Machine Learning with Graphs (MLG)

RecSys: Factorization Machine

Incorporating Features into Matrix Factorization

Cheng-Te Li (李政德)

Institute of Data Science

National Cheng Kung University

chengte@mail.ncku.edu.tw

References

- P. Jain and I. S. Dhillon. "**Provable Non-linear Inductive Matrix Completion**" NIPS 2013 123 cites
- S. Rendle. "**Factorization Machines**" IEEE ICDM 2010 1050 cites
- S. Rendle. "**Factorization Machines with libFM**" ACM TIST 2012 851 cites
- FM Math: https://www.jefkine.com/recsys/2017/03/27/factorization-machines/
- FM Use Example https://yahooresearch.tumblr.com/post/133013312756/birds-apps-and-users-scalable-factorization

Towards Realistic Settings of RecSys

- Context-aware RecSys
- Click-through Prediction
- Student Performance Prediction
- Link Prediction in Social Networks

Context-aware RecSys

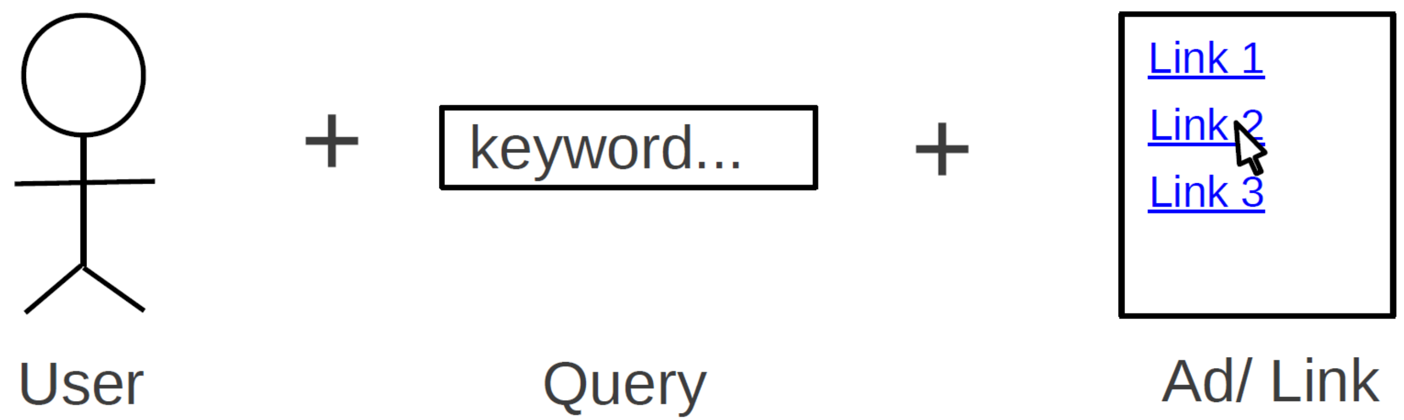
Main Targets	Features
User ID (one-hot)	Time
Item ID (one-hot)	Mood
	User profiles
	Item meta data

Examples: Netflix prize, Movielens, KDDCup 2011

Click-Through Prediction

Main Targets	Features
User ID (one-hot)	Query tokens
Query ID (one-hot)	User profiles
Ad/URL ID (one-hot)	Ad/URL meta data
	Time

Examples: KDDCup 2012 Track 2 (FM placed 3rd/171)

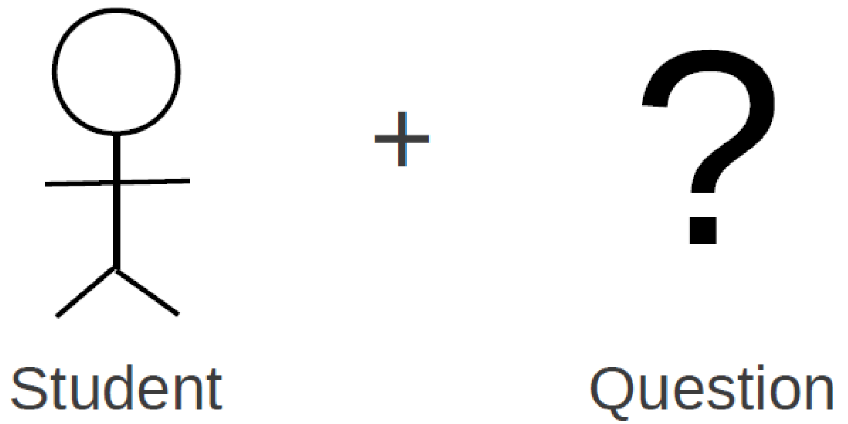


Student Performance Prediction

Main Targets	Features
Student ID (one-hot)	Question hierarchy
Question ID (one-hot)	Sequence of questions
	Skills required
	Question difficulty

Examples: KDDCup 2010, Grockit Challenge2 (FM placed 1st/241)

