f_1}^{(1)} \ldots v_{i_m,f_m}^{(m)} = \sum_{f_1=1}^{k_1} \ldots \sum_{f_m=1}^{k_m} c_{f_1,\ldots,f_m} \prod_{l=1}^{m} v_{i_l,f_l}^{(l)}$$

PARAFAC vs. TD

- PARAFAC

$$\hat{y}(i_1, \ldots, i_m) := \sum_{f=1}^{k} v_{i_1,f}^{(1)} \ldots v_{i_m,f}^{(m)} = \sum_{f=1}^{k} \prod_{l=1}^{m} v_{i_l,f}^{(l)}$$

- TD

$$\hat{y}(i_1, \ldots, i_m) := \sum_{f_1=1}^{k_1} \ldots \sum_{f_m=1}^{k_m} c_{f_1,\ldots,f_m}\, v_{i_1,f_1}^{(1)} \ldots v_{i_m,f_m}^{(m)} = \sum_{f_1=1}^{k_1} \ldots \sum_{f_m=1}^{k_m} c_{f_1,\ldots,f_m} \prod_{l=1}^{m} v_{i_l,f_l}^{(l)}$$

- TD is more general, as C is free.

PARAFAC vs. TD

- PARAFAC

$$\hat{y}(i_1, \ldots, i_m) := \sum_{f=1}^{k} v^{(1)}_{i_1,f} \ldots v^{(m)}_{i_m,f} = \sum_{f=1}^{k} \prod_{l=1}^{m} v^{(l)}_{i_l,f}$$

- TD

$$\hat{y}(i_1, \ldots, i_m) := \sum_{f_1=1}^{k_1} \ldots \sum_{f_m=1}^{k_m} c_{f_1,\ldots,f_m}\, v^{(1)}_{i_1,f_1} \ldots v^{(m)}_{i_m,f_m} = \sum_{f_1=1}^{k_1} \ldots \sum_{f_m=1}^{k_m} c_{f_1,\ldots,f_m} \prod_{l=1}^{m} v^{(l)}_{i_l,f_l}$$

- TD is more general, as C is free.
- Computational complexity:
  - PARAFAC: $\mathcal{O}(k\, m)$.
  - TD: $\mathcal{O}(k^m)$ if $k_1 = \ldots = k_m =: k$.

Tensor Factorization as Machine Learning Models

- PARAFAC and TD model m-ary interactions directly.
- PARAFAC and TD have problems when the number of observations for some levels is small:
  - E.g. assume that there are no observations for a level l, then for the estimated factors $\mathbf{v}_l = 0$ (in case of L2 regularization) and thus all predictions involving this level will always be 0 as well (for PARAFAC and TD).
  - Similar problems can occur if the number of observations of a level is small.

Tensor Factorization as Machine Learning Models

- PARAFAC and TD model m-ary interactions directly.
- PARAFAC and TD have problems when the number of observations for some levels is small:
  - E.g. assume that there are no observations for a level l, then for the estimated factors $\mathbf{v}_l = 0$ (in case of L2 regularization) and thus all predictions involving this level will always be 0 as well (for PARAFAC and TD).
  - Similar problems can occur if the number of observations of a level is small.
- Standard L2-regularization alone cannot solve this problem.

Tensor Factorization as Machine Learning Models

- PARAFAC and TD model m-ary interactions directly.
- PARAFAC and TD have problems when the number of observations for some levels is small:
  - E.g. assume that there are no observations for a level l, then for the estimated factors $\mathbf{v}_l = 0$ (in case of L2 regularization) and thus all predictions involving this level will always be 0 as well (for PARAFAC and TD).
  - Similar problems can occur if the number of observations of a level is small.
- Standard L2-regularization alone cannot solve this problem.
- If a m-ary interaction cannot be estimated reliably, often a lower-level interaction (e.g. $(m-1)$-ary) can be estimated reliably.

TF with Lower-level Interactions

Model equation of m-ary tensor factorization with nested lower-level interactions

$$\hat{y}^{\text{LLTF}}(i_1, \ldots, i_m) := c + \sum_{l=1}^{m} w_{i_l}^{(l)} + \sum_{l_1=1}^{m} \sum_{l_2>l_1}^{m} \hat{y}^{\text{TF}}(i_{l_1}, i_{l_2}) + \ldots + \hat{y}^{\text{TF}}(i_1, \ldots, i_m)$$

[e.g. Rendle et al. 2010; Cai et al. 2011]

TF with Lower-level Interactions

Model equation of m-ary tensor factorization with nested lower-level interactions

$$\hat{y}^{\mathrm{LLTF}}(i_1, \ldots, i_m) := c + \sum_{l=1}^{m} w_{i_l}^{(l)} + \sum_{l_1=1}^{m} \sum_{l_2 > l_1}^{m} \hat{y}^{\mathrm{TF}}(i_{l_1}, i_{l_2}) + \ldots + \hat{y}^{\mathrm{TF}}(i_1, \ldots, i_m)$$

Model parameters

$$c \in \mathbb{R}, \quad \mathbf{w}^{(l)} \in \mathbb{R}^{|I_l|}, \quad \ldots, \quad V^{(l)} \in \mathbb{R}^{|I_l| \times k}$$

[e.g. Rendle et al. 2010; Cai et al. 2011]

TF with Lower-level Interactions

Model equation of m-ary tensor factorization with nested lower-level interactions

$$\hat{y}^{\text{LLTF}}(i_1, \ldots, i_m) := c + \sum_{l=1}^{m} w_{i_l}^{(l)} + \sum_{l_1=1}^{m} \sum_{l_2>l_1}^{m} \hat{y}^{\text{TF}}(i_{l_1}, i_{l_2}) + \ldots + \hat{y}^{\text{TF}}(i_1, \ldots, i_m)$$

Model parameters

$$c \in \mathbb{R}, \quad \mathbf{w}^{(l)} \in \mathbb{R}^{|I_l|}, \quad \ldots, \quad V^{(l)} \in \mathbb{R}^{|I_l| \times k}$$

- Estimating a lower level effect (e.g. a pairwise one) reliably is easier than estimating a higher level one.
- Often lower level effects can explain the data sufficiently and higher level ones can be dropped completely.

[e.g. Rendle et al. 2010; Cai et al. 2011]

Outline

Standard Fitting Algorithms

Standard algorithms assume:

- Y is observed completely, i.e. for all combinations $(i_1, \dots, i_m) \in I_1 \times \dots \times I_m$, $y_{i_1,\dots,i_m}$ is known.
  - Missing values are imputed.
- Optimization is done with respect to least squares:

$$\operatorname*{argmin}_{\Theta} \sum_{(i_1,\dots,i_m)\in I_1\times\dots\times I_m} (y_{i_1,\dots,i_m} - \hat{y}_{i_1,\dots,i_m})^2$$

- No regularization/ prior assumptions.

Standard Fitting Algorithms

Standard algorithms assume:

- Y is observed completely, i.e. for all combinations $(i_1, \ldots, i_m) \in I_1 \times \ldots \times I_m$, $y_{i_1,\ldots,i_m}$ is known.
  - Missing values are imputed.
  - *In ML problems most elements are missing (often* $> 99.9\%$*).*
- Optimization is done with respect to least squares:

$$\underset{\Theta}{\operatorname{argmin}} \sum_{(i_1,\ldots,i_m) \in I_1 \times \ldots \times I_m} (y_{i_1,\ldots,i_m} - \hat{y}_{i_1,\ldots,i_m})^2$$

  - *ML: Other losses are also of interest, e.g. classification, ranking, ...*
- No regularization/ prior assumptions.
  - *ML: Prior knowledge should be included.*

Example: Higher-Order SVD (HOSVD)

HOSVD is such an approximative fitting algorithm:

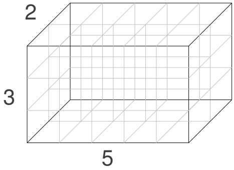
- Loss: Least-squares loss without regularization; no missing value treatment.
- Model: Tucker decomposition
- Algorithm:
  - For each mode l
    - Unfold Y to matrix form.
    - Compute SVD.
    - $V^{(l)}$ are the left singular vectors of the SVD.
  - Compute core tensor
    $C = Y \times_1 (V^{(1)})^T \times_2 (V^{(2)})^T \times_3 \ldots \times_m (V^{(m)})^T$.

[Tucker 66, Lathauwer et al. 2000]

Example: Higher-Order SVD (HOSVD)

HOSVD is such an approximative fitting algorithm:

- Loss: Least-squares loss without regularization; no missing value treatment.
- Model: Tucker decomposition
- Algorithm:
  - For each mode l
    - Unfold Y to matrix form.
    - Compute SVD.
    - $V^{(l)}$ are the left singular vectors of the SVD.
  - Compute core tensor
    $C = Y \times_1 (V^{(1)})^T \times_2 (V^{(2)})^T \times_3 \ldots \times_m (V^{(m)})^T$.

