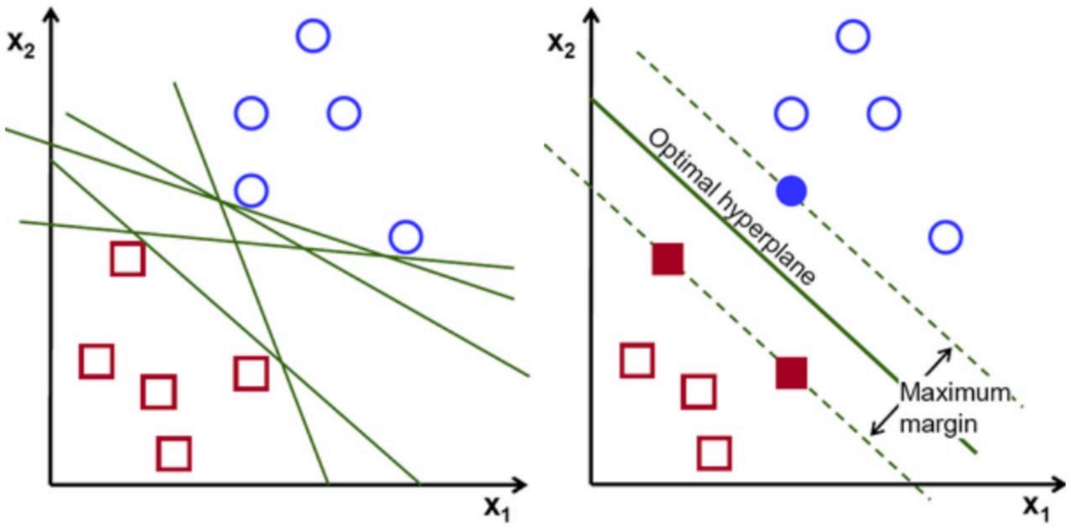
Factorization Machines

KAIST 산업및시스템공학과 이정우

Contents

1. Introduction



Background

1. SVM

- 주로 classification에 이용된다.
- margin을 최대화하는 선(decision boundary)
- decision boundary와 가장 가까운 점 support vector
- Outlier 구분
- Work on any real valued feature vector
- Cannot derive hyperplane in nonlinear kernel spaces under sparse data

1. Introduction

Background

2. Factorization Models

- 사용자와 아이템을 latent factor space에 mapping
- 사용자-아이템 상호작용은 내적을 이용하여 모델링 함

$$\min_{q,p} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

SGD(Stochastic Gradient Descent), ALS(Alternating Least Squares)

- 각각의 목적을 가지고 만들어진 모델이므로 general하게 사용할 수 없음
- Real Valued Feature Vector에는 적용할 수 없음
- SGD 등의 방법의 computational cost가 클 수도 있음