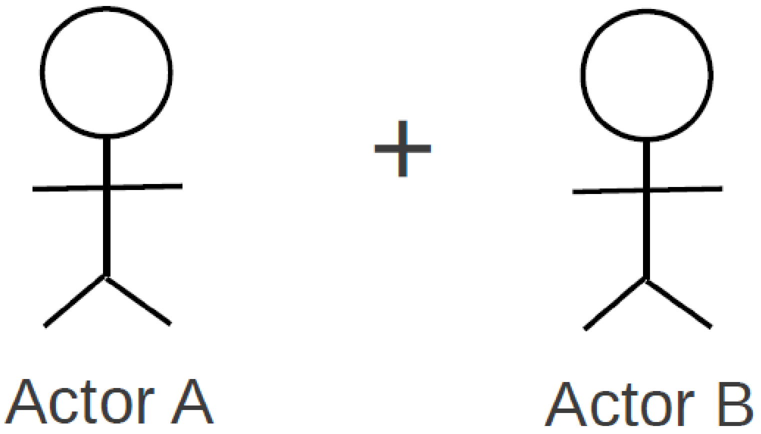
https://www.kaggle.com/c/WhatDoYouKnow



Link Prediction in Social Networks

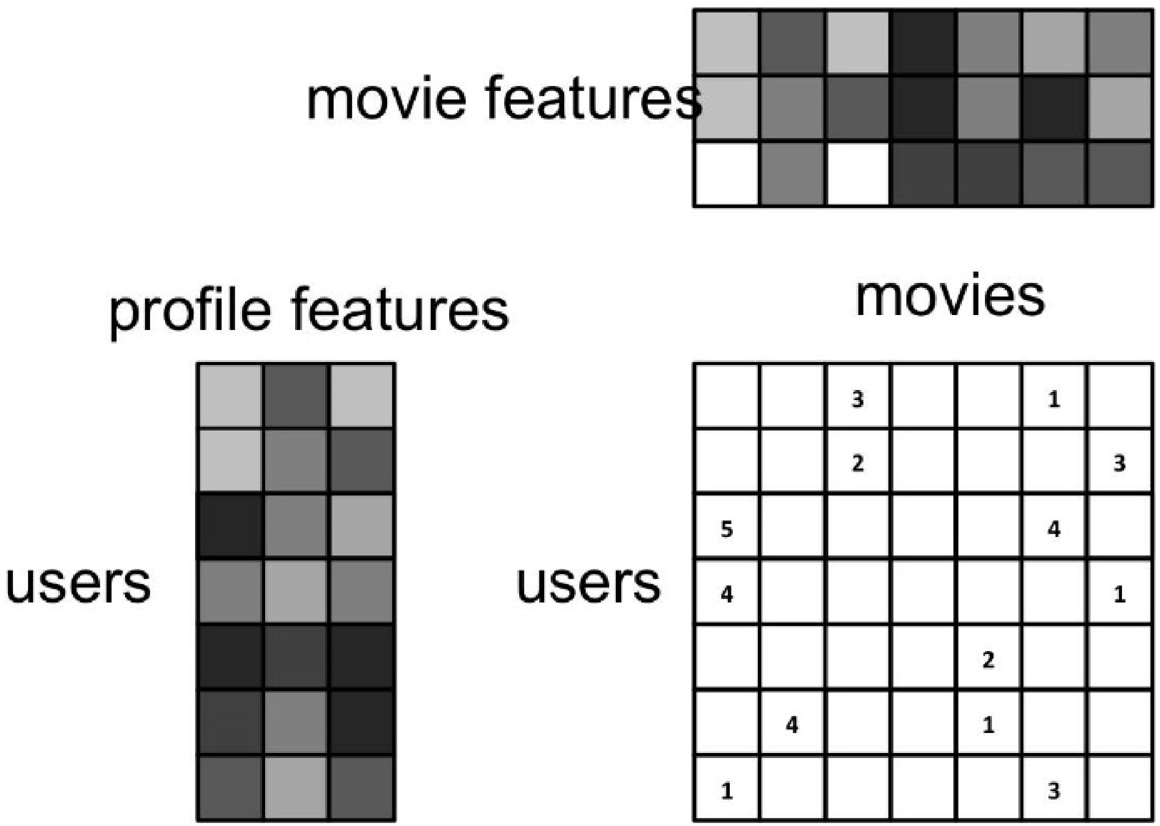
Main Targets	Features
User A ID (one-hot)	User profiles
User B ID (one-hot)	User clicks
	User posts
	User messages

Examples: KDDCup 2012 Track 1 (FM placed 2nd/658)



Matrix Factorization with Features

- Given: rating matrix $A \in \mathbb{R}^{m \times n}$, user feature matrix $X \in \mathbb{R}^{m \times d_1}$, item feature matrix $Y \in \mathbb{R}^{n \times d_2}$
- Goal: predict unknown ratings
  - Real values (regression)
  - Binary (classification)
  - Scores (ranking)



Applications

- ✓ User-item Recommendation
- ✓ Ad-word Recommendation
- ✓ Tag recommendation
- ✓ Disease-gene linkage prediction
- ✓ Document retrieval

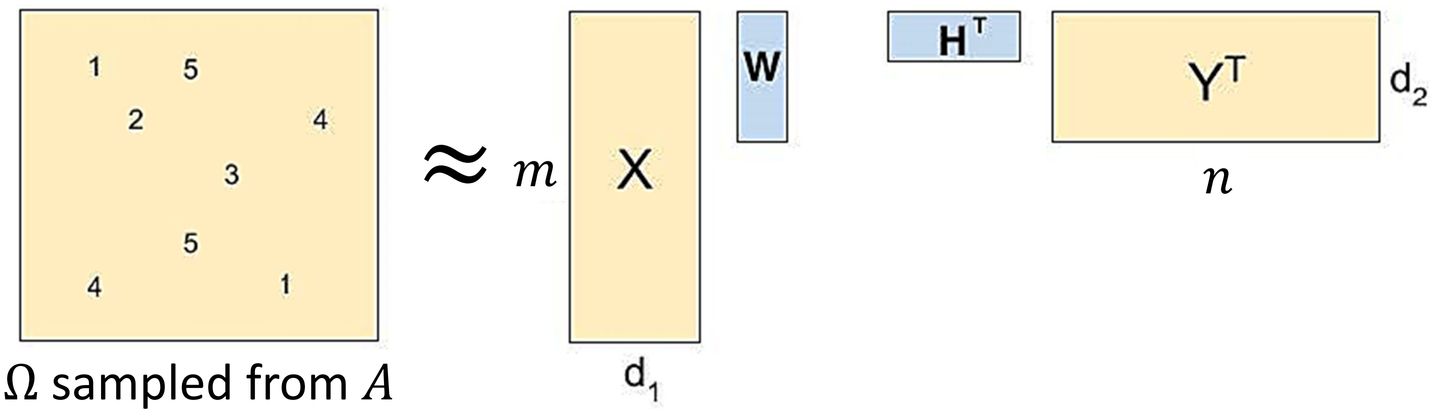
Inductive Matrix Completion (IMC)

- A popular approach to incorporate features to MF
- Given: row matrix $X \in \mathbb{R}^{m \times d_1}$, column matrix $Y \in \mathbb{R}^{n \times d_2}$, observation set Ω sampled from A
- The IMC objective:

$$\min_{\substack{W \in \mathbb{R}^{d_1 \times k} \\ H \in \mathbb{R}^{d_2 \times k}}} \sum_{(i,j) \in \Omega} \left(\mathbf{x}_i^{\mathrm{T}} W H^{\mathrm{T}} \mathbf{y}_j - A_{ij} \right)^2 + \frac{\lambda}{2} \|W\|_F^2 + \frac{\lambda}{2} \|H\|_F^2$$

$W\mathbf{x}_i$ & $H^{\mathrm{T}}\mathbf{y}_j$: k-dimensional embeddings of user i and item j, respectively

Inner product in the k-dimensional embedding space $\Longrightarrow$ prediction value



Factorization Machine (FM)

- Observation: Recommendation systems can be transformed to classification or regression

$$\left(i, j, A_{ij}\right) \Rightarrow \left(\mathbf{x}^{(i,j)}, A_{ij}\right)$$