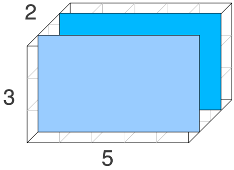


[Tucker 66, Lathauwer et al. 2000]

Example: Higher-Order SVD (HOSVD)

HOSVD is such an approximative fitting algorithm:

- Loss: Least-squares loss without regularization; no missing value treatment.
- Model: Tucker decomposition
- Algorithm:
  - For each mode l
    - Unfold Y to matrix form.
    - Compute SVD.
    - $V^{(l)}$ are the left singular vectors of the SVD.
  - Compute core tensor
    $C = Y \times_1 (V^{(1)})^T \times_2 (V^{(2)})^T \times_3 \ldots \times_m (V^{(m)})^T$.

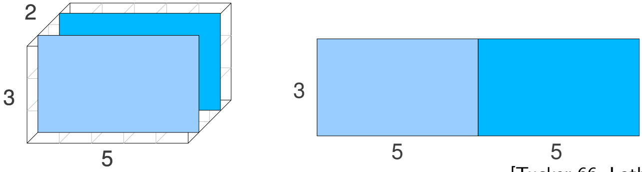


[Tucker 66, Lathauwer et al. 2000]

Example: Higher-Order SVD (HOSVD)

HOSVD is such an approximative fitting algorithm:

- Loss: Least-squares loss without regularization; no missing value treatment.
- Model: Tucker decomposition
- Algorithm:
  - For each mode l
    - Unfold Y to matrix form.
    - Compute SVD.
    - $V^{(l)}$ are the left singular vectors of the SVD.
  - Compute core tensor
    $C = Y \times_1 (V^{(1)})^T \times_2 (V^{(2)})^T \times_3 \ldots \times_m (V^{(m)})^T.$

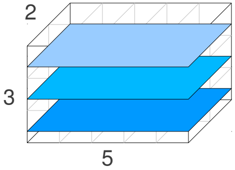


[Tucker 66, Lathauwer et al. 2000]

Example: Higher-Order SVD (HOSVD)

HOSVD is such an approximative fitting algorithm:

- Loss: Least-squares loss without regularization; no missing value treatment.
- Model: Tucker decomposition
- Algorithm:
  - For each mode l
    - Unfold Y to matrix form.
    - Compute SVD.
    - $V^{(l)}$ are the left singular vectors of the SVD.
  - Compute core tensor $C = Y \times_1 (V^{(1)})^T \times_2 (V^{(2)})^T \times_3 \ldots \times_m (V^{(m)})^T$.

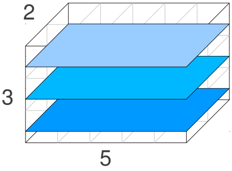


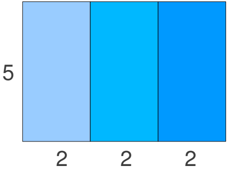
[Tucker 66, Lathauwer et al. 2000]

Example: Higher-Order SVD (HOSVD)

HOSVD is such an approximative fitting algorithm:

- Loss: Least-squares loss without regularization; no missing value treatment.
- Model: Tucker decomposition
- Algorithm:
  - For each mode l
    - Unfold Y to matrix form.
    - Compute SVD.
    - $V^{(l)}$ are the left singular vectors of the SVD.
  - Compute core tensor
    $C = Y \times_1 (V^{(1)})^T \times_2 (V^{(2)})^T \times_3 \ldots \times_m (V^{(m)})^T$.



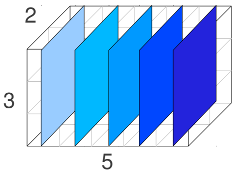


[Tucker 66, Lathauwer et al. 2000]

Example: Higher-Order SVD (HOSVD)

HOSVD is such an approximative fitting algorithm:

- Loss: Least-squares loss without regularization; no missing value treatment.
- Model: Tucker decomposition
- Algorithm:
  - For each mode l
    - Unfold Y to matrix form.
    - Compute SVD.
    - $V^{(l)}$ are the left singular vectors of the SVD.
  - Compute core tensor $C = Y \times_1 (V^{(1)})^T \times_2 (V^{(2)})^T \times_3 \ldots \times_m (V^{(m)})^T$.

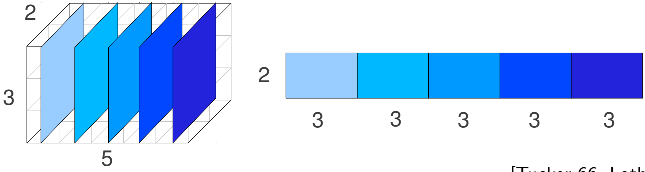


[Tucker 66, Lathauwer et al. 2000]

Example: Higher-Order SVD (HOSVD)

HOSVD is such an approximative fitting algorithm:

- Loss: Least-squares loss without regularization; no missing value treatment.
- Model: Tucker decomposition
- Algorithm:
  - For each mode l
    - Unfold Y to matrix form.
    - Compute SVD.
    - $V^{(l)}$ are the left singular vectors of the SVD.
  - Compute core tensor
    $C = Y \times_1 (V^{(1)})^T \times_2 (V^{(2)})^T \times_3 \dots \times_m (V^{(m)})^T$.



[Tucker 66, Lathauwer et al. 2000]

Example: Higher-Order SVD (HOSVD)

HOSVD is such an approximative fitting algorithm:

- Loss: Least-squares loss without regularization; no missing value treatment.
- Model: Tucker decomposition
- Algorithm:
  - For each mode l
    - Unfold Y to matrix form.
    - Compute SVD.
    - $V^{(l)}$ are the left singular vectors of the SVD.
  - Compute core tensor
    $C = Y \times_1 (V^{(1)})^T \times_2 (V^{(2)})^T \times_3 \ldots \times_m (V^{(m)})^T$.

Additional Alternating Least-Squares (ALS) steps can improve the fit.

[Tucker 66, Lathauwer et al. 2000]

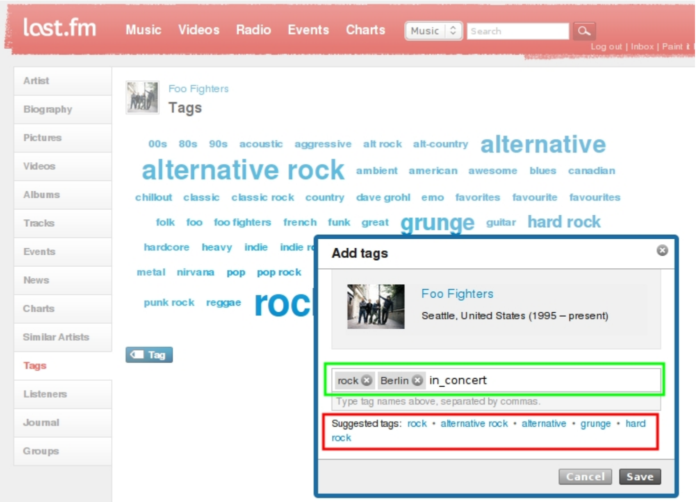
Machine Learning with TF Models

- Optimize only w.r.t. observed elements of Y.
  - Comparable to MF: *Weighted Low-Rank Approximations* [Srebro et al. 2003]
- Choose loss/ likelihood according to the target variables/ task.
  - E.g. logit for classification, pairwise classification for ranking, etc.
- Add priors / regularization to model parameters.
  - E.g. L2/ Gaussian priors.
- Model lower-level interactions.
  - E.g. add factorized pairwise interactions [Rendle et al. 2010]

TF models are multilinear $\Rightarrow$ simple SGD or ALS algorithms can be used for optimization.