1. Introduction

SVM(Support Vector Machines) + Factorization Models

=

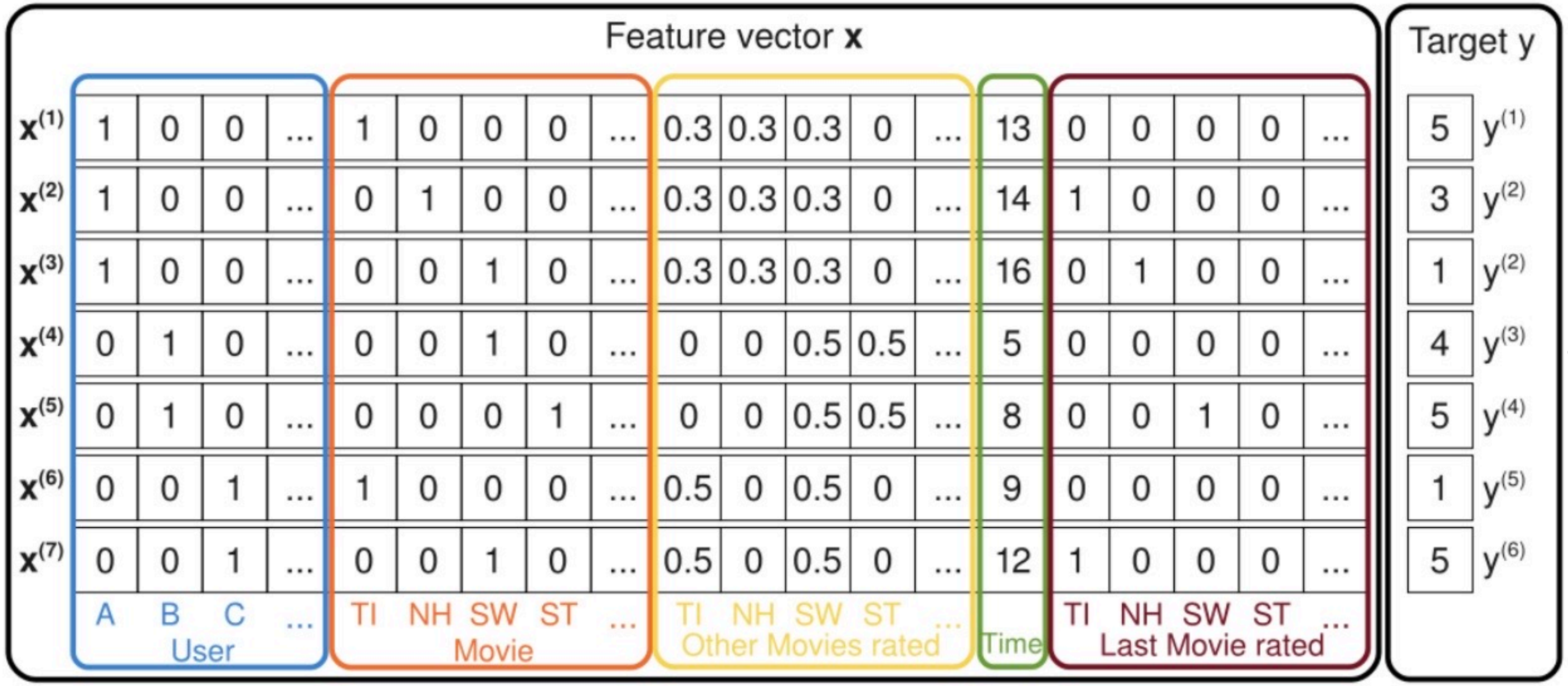
Factorization Machines

1. 어떤 실수값도 feature vector로 받을 수 있다.
2. factorized parameter를 이용하여 모든 interaction을 모델화할 수 있다. →

   데이터가 매우 sparse한 경우에도 성능이 좋음
3. 최적화 과정이 선형시간에 계산되므로 support vector 없이도 예측 가능

2. Prediction Under Sparsity

More Categorical Variables make data more Sparse



	User				Movie					Other Movies rated					Time	Last Movie rated					Target y	
	A	B	C	...	TI	NH	SW	ST	...	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	...	13	0	0	0	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1	$y^{(2)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4	$y^{(3)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5	$y^{(4)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1	$y^{(5)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5	$y^{(6)}$

FM input representation

$m(x)$: # of non-zero elements in vector x

$\overline{m_D}$: average of x

data S = {(Alice, Titanic, 2010-1, 5), (Bob, Star Wars, 2010-2, 3) … }

2. Prediction Under Sparsity

More Categorical Variables make data more Sparse

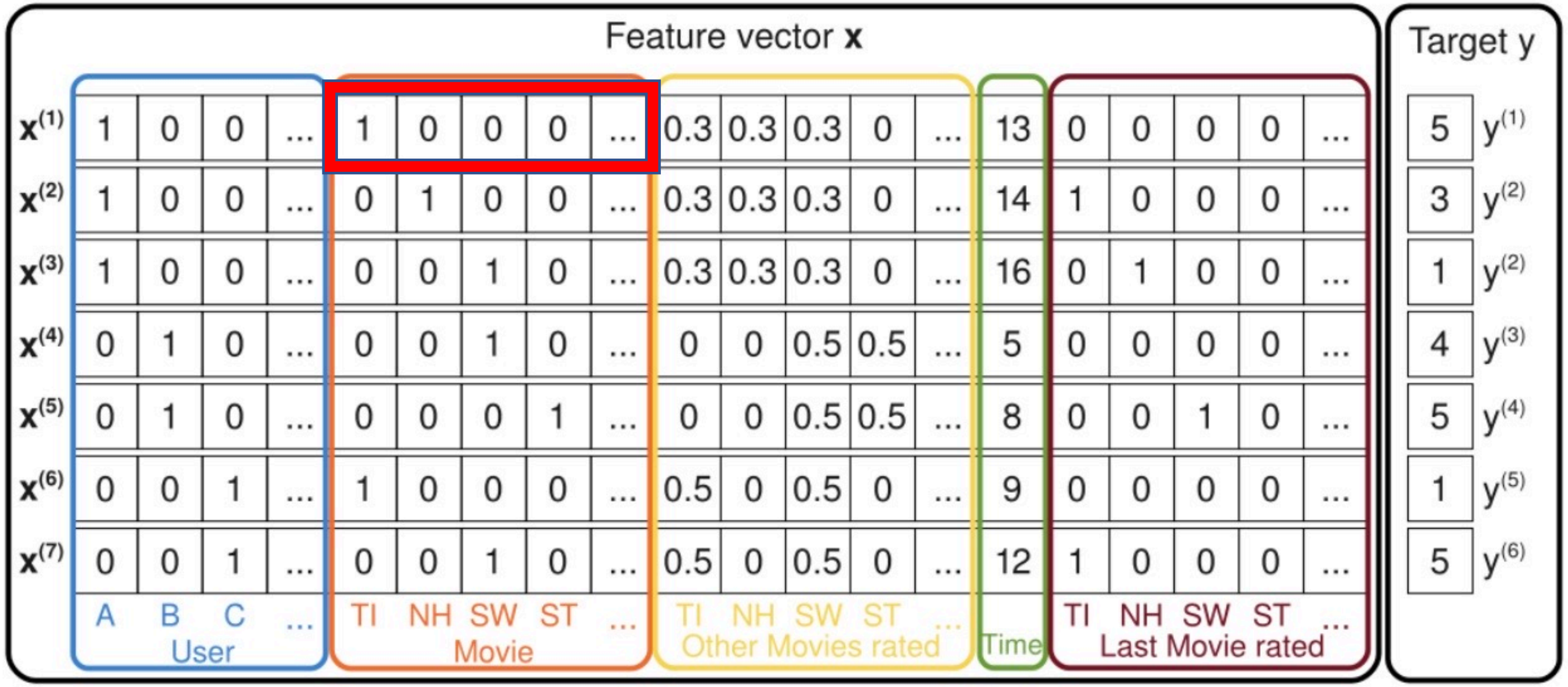
	Feature vector **x**																					Target y	
	User				Movie					Other Movies rated					Time	Last Movie rated							
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...		TI	NH	SW	ST	...			
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...		5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...		3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...		1	$y^{(2)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...		4	$y^{(3)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...		5	$y^{(4)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...		1	$y^{(5)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...		5	$y^{(6)}$