$\mathbf{x}^{(i,j)}$: the feature vector extracted for user i and item j

  - $\mathbf{x}^{(i,j)} = \left[\mathbf{e}_i\ \mathbf{e}_j\ \mathbf{x}_i^{\mathrm{T}}\ \mathbf{x}_j^{\mathrm{T}}\right]$

$\mathbf{e}_i$ & $\mathbf{e}_j$ are **one-hot encodings** of user i & item j

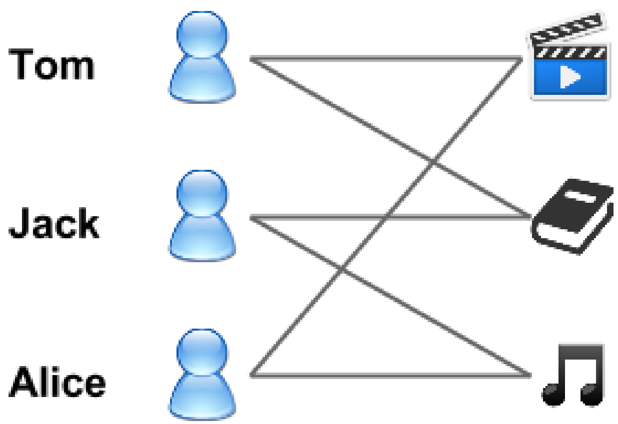
$\mathbf{x}_i$ & $\mathbf{x}_j$ are **feature vectors** of user i & item j

- Now we have a classification or regression problem with training data $\left\{\left(\mathbf{x}^{(i,j)}, A_{ij}\right)\right\}_{(i,j)\in\Omega}$

Example Training Data for FM

Feature vector / Response

	User Index			Item Index			User Features				Item Features				Response	
	Tom	Jack	Alice				Age	Male	Female		YCT	WIKI	Phrase			
x_1	1	0	0	0	1	0	14	1	0	...	.1	.2	.3	...	1	y_1
x_2	1	0	0	1	0	0	14	1	0	...	.1	.4	0	...	3	y_2
x_3	0	1	0	0	0	1	18	1	0	...	.2	.2	.9	...	2	y_3
x_4	0	1	0	0	1	0	18	1	0	...	.1	2	3	...	4	y_4
x_5	0	0	1	1	0	0	23	0	1	...	.1	.4	0	...	5	y_5
x_6	0	0	1	0	0	1	23	0	1	...	.2	.2	.9	...	2	y_6



Linear Model

- What model should we use for classification/regression?
- Option 1: Linear Regression

d: dimension of vector $\mathbf{x}$

$$\hat{y}(\mathbf{x}) = w_0 + \mathbf{w}^{\mathrm{T}}\mathbf{x} = w_0 + \sum_{i=1}^{d} w_i x_i$$

<u>**Example**</u>

Publisher	Advertiser
ESPN	Nike

$$\hat{y}(\mathbf{x}) = w_0 + s_{\mathrm{ESPN}} + s_{\mathrm{NIKE}}$$

- Drawback 1: Cannot learn cross-feature effects like:

  "Nike has super high CTR on ESPN"

- Drawback 2: usually too simple and tend to underfitting

Polynomial Regression (Poly2)

- Option 2: Degree-2 Polynomial

d: dimension of vector $\mathbf{x}$

$$\hat{y}(\mathbf{x}) = w_0 + \mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathbf{x}^{\mathrm{T}}M\mathbf{x} = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=i+1}^{d} w_{ij} x_i x_j$$

Example

Publisher	Advertiser
ESPN	Nike

$$M = [w_{ij}]$$

$$\hat{y}(\mathbf{x}) = w_0 + s_{\text{ESPN}} + s_{\text{NIKE}} + s_{\text{ESPN,NIKE}}$$

- Drawback: weak generalization ability
  i.e., cannot estimate parameter w_{ij}
  where (i, j) never co-occurs in feature vectors
  (not work well for sparse data)

Poly2: Illustration

First-order: Linear Regression

Second-order: Pair-wise interactions between **non-zero** features

$$\hat{y}(\mathbf{x}) = w_0 + \underbrace{\sum_{i=1}^{d} w_i x_i}_{\text{First-order}} + \underbrace{\sum_{i=1}^{d} \sum_{j=i+1}^{d} \boxed{w_{ij}} x_i x_j}_{\text{Second-order}}$$

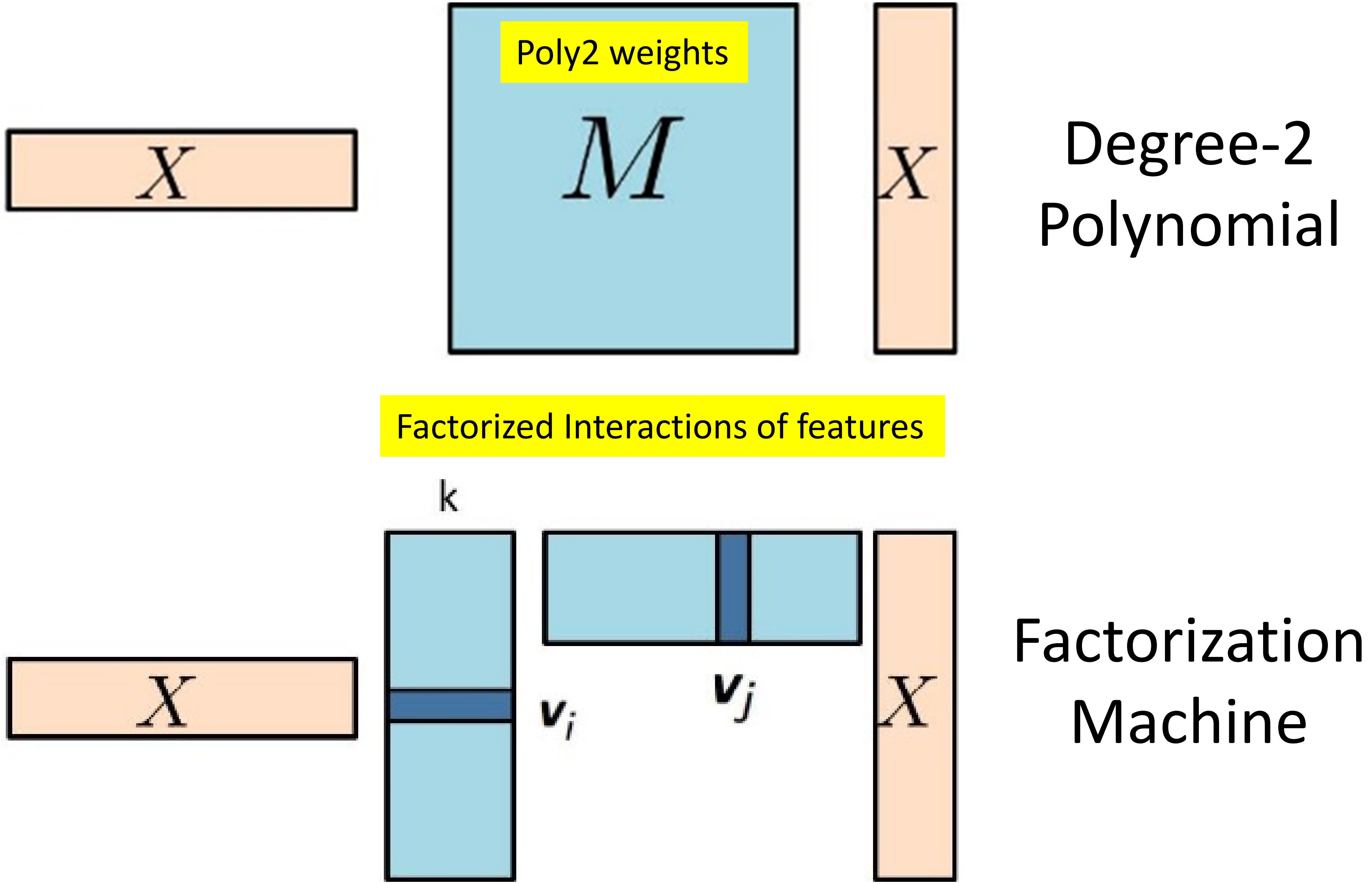
Input $\mathbf{x}$:

0	1	0	0	0	1	0	0	0.3	0	0.2	1	0	0.4	0	0.1	0

$$\underbrace{x_0 \quad x_1 \quad \cdots}_{\text{User ID}} \quad \underbrace{\cdots \quad x_i \quad \cdots}_{\text{Item ID}} \quad \underbrace{\cdots \quad x_j \quad \cdots}_{\text{User Features}} \quad \underbrace{\cdots \quad x_{n-1} \quad x_n}_{\text{Item Features}}$$

- $M = [w_{ij}]_{d \times d}$ brings too many parameters ($\boldsymbol{d^2}$)
- $\mathbf{x}$ is sparse → matrix M is more sparse
- Some pairs rarely occur in user-item interactions: w_{ij} hard to train

Poly2 vs. FM



Poly2 weights
X
M
X
Degree-2
Polynomial
Factorized Interactions of features
k
X
v_i
v_j
X
Factorization
Machine

FM: Illustration

Poly2

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \boxed{w_{ij}} x_i x_j$$

Replace w_{ij} by dot product of vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ with dimension k

FM

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \mathbf{v}_i^{\mathrm{T}} \mathbf{v}_j x_i x_j$$

$$V = [v_i]_{d \times k}$$

#parameters= dk where $d \gg k$

Learn:

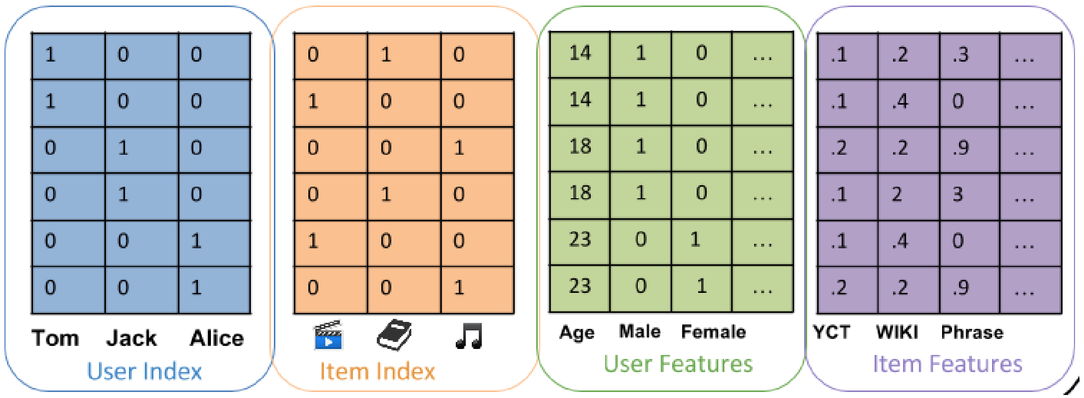
Latent vector $\mathbf{v}_i$

Latent vector $\mathbf{v}_j$

Input $\mathbf{x}$:

0	1	0	0	0	1	0	0	0.3	0	0.2	1	0	0.4	0	0.1	0
x_0	x_1	$\cdots$	$\cdots$		x_i	$\cdots$		$\cdots$		x_j	$\cdots$	$\cdots$			x_{n-1}	x_n