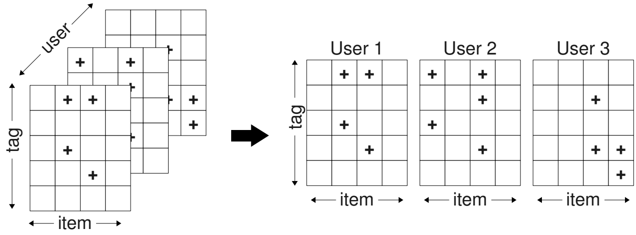
Outline

Personalized Tag Recommendation



Task: Recommend a user a (personalized) list of tags for a specific item. [Hotho et al., 2006]

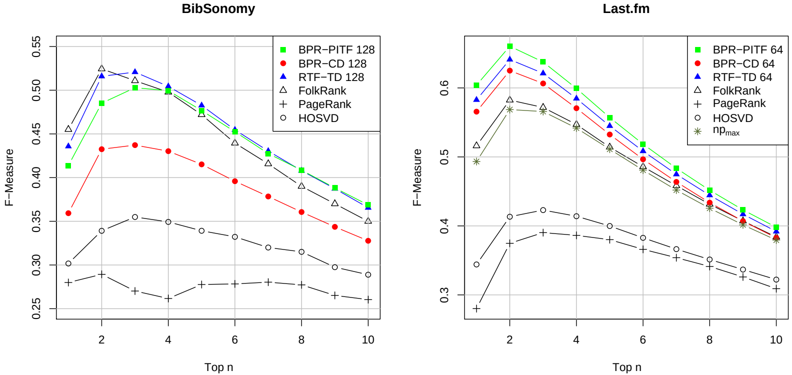
Personalized Tag Recommendation



- U ... users
- I ... items
- T ... tags
- $S \subseteq U \times I \times T$ ... observed tags
- $P_S = \{(u,i) | \exists t \in T : (u,i,t) \in S\}$ ... observed tagging posts

[Hotho et al., 2006]

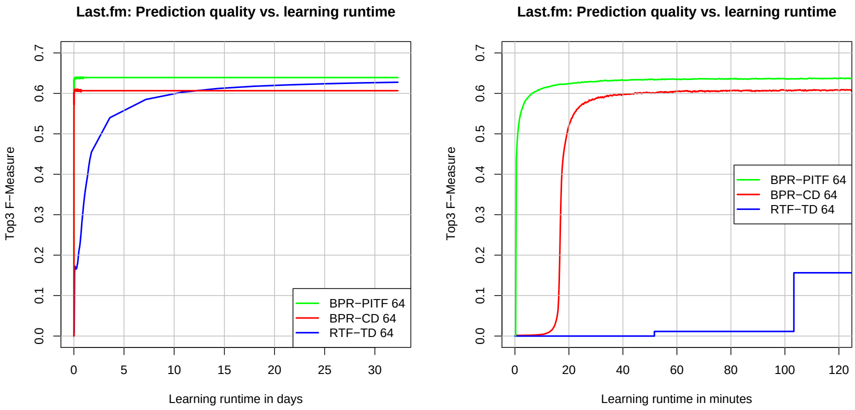
Evaluation: Prediction Quality



- adapted PageRank for tag recommendation/ Folkrank [Hotho et al. 2006]
- HOSVD: TD for least squares, no missing values, no reg. [Symeonidis et al. 2008]
- RTF-TD: TD model optimized for regularized ranking [Rendle et al. 2009]
- BPR-PITF, BPR-CD: PITF/ PARAFAC model optimized for regularized ranking [Rendle et al. 2010]

[Rendle et. al 2010]

Evaluation: Learning Runtime



[Rendle et. al 2010]

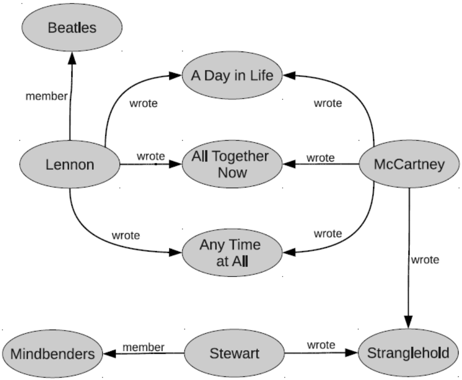
ECML/PKDD Discovery Challenge 2009

Rank	Method	Top-5 F-Measure
1	**BPR-PITF** + adaptive list size	0.35594
-	**BPR-PITF** *(not submitted)*	*0.345*
2	Relational Classification [Marinho et al. 09]	0.33185
3	Content-based [Lipczak et al. 09]	0.32461
4	Content-based [Zhang et al. 09]	0.32230
5	Content-based [Ju and Hwang 09]	0.32134
6	Personomy translation [Wetzker et al. 09]	0.32124
...	...	...

Task 2: ECML/ PKDD Challenge 2009,
http://www.kde.cs.uni-kassel.de/ws/dc09/results

[Rendle et. al 2010]

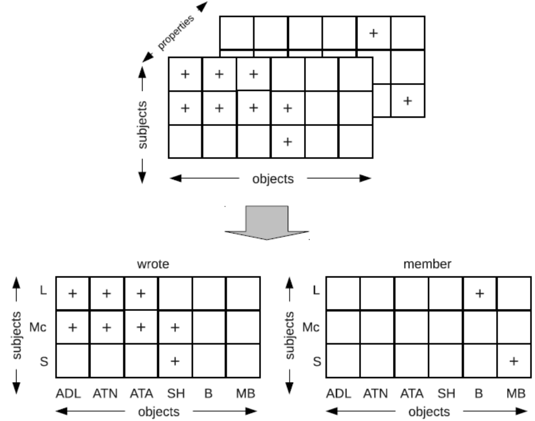
Querying Incomplete RDF-Graphs



- **Task:** Answer queries about subject-predicate-pairs. E.g. What is McCartney member of?

[Franz et al. 2009, Drumond et al. 2012]

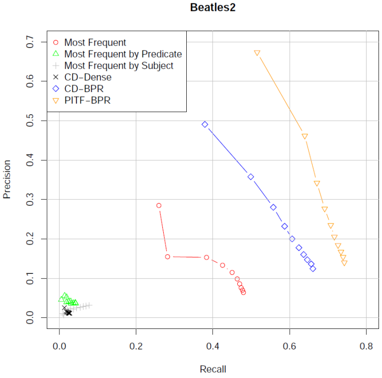
Querying Incomplete RDF-Graphs

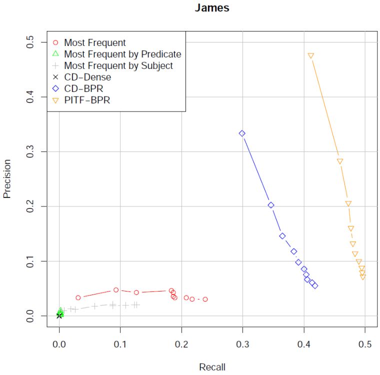


An RDF-Graph can be expressed as a function over three categorical domains: $y : S \times P \times O \rightarrow \{0, 1\}$

[Franz et al. 2009, Drumond et al. 2012]

Prediction Quality





- CD Dense: PARAFAC optimized for least-squares, no missing values, no reg.
- CD-BPR: PARAFAC optimized for regularized ranking.
- PITF-BPR: PITF (pairwise interactions) optimized for regularized ranking.

[Drumond et al. 2012]

Other Applications: Examples

- Multiverse Recommendation [Karatzoglou et al. 2010]
  - Task: Context-aware Rating prediction.
  - Model: Tucker Decomposition.
  - Missing values are handled.
  - Loss: task dependent, e.g. MAE, RMSE.
  - Regularization: L1, L2.
  - Algorithm: Stochastic Gradient Descent (SGD).
- CubeSVD [Sun et al. 2005]
  - Task: Clickthrough prediction.
  - Approach: HOSVD.

Outline

Summary

- Prediction functions with m categorical variables can be modeled with tensor factorization.
- Parallel Factor Analysis (PARAFAC) generalizes matrix factorization to m modes.
- Tucker Decomposition allows a free core tensor. (High computational complexity!)
- Lower-order interactions, e.g. pairwise ones should be integrated for better prediction quality in sparse settings.
- For learning: missing values, loss/likelihood and regularization/ priors should be considered.

Summary

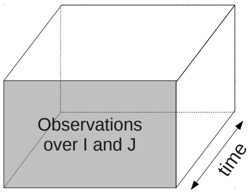
- Prediction functions with m categorical variables can be modeled with tensor factorization.
- Parallel Factor Analysis (PARAFAC) generalizes matrix factorization to m modes.
- Tucker Decomposition allows a free core tensor. (High computational complexity!)
- Lower-order interactions, e.g. pairwise ones should be integrated for better prediction quality in sparse settings.
- For learning: missing values, loss/likelihood and regularization/ priors should be considered.

Problem: Only categorical variables can be handled.

Outline

Time-Aware: Problem Setting

- 3 predictor variables:
  - two variables of categorical domain I and J.
  - one numerical variable (time), $t \in \mathbb{R}$.
- Target y: Real-valued (regression), binary (classification), scores (ranking).
- Supervised task: set of observations $S = \{(i, j, t, y), \ldots\}$
- Modelling: function $\hat{y} : I \times J \times \mathbb{R} \rightarrow \mathcal{Y}$.