FM input representation

data S = {(Alice, Titanic, 2010-1, 5), (Bob, Star Wars, 2010-2, 3) … }

2. Prediction Under Sparsity

More Categorical Variables make data more Sparse

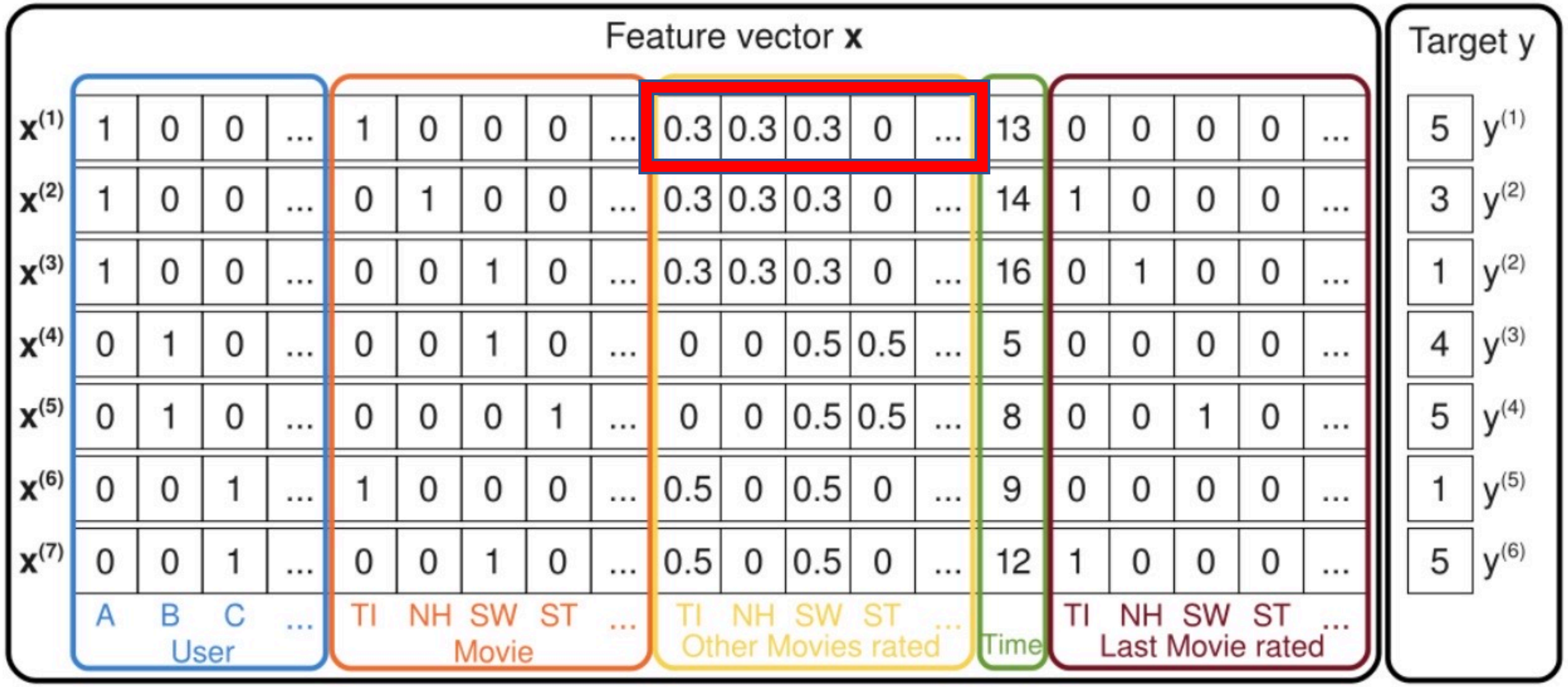


FM input representation

data S = {(Alice, Titanic, 2010-1, 5), (Bob, Star Wars, 2010-2, 3) … }

2. Prediction Under Sparsity

More Categorical Variables make data more Sparse

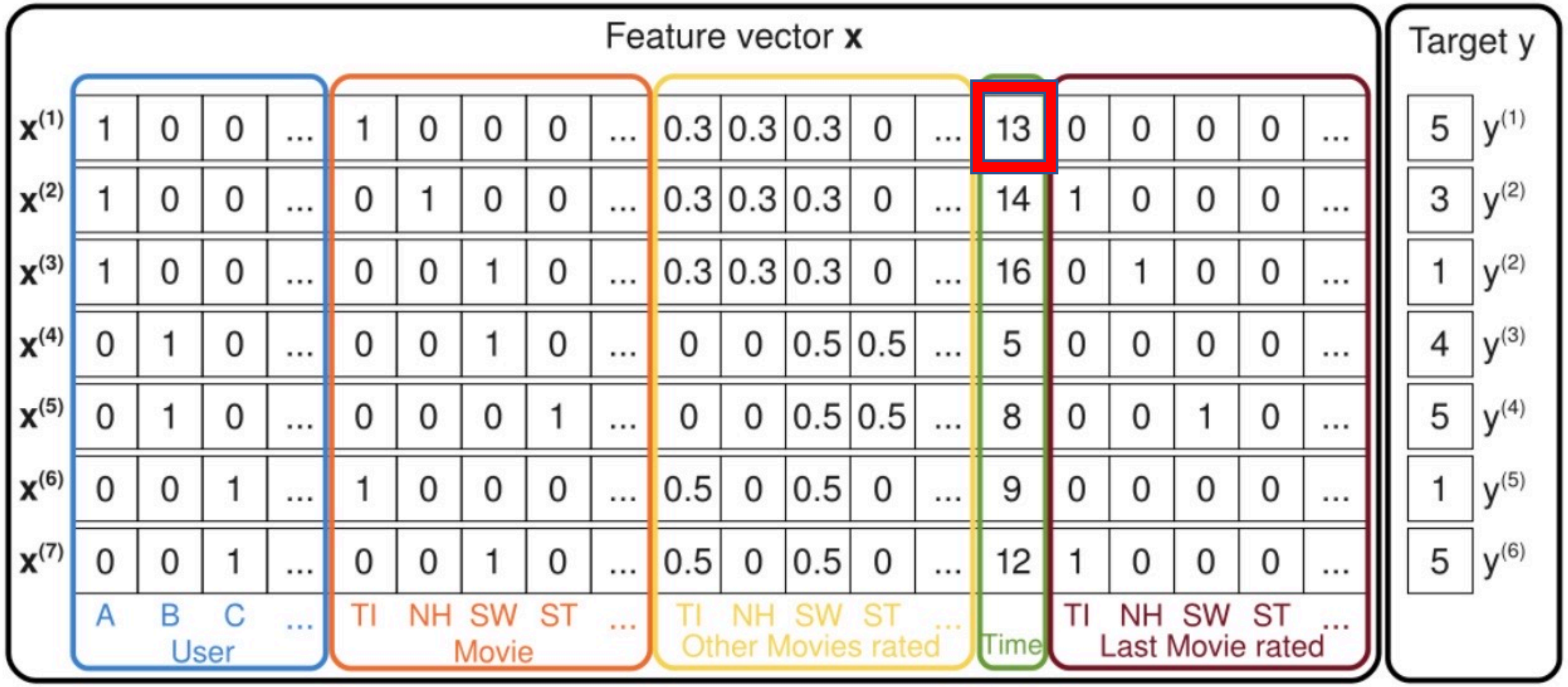


FM input representation

data S = {(Alice, Titanic, 2010-1, 5), (Bob, Star Wars, 2010-2, 3) … }

2. Prediction Under Sparsity

More Categorical Variables make data more Sparse

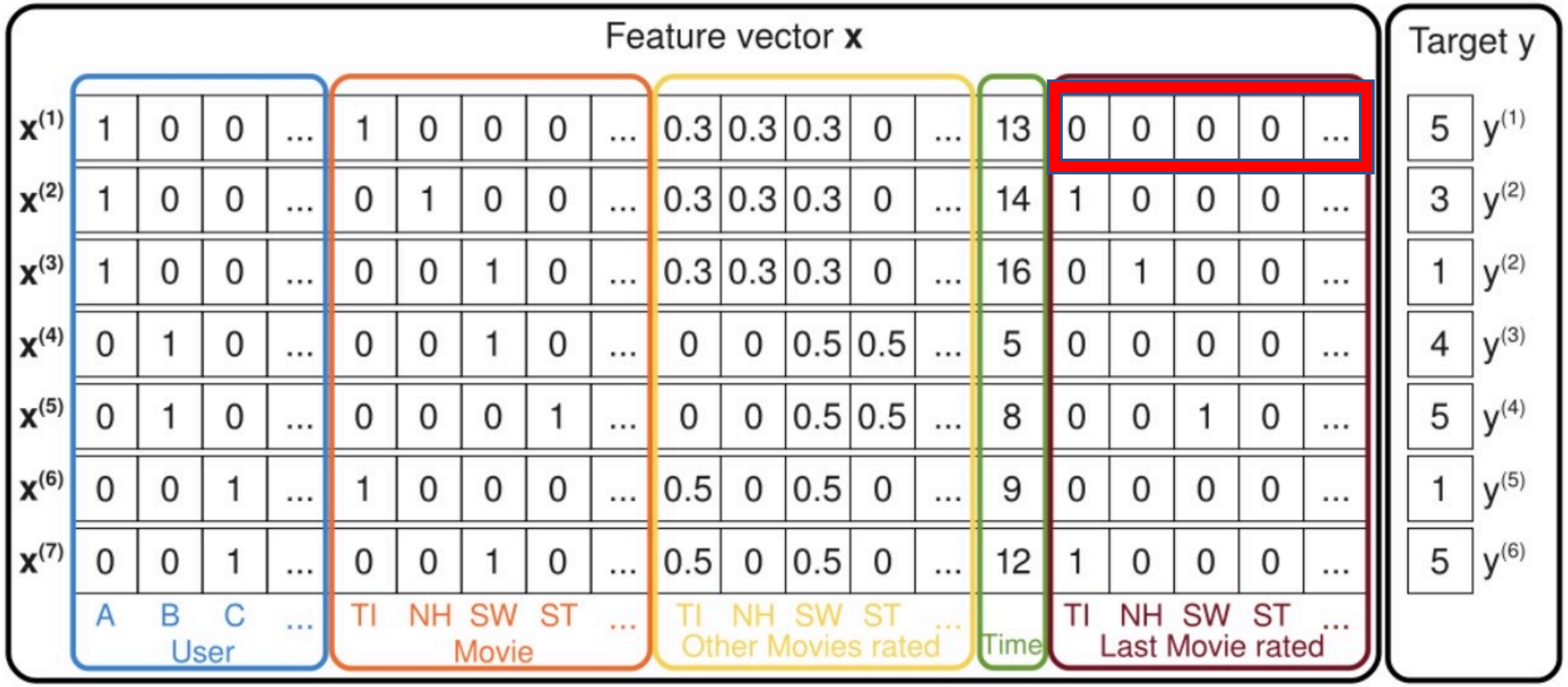


FM input representation

data S = {(Alice, Titanic, 2010-1, 5), (Bob, Star Wars, 2010-2, 3) … }

2. Prediction Under Sparsity

More Categorical Variables make data more Sparse



FM input representation

data S = {(Alice, Titanic, 2010-1, 5), (Bob, Star Wars, 2010-2, 3) … }

2. Prediction Under Sparsity

More Categorical Variables make data more Sparse

	Feature vector **x**																					Target y	
	User				Movie					Other Movies rated					Time	Last Movie rated							