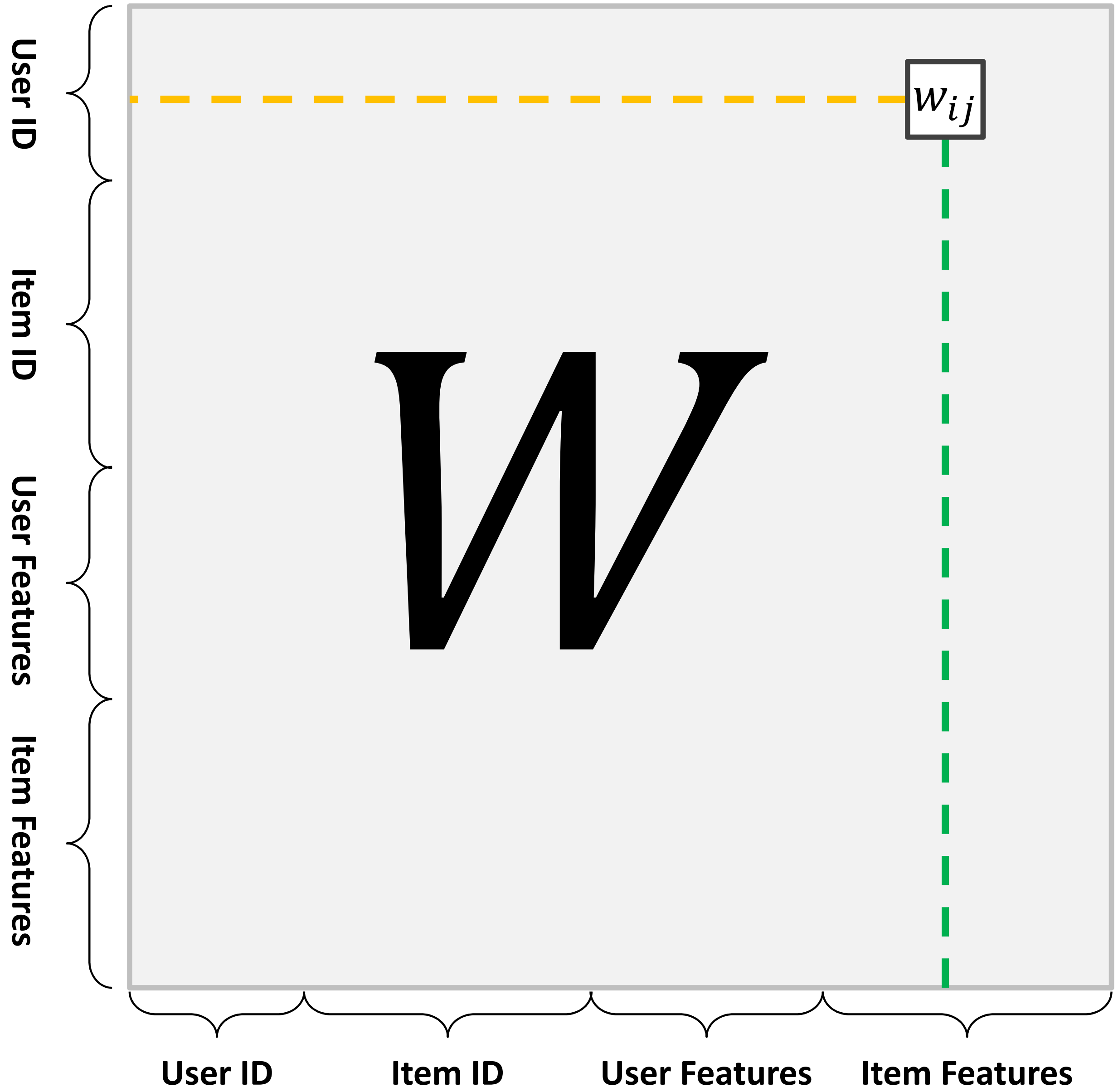
User ID | Item ID | User Features | Item Features



Tom	Jack	Alice				Age	Male	Female		YCT	WIKI	Phrase	
1	0	0	0	1	0	14	1	0	…	.1	.2	.3	…
1	0	0	1	0	0	14	1	0	…	.1	.4	0	…
0	1	0	0	0	1	18	1	0	…	.2	.2	.9	…
0	1	0	0	1	0	18	1	0	…	.1	2	3	…
0	0	1	1	0	0	23	0	1	…	.1	.4	0	…
0	0	1	0	0	1	23	0	1	…	.2	.2	.9	…

Interaction Matrix W

feature i



feature j

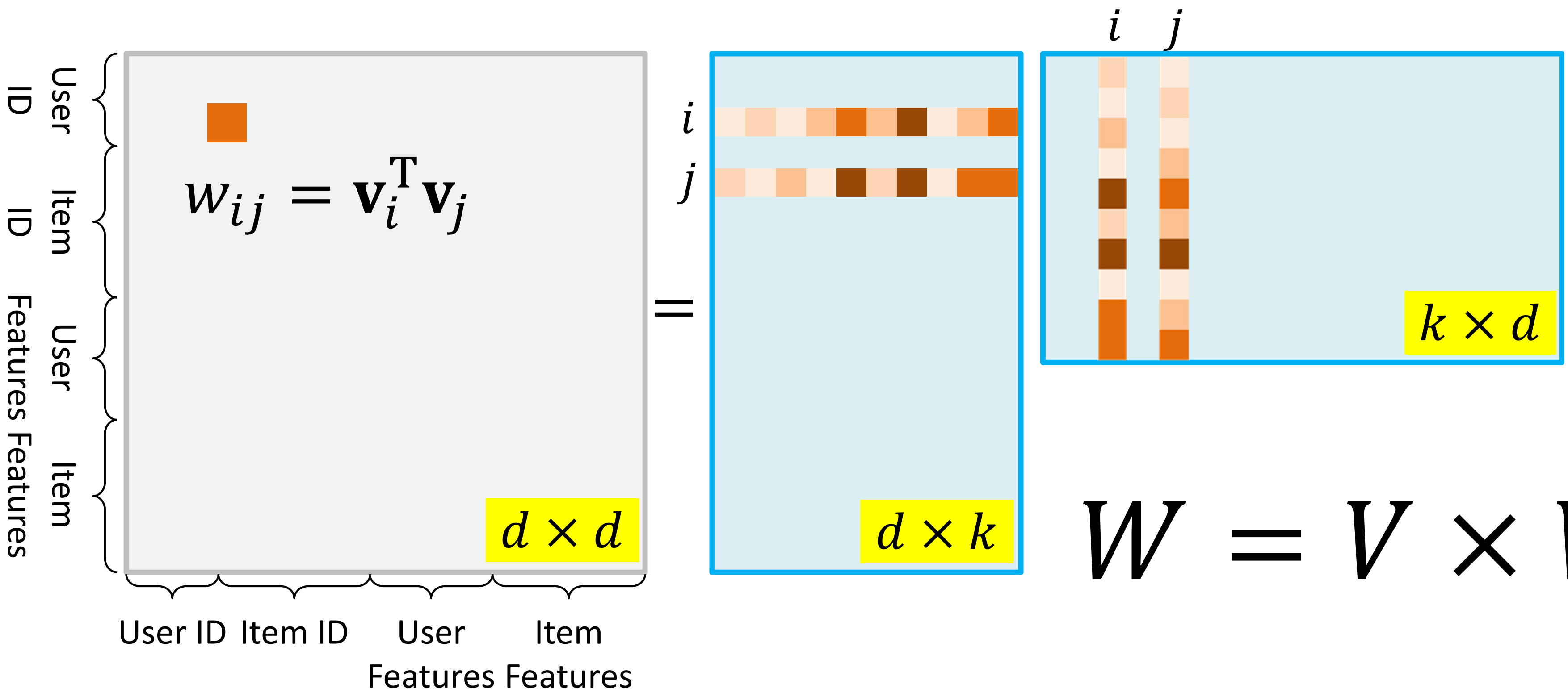
$$w_{ij} = \mathbf{v}_i^{\mathrm{T}} \mathbf{v}_j$$