Tensor Factorization

1. Discretize time variable, e.g. by binning. $\Rightarrow$ 3 cat. domains: I, J, T.

$$b : \mathbb{R} \to T, \quad \text{e.g. } b(t) := \lfloor t/(24 * 60 * 60) \rfloor$$

[Xiong et al. 2010]

Tensor Factorization

1. Discretize time variable, e.g. by binning. $\Rightarrow$ 3 cat. domains: I, J, T.

$$b : \mathbb{R} \to T, \quad \text{e.g. } b(t) := \lfloor t/(24 * 60 * 60) \rfloor$$

2. Apply tensor factorization, e.g. Tucker Decomposition, PARAFAC.

$$\hat{y}(i, j, t) := \sum_{f=1}^{k} v^{I}_{i,f} \, v^{J}_{j,f} \, v^{T}_{b(t),f}$$

[Xiong et al. 2010]

Tensor Factorization

1. Discretize time variable, e.g. by binning. $\Rightarrow$ 3 cat. domains: I, J, T.

$$b : \mathbb{R} \to T, \quad \text{e.g. } b(t) := \lfloor t/(24 * 60 * 60) \rfloor$$

2. Apply tensor factorization, e.g. Tucker Decomposition, PARAFAC.

$$\hat{y}(i,j,t) := \sum_{f=1}^{k} v^{I}_{i,f} \, v^{J}_{j,f} \, v^{T}_{b(t),f}$$

3. Smooth time factors V^T, s.th. nearby points in time have similar factors. E.g. by regularization:

$$v^{T}_{t+1,f} \sim \mathcal{N}(v^{T}_{t,f}, 1/\lambda_T), \quad \forall t \in T, f \in \{1, \ldots, k\}$$

[Xiong et al. 2010]

Tensor Factorization

1. Discretize time variable, e.g. by binning. $\Rightarrow$ 3 cat. domains: I, J, T.

$$b : \mathbb{R} \to T, \quad \text{e.g. } b(t) := \lfloor t/(24 * 60 * 60) \rfloor$$

2. Apply tensor factorization, e.g. Tucker Decomposition, PARAFAC.

$$\hat{y}(i,j,t) := \sum_{f=1}^{k} v^I_{i,f}\, v^J_{j,f}\, v^T_{b(t),f}$$

3. Smooth time factors V^T, s.th. nearby points in time have similar factors. E.g. by regularization:

$$v^T_{t+1,f} \sim \mathcal{N}(v^T_{t,f}, 1/\lambda_T), \quad \forall t \in T, f \in \{1, \ldots, k\}$$

For learning/ inference, e.g. a MCMC sampler can be used.

[Xiong et al. 2010]

Time-Aware Matrix Factorization

Time-Aware Matrix Factorization

$$\hat{y}(i,j,t) := \sum_{f=1}^{k} w_{i,f}(t)\, h_{j,f}(t)$$

where the factor matrices H and W depend on the time t:

$$W : \mathbb{R} \to \mathbb{R}^{|I| \times k}, \quad H : \mathbb{R} \to \mathbb{R}^{|J| \times k}$$

[Koren 2009]

Time-Aware Matrix Factorization

Modeling time dependent factors, e.g. for W:

- Constant

$$w_{i,f}(t) := \tilde{w}_{i,f}, \quad \tilde{W} \in \mathbb{R}^{|I| \times k}$$

[Koren 2009]

Time-Aware Matrix Factorization

Modeling time dependent factors, e.g. for W:

- Constant

$$w_{i,f}(t) := \tilde{w}_{i,f}, \quad \tilde{W} \in \mathbb{R}^{|I| \times k}$$

- Linear

$$w_{i,f}(t) := \tilde{w}_{i,f} + z_{i,f}\, t, \quad \tilde{W} \in \mathbb{R}^{|I| \times k}, Z \in \mathbb{R}^{|I| \times k}$$

[Koren 2009]

Time-Aware Matrix Factorization

Modeling time dependent factors, e.g. for W:

- Constant

$$w_{i,f}(t) := \tilde{w}_{i,f}, \quad \tilde{W} \in \mathbb{R}^{|I| \times k}$$

- Linear

$$w_{i,f}(t) := \tilde{w}_{i,f} + z_{i,f}\, t, \quad \tilde{W} \in \mathbb{R}^{|I| \times k}, Z \in \mathbb{R}^{|I| \times k}$$

- Binning with function b

$$w_{i,f}(t) := \tilde{w}_{i,f,b(t)}, \quad \tilde{W} \in \mathbb{R}^{|I| \times k \times |\mathrm{img}(b)|}$$

[Koren 2009]

Time-Aware Matrix Factorization

Modeling time dependent factors, e.g. for W:

- Constant

$$w_{i,f}(t) := \tilde{w}_{i,f}, \quad \tilde{W} \in \mathbb{R}^{|I| \times k}$$

- Linear

$$w_{i,f}(t) := \tilde{w}_{i,f} + z_{i,f}\, t, \quad \tilde{W} \in \mathbb{R}^{|I| \times k}, Z \in \mathbb{R}^{|I| \times k}$$

- Binning with function b

$$w_{i,f}(t) := \tilde{w}_{i,f,b(t)}, \quad \tilde{W} \in \mathbb{R}^{|I| \times k \times |\mathrm{img}(b)|}$$

- Spline with m_i predefined control points at position $t_{i,1}, \ldots, t_{i,m}$

$$w_{i,f}(t) := \frac{\sum_{l=1}^{m_i} \tilde{w}_{i,f,l} \exp(-\gamma |t - t_{i,l}|)}{\sum_{l=1}^{m_i} \exp(-\gamma |t - t_{i,l}|)}, \quad \tilde{W} \in \mathbb{R}^{|I| \times k \times m_i}$$

[Koren 2009]

Time-Aware Matrix Factorization

Modeling time dependent factors, e.g. for W:

- Constant

$$w_{i,f}(t) := \tilde{w}_{i,f}, \quad \tilde{W} \in \mathbb{R}^{|I| \times k}$$

- Linear

$$w_{i,f}(t) := \tilde{w}_{i,f} + z_{i,f}\, t, \quad \tilde{W} \in \mathbb{R}^{|I| \times k}, Z \in \mathbb{R}^{|I| \times k}$$

- Binning with function b

$$w_{i,f}(t) := \tilde{w}_{i,f,b(t)}, \quad \tilde{W} \in \mathbb{R}^{|I| \times k \times |\mathrm{img}(b)|}$$

- Spline with m_i predefined control points at position $t_{i,1}, \ldots, t_{i,m}$

$$w_{i,f}(t) := \frac{\sum_{l=1}^{m_i} \tilde{w}_{i,f,l} \exp(-\gamma |t - t_{i,l}|)}{\sum_{l=1}^{m_i} \exp(-\gamma |t - t_{i,l}|)}, \quad \tilde{W} \in \mathbb{R}^{|I| \times k \times m_i}$$

- Linear combinations of the functions above.

[Koren 2009]

Time-Aware Matrix Factorization

Choices for the *timeSVD++* model for the Netflix challenge:

- User factors W: linear combination of
  - linear effect
  - binning with bin size 1
- Item factors H:
  - constant
- Additional (time-unaware) implicit indicators (from SVD++ [Koren, 2008])

[Koren 2009]

Time-Aware Matrix Factorization

Choices for the *timeSVD++* model for the Netflix challenge:

- User factors W: linear combination of
  - linear effect
  - binning with bin size 1
- Item factors H:
  - constant
- Additional (time-unaware) implicit indicators (from SVD++ [Koren, 2008])

For learning, e.g. a SGD algorithm can be used.

[Koren 2009]

Comparison

- Time-aware MF with binning (TAMF) and tensor factorization with discretization (TF) treat the time variable similarly:

$$\hat{y}^{TAMF}(i,j,t) := \sum_{f=1}^{k} w_{i,f,b(t)}\, h_{j,f}$$

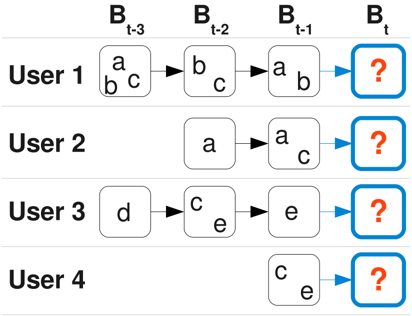
$$\hat{y}^{TF}(i,j,t) := \sum_{f=1}^{k} w_{i,f}\, h_{j,f}\, z_{b(t),f}$$

- Main difference:
  - In tensor factorization, the (i,t)-interaction is factorized.
  - In time-aware MF, the (i,t)-interaction is modeled unfactorized.

Discussion

- Binning and splines cannot make use of time for future events.
  - Future bins are empty and variables cannot be estimated.
  - Variables in (future) control points of splines cannot be estimated.
- Seasonal time indicators can help, e.g. weekday, holiday, Christmas, etc,
- Other approach: use qualitative/ sequential information

Sequential Prediction



- **Task:** Which items will be selected next?

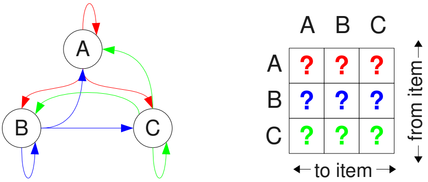
[e.g. Zimdars et al. 2001, Rendle et al. 2010]

Markov Chains

Markov chain of order 1:

$$p(j_t | l_{t-1})$$

- t is a sequential index.
- l_{t-1} is the item selected previously.
- The Markov chain is defined by a transition matrix $A \in \mathbb{R}^{|J| \times |J|}$.