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...	Time	TI	NH	SW	ST	...			
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...		5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...		3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...		1	$y^{(2)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...		4	$y^{(3)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...		5	$y^{(4)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...		1	$y^{(5)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...		5	$y^{(6)}$

FM input representation

data S = {(Alice, Titanic, 2010-1, 5), (Bob, Star Wars, 2010-2, 3) … }

3. Factorization Machine

Degree d=2

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

w_0 : $global\ bias$

w_i : $strength\ of\ i-th\ variable$

$<v_i, v_j>$ $interaction\ between\ the\ i-th\ and\ j-th\ variable$

x_i : $i-th\ feature\ vector$

And $\langle \cdot, \cdot \rangle$ is the dot product of two vectors of size k:

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle := \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}$$

3. Factorization Machine

Feature vector **x**

	User				Movie					Other Movies rated					Time	Last Movie rated					Target y	
	A	B	C	...	TI	NH	SW	ST	...	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	...	13	0	0	0	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1	$y^{(2)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4	$y^{(3)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5	$y^{(4)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1	$y^{(5)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5	$y^{(6)}$

FM input representation

$$\boldsymbol{x}^{(\mathbf{1})} = [x_{1,} x_2, x_3 \dots x_n] = [1, 0, 0, \dots, 1, 0, 0, 0, \dots, 0.33, 0.33, 0.33, 0, \dots, 13, \dots] = [\boldsymbol{v}_{\mathbf{1},} \boldsymbol{v}_{\mathbf{2}}, \boldsymbol{v}_{\mathbf{3}} \dots \boldsymbol{v}_{\boldsymbol{n}}]$$

$$\boldsymbol{v}_{\boldsymbol{i}} = [v_{1,} v_2, v_3 \dots v_k]$$

3. Factorization Machine

MF, FM

Coefficient 대신 특성 하나하나를 latent space(k 차원)에 mapping, 그 공간에서의 내적을 통해 상호관계를 계산하는 것

user와 item 사이의 상관관계

+ degree가 2여도 변수 간의 1차 상관관계($W_i \times x_i$)와 2차 상관관계 ($< (v_i , v_j) > x_i \, x_j$) 를 모두 고려함