$d \times d$

Factorized Interactions of Features

i	j
UI: 數據所學生	UI: 資工所學生
UI: 數據所學生	II: MacBook Pro
UI: 數據所學生	UF: 上Google時間
UI: 數據所學生	IF: 具備GPU顯卡
IF: 續航力高	IF: 具備GPU顯卡

i	j
II: MacBook Pro	II: iPad Pro
II: MacBook Pro	IF: 品牌Apple
UF: 上FB時間	UF: 上PTT時間
UF: 有Github帳號	IF: 品牌Apple
UF: 近視>500度	IF: 熱門手遊



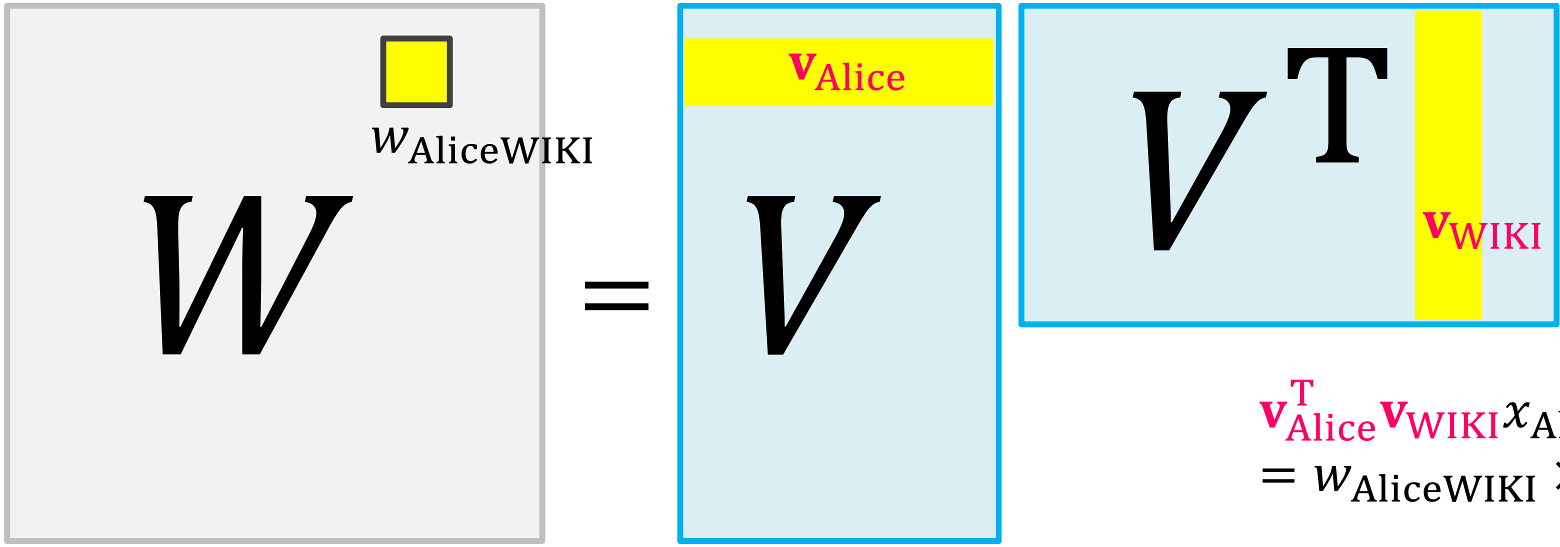
$$W = V \times V^{\mathrm{T}}$$

Interaction Matrix W

$$\underbrace{W}_{d \times d} = \underbrace{V}_{d \times k}\ \underbrace{V^{\mathrm{T}}}_{k \times d}$$

$$\hat{y}(\mathbf{x}) = w_0 + \mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathbf{x}^{\mathrm{T}} V^{\mathrm{T}} V \mathbf{x} = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \overset{w_{ij}}{\overset{\|}{\mathbf{v}_i^{\mathrm{T}} \mathbf{v}_j}} x_i x_j$$

Interaction Matrix W



$\mathbf{v}_{\text{Alice}}^{\mathrm{T}}\mathbf{v}_{\text{WIKI}}x_{\text{Alice}}x_{\text{WIKI}}$
$= w_{\text{AliceWIKI}} \times 1 \times 0.4$

$$\hat{y}(\mathbf{x}) = w_0 + \mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathbf{x}^{\mathrm{T}}V^{\mathrm{T}}V\mathbf{x} = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=i+1}^{d} \mathbf{v}_i^{\mathrm{T}}\mathbf{v}_j x_i x_j$$

User Index			Item Index			User Features				Item Features			
Tom	Jack	Alice				Age	Male	Female		YCT	WIKI	Phrase	
1	0	0	0	1	0	14	1	0	…	.1	.2	.3	…
1	0	0	1	0	0	14	1	0	…	.1	.4	0	…
0	1	0	0	0	1	18	1	0	…	.2	.2	.9	…
0	1	0	0	1	0	18	1	0	…	.1	2	3	…
0	0	1	1	0	0	23	0	1	…	.1	.4	0	…
0	0	1	0	0	1	23	0	1	…	.2	.2	.9	…

Factorization Machine (FM)

- Option 3: **FM**

First-order: Linear Regression

Second-order: Feature Interactions

$$\hat{y}(\mathbf{x}) = w_0 + \mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathbf{x}^{\mathrm{T}}V^{\mathrm{T}}V\mathbf{x} = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=i+1}^{d} \mathbf{v}_i^{\mathrm{T}}\mathbf{v}_j x_i x_j$$

$\mathbf{v}_i$ is the i-th column of V

$V \in \mathbb{R}^{d\times k}$

Example 1

Publisher	Advertiser
ESPN	Nike

$$\hat{y}(\mathbf{x}) = w_0 + s_{\mathrm{ESPN}} + s_{\mathrm{NIKE}} + \mathbf{v}_{\mathrm{ESPN}}^{\mathrm{T}}\mathbf{v}_{\mathrm{Nike}}$$

Example 2

Publisher (P)	Advertiser (A)	Gender (G)
ESPN	Nike	Male

$$\hat{y}(\mathbf{x}) = w_0 + s_{\mathrm{EPSN}} + s_{\mathrm{Nike}} + s_{\mathrm{Make}} + \mathbf{v}_{\mathrm{ESPN}}^{\mathrm{T}}\mathbf{v}_{\mathrm{Nike}} + \mathbf{v}_{\mathrm{ESPN}}^{\mathrm{T}}\mathbf{v}_{\mathrm{Make}} + \mathbf{v}_{\mathrm{Nike}}^{\mathrm{T}}\mathbf{v}_{\mathrm{Male}}$$