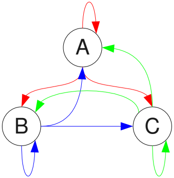


Markov Chains

Markov chain of order 1:

$$p(j_t | l_{t-1})$$

- t is a sequential index.
- l_{t-1} is the item selected previously.
- The Markov chain is defined by a transition matrix $A \in \mathbb{R}^{|J| \times |J|}$.
- Model is (weakly) personalized by taking the last item selected by a user into account.



	A	B	C
A	?	?	?
B	?	?	?
C	?	?	?

← to item →

← from item →

Factorized Personalized Markov Chain

Model equation

$$\hat{y}(i, j, t) := \hat{z}(i, j, s(i, t))$$

where $s(i, t)$ is the previously (w.r.t. t) selected entity (by i).

- $\hat{z}$ can be modeled by TD, PARAFAC, PITF, ...
- For product recommendation i is the user and j the current item.

[Rendle et al. 2010]

Factorized Personalized Markov Chain

Model equation

$$\hat{y}(i, j, t) := \hat{z}(i, j, s(i, t))$$

where $s(i, t)$ is the previously (w.r.t. t) selected entity (by i).

- $\hat{z}$ can be modeled by TD, PARAFAC, PITF, ...
- For product recommendation i is the user and j the current item.
- If a set of items can be selected previously, one can average over this set:

$$\hat{y}(i, j, t) := \frac{1}{|s(i, t)|} \sum_{l \in s(i,t)} \hat{z}(i, j, l)$$

For learning, e.g. a SGD algorithm can be used.

[Rendle et al. 2010]

Outline

Tensor Factorization

Time-aware Factorization Models
- Models
- Summary

Factorization Machines

Summary

- Time can be taken into account by:
  - Discretization and applying Tensor Factorization.
  - Time-variant factors, e.g. binning, linear effects, splines, ...
  - Sequential indicators, e.g. last item selected.
- With time-variables, the dataset split should be considered:
  - Random split: absolute time can be modeled.
  - Time split: binning not effective, time transformation that are predictive for future points in time should be chosen; e.g. seasonal or sequential.

Outline

Motivation

All the presented factorization models work empirically very well, but:

- For each new problem a new model, a new learning algorithm and implementation is necessary.
- For some of the models there are dozens of improved learning algorithms proposed (that work only with this particular model).
- For non-experts in factorization models this is not applicable.
- How does this relate to standard models?

Data and Variable Representation

Many standard ML approaches work with real valued input data (a *design matrix*). It allows to represent, e.g.:

- any number of variables
- categorical domains by using dummy indicator variables
- numerical domains
- set-categorical domains by using dummy indicator variables

Using this representation allows to apply a wide variety of standard models (e.g. linear regression, SVM, etc.).

Data and Variable Representation: Example

User	Movie	Rating
Alice	Titanic	5
Alice	Notting Hill	3
Alice	Star Wars	1
Bob	Star Wars	4
Bob	Star Trek	5
Charlie	Titanic	1
Charlie	Star Wars	5
...	...	...

2 categorical variables

Data and Variable Representation: Example

User	Movie	Rating
Alice	Titanic	5
Alice	Notting Hill	3
Alice	Star Wars	1
Bob	Star Wars	4
Bob	Star Trek	5
Charlie	Titanic	1
Charlie	Star Wars	5
...	...	...

Feature vector $\mathbf{x}$ / Target y

	A	B	C	...	TI	NH	SW	ST	...	Target y	
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	1	$y^{(3)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	4	$y^{(4)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	5	$y^{(5)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	1	$y^{(6)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	5	$y^{(7)}$
	User				Movie						

2 categorical variables

$|U| + |I|$ real valued variables

Problem Setting

- Predictor variables: p variables of real-valued domain $X_1, \ldots, X_p \in \mathbb{R}$.
- Target y: Real-valued (regression), binary (classification), scores (ranking).
- Supervised task: set of observations $S = \{(x_1, \ldots, x_p, y), \ldots\}$

Problem Setting

- Predictor variables: p variables of real-valued domain $X_1, \ldots, X_p \in \mathbb{R}$.
- Target y: Real-valued (regression), binary (classification), scores (ranking).
- Supervised task: set of observations $S = \{(x_1, \ldots, x_p, y), \ldots\}$

This is the most common machine learning task.

Outline

Standard Machine Learning Models

- Categorical variables can be represented with real-valued ones.
- There are many well-studied standard ML models that can work with real-valued variables.
- Why shouldn't we work with them? Why do we need factorization models?

Linear Regression

- Let $\mathbf{x} \in \mathbb{R}^p$ be an input vector with p predictor variables.
- Model equation:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i \, x_i$$

- Model parameters:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^p$$

$\mathcal{O}(p)$ model parameters.

Polynomial Regression

- Let $\mathbf{x} \in \mathbb{R}^p$ be an input vector with p predictor variables.
- Model equation (degree 2):

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i x_i + \sum_{i=1}^{p} \sum_{j \geq i}^{p} w_{i,j}\, x_i\, x_j$$

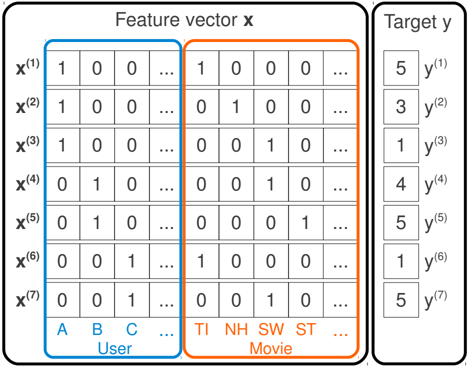
- Model parameters:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^p, \quad \mathbf{W} \in \mathbb{R}^{p \times p}$$

$\mathcal{O}(p^2)$ model parameters.

Application to Large Categorical Domains

User	Movie	Rating
Alice	Titanic	5
Alice	Notting Hill	3
Alice	Star Wars	1
Bob	Star Wars	4
Bob	Star Trek	5
Charlie	Titanic	1
Charlie	Star Wars	5
...	...	...



Applying regression models to this data leads to:

Application to Large Categorical Domains

User	Movie	Rating
Alice	Titanic	5
Alice	Notting Hill	3
Alice	Star Wars	1
Bob	Star Wars	4
Bob	Star Trek	5
Charlie	Titanic	1
Charlie	Star Wars	5
...	...	...

Feature vector **x** | Target y

	User: A	B	C	...	Movie: TI	NH	SW	ST	...	Target y	
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	1	$y^{(3)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	4	$y^{(4)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	5	$y^{(5)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	1	$y^{(6)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	5	$y^{(7)}$

Applying regression models to this data leads to:

Linear regression: $\hat{y}(\mathbf{x}) = w_0 + w_u + w_i$

Application to Large Categorical Domains

User	Movie	Rating
Alice	Titanic	5
Alice	Notting Hill	3
Alice	Star Wars	1
Bob	Star Wars	4
Bob	Star Trek	5
Charlie	Titanic	1
Charlie	Star Wars	5
...	...	...

Feature vector **x** / Target y

	User: A	User: B	User: C	User: ...	Movie: TI	Movie: NH	Movie: SW	Movie: ST	Movie: ...	Target y	
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	1	$y^{(3)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	4	$y^{(4)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	5	$y^{(5)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	1	$y^{(6)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	5	$y^{(7)}$

Applying regression models to this data leads to:

Linear regression: $\hat{y}(\mathbf{x}) = w_0 + w_u + w_i$

Polynomial regression: $\hat{y}(\mathbf{x}) = w_0 + w_u + w_i + w_{u,i}$

Application to Large Categorical Domains

User	Movie	Rating
Alice	Titanic	5
Alice	Notting Hill	3
Alice	Star Wars	1
Bob	Star Wars	4
Bob	Star Trek	5
Charlie	Titanic	1
Charlie	Star Wars	5
...	...	...

Feature vector **x** | Target y

	User: A	B	C	...	Movie: TI	NH	SW	ST	...	Target y	
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	1	$y^{(3)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	4	$y^{(4)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	5	$y^{(5)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	1	$y^{(6)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	5	$y^{(7)}$

Applying regression models to this data leads to:

Linear regression: $\hat{y}(\mathbf{x}) = w_0 + w_u + w_i$

Polynomial regression: $\hat{y}(\mathbf{x}) = w_0 + w_u + w_i + w_{u,i}$

Matrix factorization (with biases): $\hat{y}(u,i) = w_0 + w_u + h_i + \langle \mathbf{w}_u, \mathbf{h}_i \rangle$

Application to Large Categorical Domains

For the recommender data of the example:

- Linear regression has no user-item interaction.