3. Factorization Machine - Expressiveness

각각의 변수의 차원 증가 → 어떠한 Interaction Matrix도 표현 가능

Positive Definite Matrix $W = V \cdot V^T$

Sparse data → 작은 차원의 K를 선택하여 일반화 성능을 높힘

3. Factorization Machine

	Feature vector **x**																					Target y	
	User				Movie					Other Movies rated					Time	Last Movie rated							
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...	Time	TI	NH	SW	ST	...			
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...		5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...		3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...		1	$y^{(2)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...		4	$y^{(3)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...		5	$y^{(4)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...		1	$y^{(5)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...		5	$y^{(6)}$

FM input representation

Interaction between A(Alice) and ST(Star Trek)
Direct estimate, 0 (no interaction)

3. Factorization Machine

Feature vector **x**

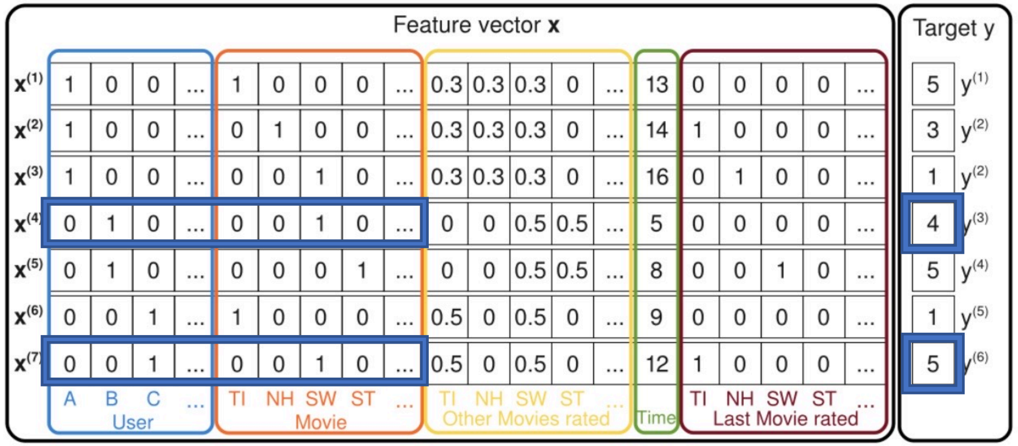
	User				Movie					Other Movies rated					Time	Last Movie rated					Target y	
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...	Time	TI	NH	SW	ST	...		
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1	$y^{(2)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4	$y^{(3)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5	$y^{(4)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1	$y^{(5)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5	$y^{(6)}$

FM input representation

Interaction between A(Alice) and ST(Star Trek)
Direct estimate, 0 (no interaction)

Factorized interaction
$<v_A, v_{ST}>$

3. Factorization Machine



FM input representation

$$<v_B, v_{SW}> \approx <v_C, v_{SW}>$$

$$v_B \approx v_C$$

3. Factorization Machine

Feature vector **x** / Target y

	User				Movie					Other Movies rated					Time	Last Movie rated					Target y	
	A	B	C	...	TI	NH	SW	ST	...	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	...	13	0	0	0	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1	$y^{(2)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4	$y^{(3)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5	$y^{(4)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1	$y^{(5)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5	$y^{(6)}$