Better Generalization of FM

- Strong generalization comes from learning latent feature vectors by their interactions

≥ 3	< 3	Channel	Brand	$\hat{y}(\mathbf{x})$
950	50	ESPN	Nike	$\boldsymbol{v}_{\text{ESPN}} \cdot \boldsymbol{v}_{\text{Nike}} + \cdots$
2	0	ESPN	Gucci	$\boldsymbol{v}_{\text{ESPN}} \cdot \boldsymbol{v}_{\text{Gucci}} + \cdots$
0	0	Vogue	Nike	$\boldsymbol{v}_{\text{Vogue}} \cdot \boldsymbol{v}_{\text{Nike}} + \cdots$
950	50	Vogue	Gucci	$\boldsymbol{v}_{\text{Vogue}} \cdot \boldsymbol{v}_{\text{Gucci}} + \cdots$

$\mathbf{v}_{\text{Vogue}}$ is learned from 1000 data points

$\mathbf{v}_{\text{Nike}}$ learned from 1000 data points

Feature interaction of Vogue and Nike never appears in data, but their latent vectors can be still trained if each is interacted by other features

→ lead to better generalization (than poly2) and deal with data sparsity

Training FM

- FM objective:

$$\min_{\substack{V \in \mathbb{R}^{d\times k}\\ \mathbf{w}}} \sum_{(i,j)\in\Omega} loss\left(\underbrace{w_0 + \mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i,j)} + \mathbf{x}^{(i,j)^{\mathrm{T}}} V^{\mathrm{T}} V \mathbf{x}^{(i,j)}}_{\text{predicted rating}}, \underbrace{A_{ij}}_{\text{ground-truth rating}}\right) + \lambda \|V\|_F^2 + \lambda\|\mathbf{w}\|_F^2$$

  - Ω: observed user-item entries
  - $\mathbf{x}^{(i,j)} = [\mathbf{e}_i \; \mathbf{e}_j \; \mathbf{x}_i^{\mathrm{T}} \; \mathbf{x}_j^{\mathrm{T}}]$

$\mathbf{e}_i$ & $\mathbf{e}_j$ are **one-hot encodings** of user i & item j

$\mathbf{x}_i$ & $\mathbf{x}_j$ are **feature vectors** of user i & item j

- Loss
  - **Squared error**
    $l(y_1, y_2) = (y_1 - y_2)^2$ // ratings (real values)
  - **Logistic error**
    $l(y_1, y_2) = \ln(1 + \exp(-y_1 y_2))$ // implicit feedback (binary)

FM: Discussion

- FM works with real values
- FM includes feature interactions like poly2 regression
- Model parameters for interactions are factorized
- # of model parameters is only $d \times k$
  - Instead of d^2 for poly2 regression, where $d \gg k$
- FM can be generalized to different "**factorization**" tasks
  - Matrix Factorization as FM
  - Multi-Labeling (Tag Recommendation) as FM
  - Temporal FM

MF as FM

Categorical Data

$$\hat{y}(\mathbf{x}) = w_0 + s_u + s_i + \mathbf{v}_u^{\mathrm{T}}\mathbf{v}_i$$

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

	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						

FM is identical to MF with biases

Applying regression models to this data leads to:

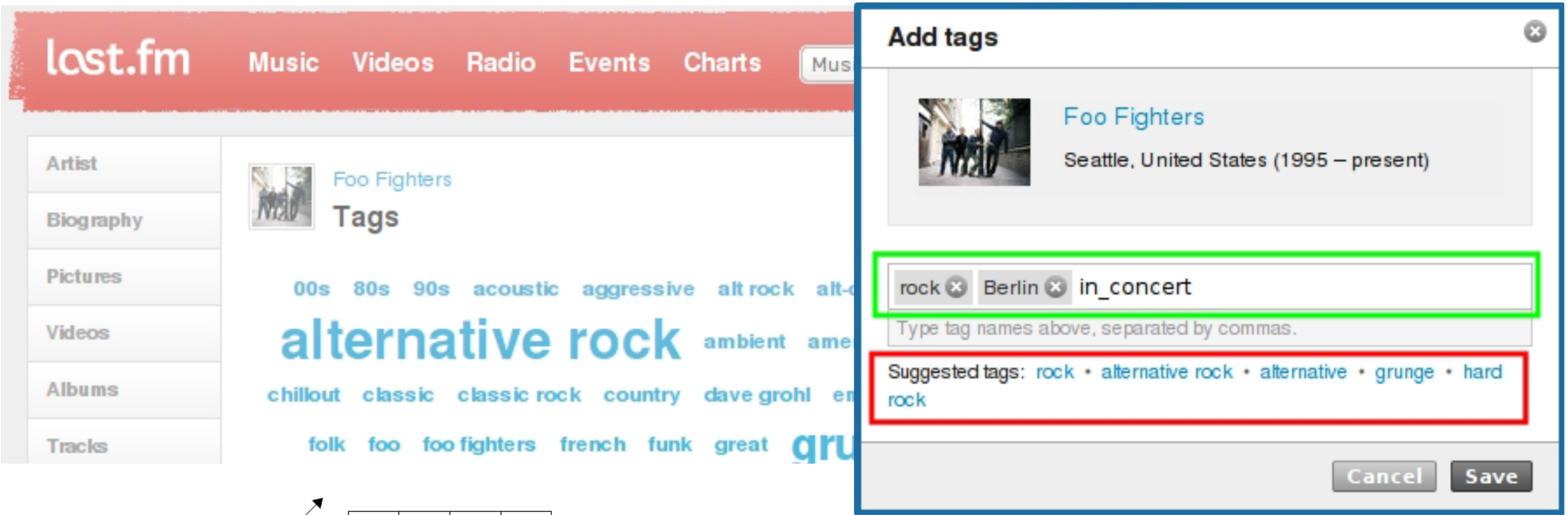
Linear Regression: $\hat{y}(\mathbf{x}) = w_0 + s_u + s_i$