Application to Large Categorical Domains

For the recommender data of the example:

- Linear regression has no user-item interaction.
  - $\Rightarrow$ Linear regression is not expressive enough.

Application to Large Categorical Domains

For the recommender data of the example:

- Linear regression has no user-item interaction.
  - $\Rightarrow$ Linear regression is not expressive enough.

- Polynomial regression includes pairwise interactions but cannot estimate them from the data.

Application to Large Categorical Domains

For the recommender data of the example:

- Linear regression has no user-item interaction.
  - $\Rightarrow$ Linear regression is not expressive enough.
- Polynomial regression includes pairwise interactions but cannot estimate them from the data.
  - $n \ll p^2$: number of cases is much smaller than number of model parameters.

Application to Large Categorical Domains

For the recommender data of the example:

- Linear regression has no user-item interaction.
  - $\Rightarrow$ Linear regression is not expressive enough.

- Polynomial regression includes pairwise interactions but cannot estimate them from the data.
  - $n \ll p^2$: number of cases is much smaller than number of model parameters.
  - Max.-likelihood estimator for a pairwise effect is:

$$w_{i,j} = \begin{cases} y - w_0 - w_i - w_u, & \text{if } (i,j,y) \in S. \\ \text{not defined}, & \text{else} \end{cases}$$

Application to Large Categorical Domains

For the recommender data of the example:

- Linear regression has no user-item interaction.
  - $\Rightarrow$ Linear regression is not expressive enough.

- Polynomial regression includes pairwise interactions but cannot estimate them from the data.
  - $n \ll p^2$: number of cases is much smaller than number of model parameters.
  - Max.-likelihood estimator for a pairwise effect is:

$$w_{i,j} = \begin{cases} y - w_0 - w_i - w_u, & \text{if } (i,j,y) \in S. \\ \text{not defined}, & \text{else} \end{cases}$$

  - Polynomial regression cannot generalize to *any* unobserved pairwise effect.

Factorization Models and Real-valued Variables

- Factorization models work well for categorical variables of large domain.
- Standard Models are more flexible as they allow real-valued predictor variables that can be used for encoding several kind of variables.

Factorization Models and Real-valued Variables

- Factorization models work well for categorical variables of large domain.
- Standard Models are more flexible as they allow real-valued predictor variables that can be used for encoding several kind of variables.
- How can these advantages be combined?

Outline

Factorization Machine (FM)

- Let $\mathbf{x} \in \mathbb{R}^p$ be an input vector with p predictor variables.
- Model equation (degree 2):

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i\, x_i + \sum_{i=1}^{p} \sum_{j>i}^{p} \langle \mathbf{v}_i, \mathbf{v}_j \rangle\, x_i\, x_j$$

- Model parameters:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^p, \quad \mathbf{V} \in \mathbb{R}^{p \times k}$$

[Rendle 2010, Rendle 2012]

Factorization Machine (FM)

- Let $\mathbf{x} \in \mathbb{R}^p$ be an input vector with p predictor variables.
- Model equation (degree 2):

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i\, x_i + \sum_{i=1}^{p} \sum_{j>i}^{p} \langle \mathbf{v}_i, \mathbf{v}_j \rangle\, x_i\, x_j$$

- Model parameters:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^p, \quad \mathbf{V} \in \mathbb{R}^{p \times k}$$

Compared to Polynomial regression:

- Model equation (degree 2):

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i x_i + \sum_{i=1}^{p} \sum_{j \geq i}^{p} w_{i,j}\, x_i\, x_j$$

- Model parameters:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^p, \quad \mathbf{W} \in \mathbb{R}^{p \times p}$$

[Rendle 2010, Rendle 2012]

Factorization Machine (FM)

- Let $\mathbf{x} \in \mathbb{R}^p$ be an input vector with p predictor variables.
- Model equation (degree 2):

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i \, x_i + \sum_{i=1}^{p} \sum_{j>i}^{p} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j$$

- Model parameters:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^p, \quad \mathbf{V} \in \mathbb{R}^{p \times k}$$

[Rendle 2010, Rendle 2012]

Factorization Machine (FM)

- Let $\mathbf{x} \in \mathbb{R}^p$ be an input vector with p predictor variables.
- Model equation (degree 3):

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i\, x_i + \sum_{i=1}^{p} \sum_{j>i}^{p} \langle \mathbf{v}_i, \mathbf{v}_j \rangle\, x_i\, x_j + \sum_{i=1}^{p} \sum_{j>i}^{p} \sum_{l>j}^{p} \sum_{f=1}^{k} v_{i,f}^{(3)}\, v_{j,f}^{(3)}\, v_{l,f}^{(3)}\, x_i\, x_j\, x_l$$

- Model parameters:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^p, \quad \mathbf{V} \in \mathbb{R}^{p \times k}, \quad \mathbf{V}^{(3)} \in \mathbb{R}^{p \times k}$$

[Rendle 2010, Rendle 2012]

Factorization Machines: Discussion

- FMs work with real valued input.
- FMs include variable interactions like polynomial regression.
- Model parameters for interactions are factorized.
- Number of model parameters is $\mathcal{O}(k\,p)$ (instead of $\mathcal{O}(p^2)$ for poly. regr.).

Factorization Machines: Discussion

- FMs work with real valued input.
- FMs include variable interactions like polynomial regression.
- Model parameters for interactions are factorized.
- Number of model parameters is $\mathcal{O}(k\,p)$ (instead of $\mathcal{O}(p^2)$ for poly. regr.).
- How are FMs related to the factorization models we have seen so far?

Matrix Factorization and Factorization Machines

Two categorical variables encoded with real valued predictor variables:

Feature vector **x**

	User				Movie				
	A	B	C	...	TI	NH	SW	ST	...
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...

With this data, the FM is identical to MF with biases:

$$\hat{y}(\mathbf{x}) = w_0 + w_u + w_i + \underbrace{\langle \mathbf{v}_u, \mathbf{v}_i \rangle}_{\text{MF}}$$

Tag-Recommendation with Factorization Machines

Three categorical variables encoded with real valued predictor variables:

Feature vector **x**

	User				Song					Tag				
	A	B	C	...	S1	S2	S3	S4	...	T1	T2	T3	T4	...
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	1	0	0	0	...
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0	1	0	0	...
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0	0	0	1	...
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	1	0	...
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	1	0	...
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	1	0	0	0	...
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0	0	0	1	...

With this data, the FM is a tensor factorization model with lower-level interactions (here up to pairwise ones):

$$\hat{y}(\mathbf{x}) := w_0 + w_i + w_u + w_t + \langle \mathbf{v}_u, \mathbf{v}_t \rangle + \langle \mathbf{v}_i, \mathbf{v}_t \rangle + \langle \mathbf{v}_u, \mathbf{v}_i \rangle$$

Time with Factorization Machines

Two categorical variables and time as linear predictor:

Feature vector **x**

	A	B	C	...	TI	NH	SW	ST	...	Time
	User				Movie					Time
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.2
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.6
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.61
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0.3
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0.5
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.1
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.8

The FM model would correspond to:

$$\hat{y}(\mathbf{x}) := w_0 + w_i + w_u + t\, w_{\text{time}} + \langle \mathbf{v}_u, \mathbf{v}_i \rangle + t \langle \mathbf{v}_u, \mathbf{v}_{\text{time}} \rangle + t \langle \mathbf{v}_i, \mathbf{v}_{\text{time}} \rangle$$

Time with Factorization Machines

Two categorical variables and time discretized in bins ($b(t)$):

Feature vector **x**

	User				Movie					Time		
	A	B	C	...	TI	NH	SW	ST	...	T1	T2	T3
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	1	0	0
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0	1	0
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0	1	0
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	1	0	0
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	1	0
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	1	0	0
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0	0	1

With this data, a three-order FM includes the time-aware tensor factorization model described before:

$$\hat{y}(\mathbf{x}) := w_0 + w_i + w_u + w_{b(t)} + \langle \mathbf{v}_u, \mathbf{v}_i \rangle + \langle \mathbf{v}_u, \mathbf{v}_{b(t)} \rangle + \langle \mathbf{v}_i, \mathbf{v}_{b(t)} \rangle + \underbrace{\sum_{f=1}^{k} \mathbf{v}^{(3)}_{u,f}\, \mathbf{v}^{(3)}_{i,f}\, \mathbf{v}^{(3)}_{b(t),f}}_{\text{Time Tensor Factorization Model}}$$

Time with Factorization Machines

Two categorical variables and time discretized in bins ($b(t)$):

Feature vector **x**

	User-Time										Movie				
	AT_1	AT_2	AT_3	BT_1	BT_2	BT_3	CT_1	CT_2	CT_3	...	TI	NH	SW	ST	...
$\mathbf{x}^{(1)}$	1	0	0	0	0	0	0	0	0	...	1	0	0	0	...
$\mathbf{x}^{(2)}$	0	1	0	0	0	0	0	0	0	...	0	1	0	0	...
$\mathbf{x}^{(3)}$	0	1	0	0	0	0	0	0	0	...	0	0	1	0	...
$\mathbf{x}^{(4)}$	0	0	0	1	0	0	0	0	0	...	0	0	1	0	...
$\mathbf{x}^{(5)}$	0	0	0	0	1	0	0	0	0	...	0	0	0	1	...
$\mathbf{x}^{(6)}$	0	0	0	0	0	0	1	0	0	...	1	0	0	0	...
$\mathbf{x}^{(7)}$	0	0	0	0	0	0	0	0	1	...	0	0	1	0	...