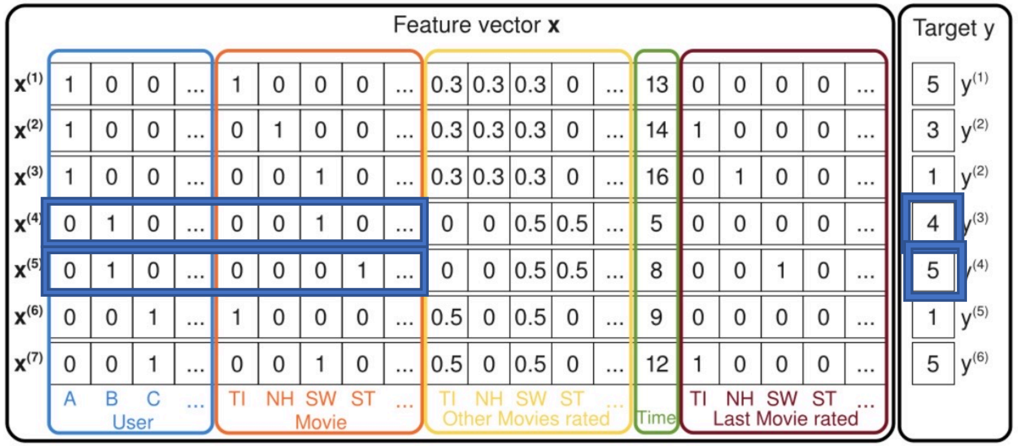
FM input representation

$$<v_A, v_{TI}> \not\approx <v_C, v_{TI}>$$

$$<v_A, v_{SW}> \not\approx <v_C, v_{SW}>$$

$$v_A \not\approx v_C$$

3. Factorization Machine

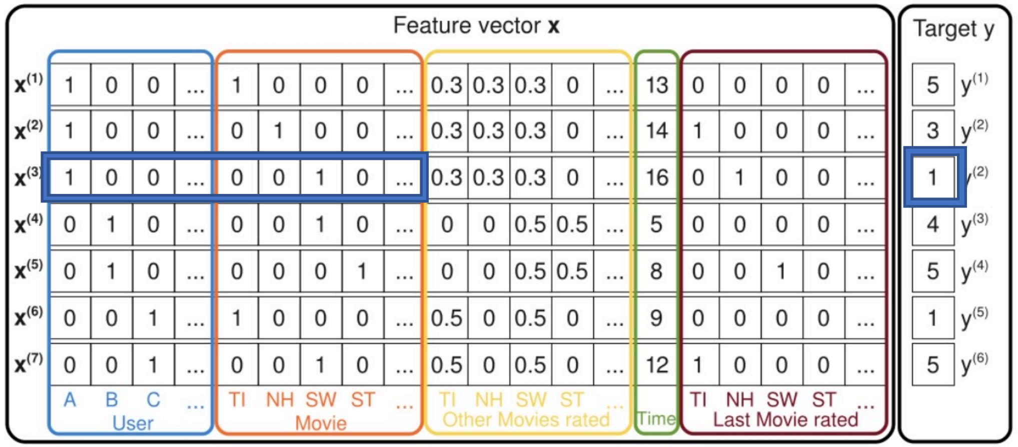


FM input representation

$$<\boldsymbol{v_B}, \boldsymbol{v_{SW}}> \approx <\boldsymbol{v_B}, \boldsymbol{v_{ST}}>$$

$$\boldsymbol{v_{SW}} \approx \boldsymbol{v_{ST}}$$

3. Factorization Machine



FM input representation

In total,

$$<v_A, v_{ST}> \approx <v_A, v_{SW}>$$

3. Factorization Machine – Linear Computation

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

$O(kn^2)$

$$\begin{aligned}
&\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j \\
=&\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j - \frac{1}{2} \sum_{i=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_i \rangle \, x_i \, x_i \\
=&\frac{1}{2} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{f=1}^{k} v_{i,f} \, v_{j,f} \, x_i \, x_j - \sum_{i=1}^{n} \sum_{f=1}^{k} v_{i,f} \, v_{i,f} \, x_i \, x_i \right) \\
=&\frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f} \, x_i \right) \left(\sum_{j=1}^{n} v_{j,f} \, x_j \right) - \sum_{i=1}^{n} v_{i,f}^2 \, x_i^2 \right) \\
=&\frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f} \, x_i \right)^2 - \sum_{i=1}^{n} v_{i,f}^2 \, x_i^2 \right)
\end{aligned}$$

$O(kn), actually\ O(k\overline{m_D})$

3. Factorization Machine- as predictors

FM can be applied to a variety of prediction tasks.

Regression : $\hat{y}(x)$ can be used directly as the predictor

Binary classification : use sign of $\hat{y}(x)$

Ranking : vectors x are ordered by score of $\hat{y}(x)$

In all cases, L2 regularization terms are added to prevent overfitting.

3. Factorization Machine- learning FM

FM has a model equation that is linearly calculated.

→Learning by Gradient Descent (SGD)

$$\frac{\partial}{\partial\theta}\hat{y}(\mathbf{x}) = \begin{cases} 1, & \text{if } \theta \text{ is } w_0 \\ x_i, & \text{if } \theta \text{ is } w_i \\ x_i \sum_{j=1}^{n} v_{j,f} x_j - v_{i,f} x_i^2, & \text{if } \theta \text{ is } v_{i,f} \end{cases}$$

Can be precomputed (independent to i)

O(kn) -> O(km(x))

3. Factorization Machine- Generalization

d-way FM

$$\hat{y}(x) := w_0 + \sum_{i=1}^{n} w_i \, x_i$$
$$+ \sum_{l=2}^{d} \sum_{i_1=1}^{n} \cdots \sum_{i_l=i_{l-1}+1}^{n} \left(\prod_{j=1}^{l} x_{i_j} \right) \left(\sum_{f=1}^{k_l} \prod_{j=1}^{l} v_{i_j,f}^{(l)} \right) \quad (5)$$

$$O(kn^d) \rightarrow O(kn)$$

3. Factorization Machine- Summary

FM uses factorized interactions to model interactions between the values of x

Advantages

1. Interactions between values can be estimated even under high sparsity.
2. The time for prediction and learning is linear →

\# of parameter is linear, can use SGD, variety of of loss function

4. FMs VS SVMs

SVM with Linear Kernel

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i \, x_i, \quad w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^n \qquad (7)$$

= FM of degree d=1

SVM with Polynomial Kernel

$$\hat{y}(\mathbf{x}) = w_0 + \sqrt{2} \sum_{i=1}^{n} w_i \, x_i + \sum_{i=1}^{n} w_{i,i}^{(2)} x_i^2 + \sqrt{2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{i,j}^{(2)} \, x_i \, x_j \qquad (9)$$

$$\hat{y}(\mathbf{x}) = w_0 + \sqrt{2}(w_u + w_i) + w_{u,u}^{(2)} + w_{i,i}^{(2)} + \sqrt{2} w_{u,i}^{(2)}$$