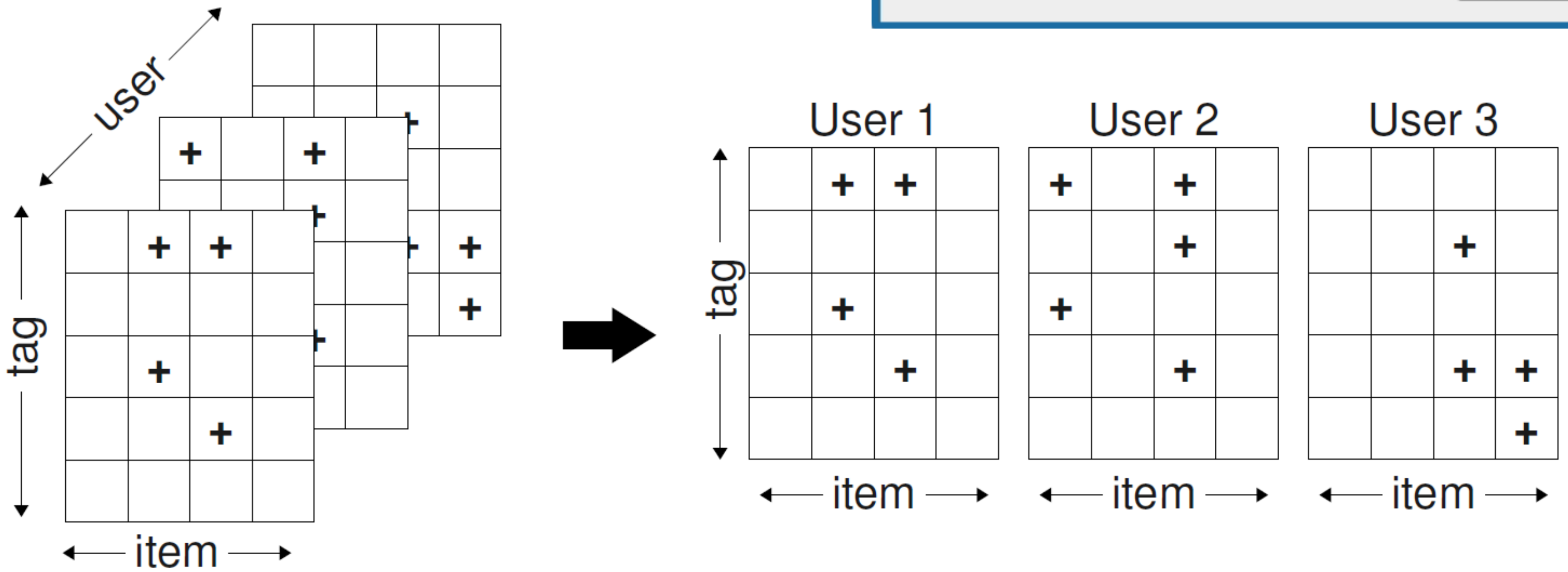
Polynomial Regression: $\hat{y}(\mathbf{x}) = w_0 + s_u + s_i + s_{u,i}$

Matrix Factorization (with biases): $\hat{y}(\mathbf{x}) = w_0 + b_u + b_i + \mathbf{p}_u^{\mathrm{T}}\mathbf{q}_i$

Personalized Tag Recommendation

Task: Recommend a user a (personalized) list of tags for a specific item





Tag Recommendation as FM

Three categorical variables encoded with real valued predictor

Feature vector $\mathbf{x}$

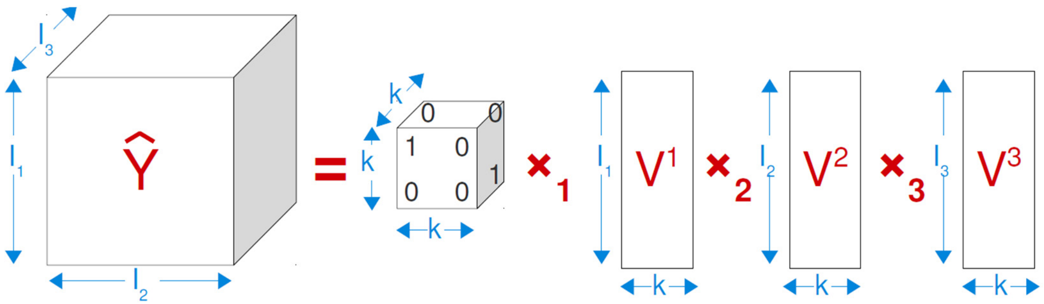
	User				Song					Tag				
	A	B	C	...	S1	S2	S3	S4	...	T1	T2	T3	T4	...
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	1	0	0	0	...
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0	1	0	0	...
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0	0	0	1	...
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	1	0	...
$\mathbf{x}^{(5)}$	0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

FM is a tensor factorization model with lower-level interactions (pairwise interaction)

$$\hat{y}(\mathbf{x}) = w_0 + s_u + s_i + s_t + \mathbf{v}_u^{\mathrm{T}}\mathbf{v}_i + \mathbf{v}_i^{\mathrm{T}}\mathbf{v}_t + \mathbf{v}_u^{\mathrm{T}}\mathbf{v}_t$$

FM with Time

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

A three-order FM includes the time-aware tensor factorization model

$$\hat{y}(\mathbf{x}) = w_0 + s_u + s_i + s_{b(t)} + \mathbf{v}_u^{\mathrm{T}}\mathbf{v}_i + \mathbf{v}_u^{\mathrm{T}}\mathbf{v}_{b(t)} + \mathbf{v}_i^{\mathrm{T}}\mathbf{v}_{b(t)}$$

$$+ \underbrace{\sum_{f=1}^{k} \mathbf{v}_{u,f}^{(3)}\mathbf{v}_{i,f}^{(3)}\mathbf{v}_{b(t),f}^{(3)}}_{}$$

Time Tensor Factorization Model

FM Short Summary

- Integrating real-valued features with categorical (one-hot) features into model encoding
- FM = linear regression + polynomial regression + matrix factorization (factorized interaction parameters)
  - Incorporating feature interactions
  - Deal with data sparsity in user-item matrix
  - Better generalization (learning latent vector of each feature)
- FM is a generalization of a variety of RecSys w/ features

Packages/Code of FM

- pywFM: Python wrapper for Steffen Rendle's libFM
  https://github.com/jfloff/pywFM
- pyFM: https://github.com/coreylynch/pyFM
- fastFM: https://github.com/ibayer/fastFM
- FM with PyTorch: https://github.com/rixwew/pytorch-fm
- FM with Tensorflow: https://github.com/geffy/tffm
- FM implemented in PyTorch
  https://www.kaggle.com/gennadylaptev/factorization-machine-implemented-in-pytorch
- More on PyTorch FM
  - https://github.com/mzaradzki/factorization-machine-for-prediction
  - https://github.com/jmhessel/fmpytorch