With this data, an FM includes the time-aware matrix factorization model with binned user-time interactions:

$$\hat{y}(\mathbf{x}) := w_0 + w_i + w_{u,b(t)} + \underbrace{\langle \mathbf{v}_{u,b(t)}, \mathbf{v}_i \rangle}_{\text{MF with time variant factors}}$$

[Koren, 2009]

SVD++

Feature vector **x**

	User				Movie					Other Movies rated				
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...

With this data, the FM is identical to:

$$\hat{y}(\mathbf{x}) = \overbrace{w_0 + w_u + w_i + \langle \mathbf{v}_u, \mathbf{v}_i \rangle + \frac{1}{\sqrt{|N_u|}} \sum_{l \in N_u} \langle \mathbf{v}_i, \mathbf{v}_l \rangle}^{\text{SVD++}}$$

$$+ \frac{1}{\sqrt{|N_u|}} \sum_{l \in N_u} \left(w_l + \langle \mathbf{v}_u, \mathbf{v}_l \rangle + \frac{1}{\sqrt{|N_u|}} \sum_{l' \in N_u, l' > l} \langle \mathbf{v}_l, \mathbf{v}_l' \rangle \right)$$

[Koren, 2008]

Factorization Machines: Discussion II

- Representing categorical variables with real-valued variables and applying FMs is comparable to the factorization models that have been derived individually before (e.g. (bias) MF, tensor factorization, SVD++).
- FMs are much more flexible and can handle also non-categorical variables.
- Applying FMs is simple, as only data preprocessing has to be done (defining the real-valued predictor variables).

Computation Complexity

Factorization Machine model equation:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i\, x_i + \sum_{i=1}^{p} \sum_{j>i}^{p} \langle \mathbf{v}_i, \mathbf{v}_j \rangle\, x_i\, x_j$$

- Trivial computation: $\mathcal{O}(p^2\, k)$

Computation Complexity

Factorization Machine model equation:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i \, x_i + \sum_{i=1}^{p} \sum_{j>i}^{p} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j$$

- Trivial computation: $\mathcal{O}(p^2 \, k)$
- Efficient computation can be done in: $\mathcal{O}(p \, k)$

Computation Complexity

Factorization Machine model equation:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i \, x_i + \sum_{i=1}^{p} \sum_{j>i}^{p} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j$$

- Trivial computation: $\mathcal{O}(p^2 \, k)$
- Efficient computation can be done in: $\mathcal{O}(p \, k)$
- Making use of many zeros in $\mathbf{x}$ even in: $\mathcal{O}(N_z(\mathbf{x}) \, k)$, where $N_z(\mathbf{x})$ is the number of non-zero elements in vector $\mathbf{x}$.

Efficient Computation

The model equation of an FM can be computed in $\mathcal{O}(p\,k)$.

Efficient Computation

The model equation of an FM can be computed in $\mathcal{O}(p\,k)$.

Proof:

$$\begin{aligned}\hat{y}(\mathbf{x}) &:= w_0 + \sum_{i=1}^{p} w_i\,x_i + \sum_{i=1}^{p}\sum_{j>i}^{p} \langle \mathbf{v}_i, \mathbf{v}_j \rangle\, x_i\,x_j \\ &= w_0 + \sum_{i=1}^{p} w_i\,x_i + \frac{1}{2}\sum_{f=1}^{k}\left[\left(\sum_{i=1}^{p} x_i\,v_{i,f}\right)^2 - \sum_{i=1}^{p}\left(x_i\,v_{i,f}\right)^2\right]\end{aligned}$$

Efficient Computation

The model equation of an FM can be computed in $\mathcal{O}(p\,k)$.

Proof:

$$\begin{aligned}\hat{y}(\mathbf{x}) &:= w_0 + \sum_{i=1}^{p} w_i\, x_i + \sum_{i=1}^{p}\sum_{j>i}^{p} \langle \mathbf{v}_i, \mathbf{v}_j \rangle\, x_i\, x_j \\ &= w_0 + \sum_{i=1}^{p} w_i\, x_i + \frac{1}{2}\sum_{f=1}^{k}\left[\left(\sum_{i=1}^{p} x_i\, v_{i,f}\right)^2 - \sum_{i=1}^{p} (x_i\, v_{i,f})^2\right]\end{aligned}$$

- In the sums over i, only non-zero x_i elements have to be summed up $\Rightarrow \mathcal{O}(N_z(\mathbf{x})\,k)$.
- (The complexity of polynomial regression is $\mathcal{O}(N_z(\mathbf{x})^2)$.)

Multilinearity

FMs are multilinear:

$$\forall \theta \in \Theta = \{w_0, \mathbf{w}, \mathbf{V}\} : \qquad \hat{y}(\mathbf{x}, \theta) = h_{(\theta)}(\mathbf{x})\,\theta + g_{(\theta)}(\mathbf{x})$$

where $g_{(\theta)}$ and $h_{(\theta)}$ do not depend on the value of θ.

Multilinearity

FMs are multilinear:

$$\forall \theta \in \Theta = \{w_0, \mathbf{w}, \mathbf{V}\}: \qquad \hat{y}(\mathbf{x}, \theta) = h_{(\theta)}(\mathbf{x})\, \theta + g_{(\theta)}(\mathbf{x})$$

where $g_{(\theta)}$ and $h_{(\theta)}$ do not depend on the value of θ.

E.g. for second order effects ($\theta = v_{l,f}$):

$$\hat{y}(\mathbf{x}, v_{l,f}) := \overbrace{w_0 + \sum_{i=1}^{p} w_i\, x_i + \sum_{i=1}^{p} \sum_{j=i+1}^{p} \sum_{\substack{f'=1 \\ (f' \neq f) \vee (l \notin \{i,j\})}}^{k} v_{i,f'}\, v_{j,f'}\, x_i\, x_j}^{g_{(v_{l,f})}(\mathbf{x})}$$

$$+\, v_{l,f}\ \underbrace{x_l \sum_{i=1, i \neq l} v_{i,f}\, x_i}_{h_{(v_{l,f})}(\mathbf{x})}$$

Learning

Using these properties, learning algorithms can be developed:

- L2-regularized regression and classification:
  - Stochastic gradient descent [Rendle, 2010]
  - Alternating least squares/ Coordinate Descent [Rendle et al., 2011, Rendle 2012]
  - Markov Chain Monte Carlo (for Bayesian FMs) [Freudenthaler et al. 2011, Rendle 2012]
- L2-regularized ranking:
  - Stochastic gradient descent [Rendle, 2010]

All the proposed learning algorithms have a runtime of $O(k\, N_z(X)\, i)$, where i is the number of iterations and $N_z(X)$ the number of non-zero elements in the design matrix X.

Stochastic Gradient Descent (SGD)

- For each training case $(\mathbf{x}, y) \in S$, SGD updates the FM model parameter θ using:

$$\theta' = \theta - \alpha \, \left((\hat{y}(\mathbf{x}) - y) h_{(\theta)}(\mathbf{x}) + \lambda_{(\theta)} \theta\right)$$

- α is the learning rate / step size.
- $\lambda_{(\theta)}$ is the regularization value of the parameter θ.
- SGD can easily be applied to other loss functions.

[Rendle, 2010]

Alternating Least Squares (ALS)

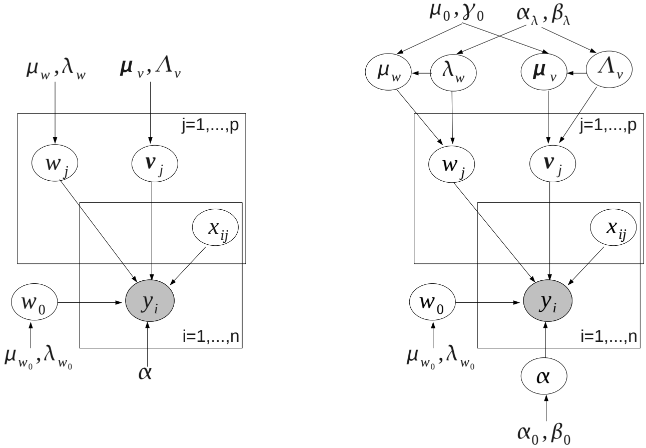
- Elementwise ALS updates each FM model parameter θ using:

$$\theta' = -\frac{\sum_{(\mathbf{x},y)\in S} \left(g_{(\theta)}(\mathbf{x}) - y\right) h_{(\theta)}(\mathbf{x})}{\sum_{(\mathbf{x},y)\in S} h^2_{(\theta)}(\mathbf{x}) + \lambda_{(\theta)}}$$

- Using caches of intermediate results, the runtime for updating all model parameters is $O(k\, N_z(X))$.
- The advantage of ALS compared to SGD is that no learning rate has to be specified.
- ALS can be extended to classification [Rendle, 2012].

[Rendle et al., 2011]

Bayesian FMs (BFM)

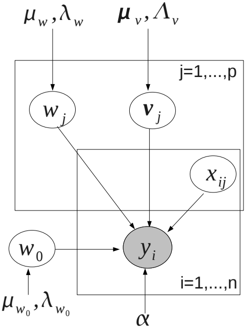


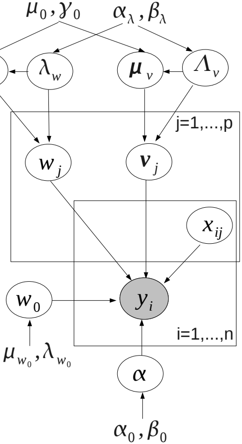
$$w_0 \sim \mathcal{N}(\mu_{w_0}, 1/\lambda_{w_0}), \quad \forall j \in \{1, \dots, p\}: \; w_j \sim \mathcal{N}(\mu_w, 1/\lambda_w), \quad \mathbf{v}_j \sim \mathcal{N}(\mu_v, \Lambda_v^{-1})$$

$$\mu_w \sim \mathcal{N}(\mu_0, \gamma_0 \lambda_w), \quad \lambda_w \sim \Gamma(\alpha_\lambda, \beta_\lambda), \quad \mu_{v,f} \sim \mathcal{N}(\mu_0, \gamma_0 \lambda_{v,f}), \quad \lambda_{v,f} \sim \Gamma(\alpha_\lambda, \beta_\lambda)$$

[Freudenthaler et al., 2011]

Bayesian FMs (BFM)





- The SGD and ALS models correspond to the left model.
- The right side is a two level model that integrates priors.

[Freudenthaler et al., 2011]

Bayesian FMs (BFM): Inference

- For Bayesian inference an efficient Gibbs sampler can be derived.
- The Gibbs posterior distribution for each model parameter θ is related to the ALS.
- Sampling all model parameters once can be done in $O(k\, N_z(X))$ as well.
- Introducing hyperpriors and integrating over priors has the advantage over ALS that the values of the priors are 'automatically' found.
- BFMs can be extended to classification [Rendle, 2012].

[Freudenthaler et al., 2011]

Outline

Applications

FMs are especially suited for ML problems:

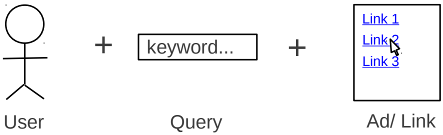
- Categorical variables of large domain.
- Number of predictor variables is large.
- Interactions between predictor variables are of interest.
- Several variables involved.

(Context-aware) Recommender Systems

- Main variables:
  - User ID (categorical)
  - Item ID (categorical)
- Additional variables:
  - time
  - mood
  - user profile
  - item meta data
  - ...
- Examples: Netflix prize, Movielens, KDDCup 2011

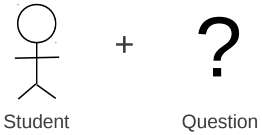
Clickthrough Prediction

- Main variables:
  - User ID
  - Query ID
  - Ad/ Link ID
- Additional variables:
  - query tokens
  - user profile
  - ...
- Example: KDDCup 2012 Track 2 (FM placed 3rd/171)



Student Performance Prediction

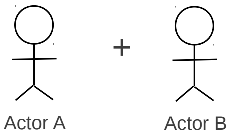
- Main variables:
  - Student ID
  - Question ID
- Additional variables:
  - question hierarchy
  - sequence of questions
  - skills required
  - ...
- Examples: KDDCup 2010, Grockit Challenge[2] (FM placed 1st/241)



[2] `http://www.kaggle.com/c/WhatDoYouKnow`

Link Prediction in Social Networks

- Main variables:
  - Actor A ID
  - Actor B ID
- Additional variables:
  - profiles
  - actions
  - ...
- Example: KDDCup 2012 Track 1 (FM placed 2nd/658)



libFM Software

libFM is an implementation of FMs

- Model: second-order FMs
- Learning/ inference: SGD, ALS, MCMC
- Classification and regression
- Uses the same data format as LIBSVM, LIBLINEAR [Lin et. al], SVMlight [Joachims].
- Supports variable grouping.
- Available with source code.

[http://www.libfm.org/]

Outline

Summary

- Real-valued predictor variables can encode information from variables of other domains, e.g. categorical variables.
- Applying linear regression to large categorical domains results in too little expressiveness; applying polynomial regression results in too much expressiveness.
- Factorization Machines (FM) are a polynomial regression model with factorized interaction parameters.
- FMs bring together the generality of standard machine learning methods with the prediction quality of factorization models.
- FMs are multilinear and can be computed efficiently.

Y. Cai, M. Zhang, D. Luo, C. Ding, and S. Chakravarthy.
Low-order tensor decompositions for social tagging recommendation.
In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 695–704, New York, NY, USA, 2011. ACM.

J. Carroll and J. Chang.
Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition.
Psychometrika, 35:283–319, 1970.

A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari.
Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation.
Wiley Publishing, 2009.

L. Drumond, S. Rendle, and L. Schmidt-Thieme.
Predicting rdf triples in incomplete knowledge bases with tensor factorization.

In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 326–331, New York, NY, USA, 2012. ACM.

T. Franz, A. Schultz, S. Sizov, and S. Staab.
Triplerank: Ranking semantic web data by tensor decomposition.
In *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, pages 213–228, Berlin, Heidelberg, 2009. Springer-Verlag.

C. Freudenthaler, L. Schmidt-Thieme, and S. Rendle.
Bayesian factorization machines.
In *Workshop on Sparse Representation and Low-rank Approximation, NIPS 2011*, 2011.

R. A. Harshman.
Foundations of the parafac procedure: models and conditions for an 'exploratory' multimodal factor analysis.
UCLA Working Papers in Phonetics, pages 1–84, 1970.

A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme.
Information retrieval in folksonomies: Search and ranking.

In Y. Sure and J. Domingue, editors, *The Semantic Web: Research and Applications*, volume 4011 of *Lecture Notes in Computer Science*, pages 411–426, Heidelberg, June 2006. Springer.

A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver.
Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering.
In *RecSys '10: Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86, New York, NY, USA, 2010. ACM.

T. G. Kolda and B. W. Bader.
Tensor decompositions and applications.
SIAM Review, 51(3):455–500, September 2009.

Y. Koren.
Factorization meets the neighborhood: a multifaceted collaborative filtering model.
In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, New York, NY, USA, 2008. ACM.

Y. Koren.
Collaborative filtering with temporal dynamics.
In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456, New York, NY, USA, 2009. ACM.

L. D. Lathauwer, B. D. Moor, and J. Vandewalle.
A multilinear singular value decomposition.
SIAM J. Matrix Anal. Appl., 21(4):1253–1278, 2000.

S. Rendle.
Factorization machines.
In *Proceedings of the 10th IEEE International Conference on Data Mining*. IEEE Computer Society, 2010.

S. Rendle.
Factorization machines with libfm.
ACM Trans. Intell. Syst. Technol., 3(3):57:1–57:22, May 2012.

S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme.
Factorizing personalized markov chains for next-basket recommendation.

In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 811–820, New York, NY, USA, 2010. ACM.

S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme.
Fast context-aware recommendations with factorization machines.
In *Proceedings of the 34th ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2011.

S. Rendle and L. Schmidt-Thieme.
Pairwise interaction tensor factorization for personalized tag recommendation.
In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 81–90, New York, NY, USA, 2010. ACM.

J.-T. Sun, H.-J. Zeng, H. Liu, Y. Lu, and Z. Chen.
Cubesvd: a novel approach to personalized web search.
In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 382–390, New York, NY, USA, 2005. ACM.

L. Tucker.
Some mathematical notes on three-mode factor analysis.

Psychometrika, 31:279–311, 1966.

L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell.
Temporal collaborative filtering with bayesian probabilistic tensor factorization.
In *Proceedings of SIAM Data Mining*, 2010.

A. Zimdars, D. M. Chickering, and C. Meek.
Using temporal data for making recommendations.
In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 580–588, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.