Sparse data

4. FMs VS SVMs

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

$$\hat{y}(\mathbf{x}) = w_0 + \sqrt{2}(w_u + w_i) + w_{u,u}^{(2)} + w_{i,i}^{(2)} + \sqrt{2}w_{u,i}^{(2)}$$

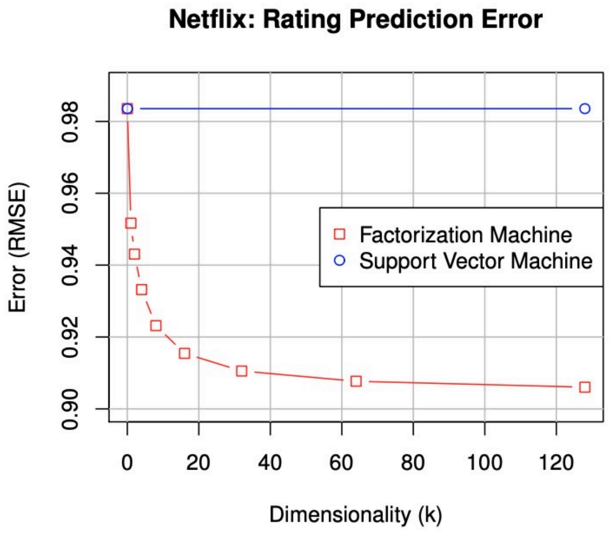


Fig. 2. FMs succeed in estimating 2-way variable interactions in very sparse problems where SVMs fail (see section III-A3 and IV-B for details.)

SVM fail to find the interaction

5. FMs VS Other Factorization Models

- Matrix and Tensor Factorization

$$n := |U \cup I|, \quad x_j := \delta\,(j = i \vee j = u)$$

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

- SVD++

$$n := |U \cup I \cup L|, \quad x_j := \begin{cases} 1, & \text{if } j = i \vee j = u \\ \frac{1}{\sqrt{|N_u|}}, & \text{if } j \in N_u \\ 0, & \text{else} \end{cases}$$

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

5. FMs VS Other Factorization Models

- PITF(Pairwise Interaction Tensor Factorization)

$$n := |U \cup I \cup T|, \quad x_j := \delta\,(j = i \vee j = u \vee j = t) \quad (13)$$

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

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

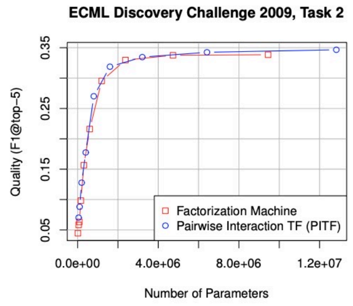


Fig. 3. Recommendation quality of a FM compared to the winning PITF model [3] of the ECML/PKDD Discovery Challenge 2009. The quality is plotted against the number of model parameters.

- FPMC(Factorized Personalized Markov Chains)

$$n := |U \cup I \cup L|, \quad x_j := \begin{cases} 1, & \text{if } j = i \vee j = u \\ \frac{1}{|B^u_{t-1}|}, & \text{if } j \in B^u_{t-1} \\ 0, & \text{else} \end{cases} \quad (15)$$

$$\hat{y}(\mathbf{x}) = w_0 + w_u + w_i + \langle \mathbf{v}_u, \mathbf{v}_i \rangle + \frac{1}{|B^u_{t-1}|} \sum_{l \in B^u_{t-1}} \langle \mathbf{v}_i, \mathbf{v}_l \rangle$$

$$+ \frac{1}{|B^u_{t-1}|} \sum_{l \in B^u_{t-1}} \left(w_l + \langle \mathbf{v}_u, \mathbf{v}_l \rangle + \frac{1}{|B^u_{t-1}|} \sum_{l' \in B^u_{t-1}, l' > l} \langle \mathbf{v}_l, \mathbf{v}'_l \rangle \right)$$

$$\hat{y}(\mathbf{x}) = w_i + \langle \mathbf{v}_u, \mathbf{v}_i \rangle + \frac{1}{|B^u_{t-1}|} \sum_{l \in B^u_{t-1}} \langle \mathbf{v}_i, \mathbf{v}_l \rangle$$

6. Summary

Even if the data is sparse, prediction performs well

It has linear complexity

It is general predictor

It is Identical or very similar to many other models

7. Codes

Tensorflow
Dataset : WDBC
K = 10, learning rate = 0.01

```python
class FM(tf.keras.Model):
    def __init__(self):
        super(FM, self).__init__()

        # 모델의 파라미터 정의
        self.w_0 = tf.Variable([0.0])
        self.w = tf.Variable(tf.zeros([p]))
        self.V = tf.Variable(tf.random.normal(shape=(p, k)))

    def call(self, inputs):
        linear_terms = tf.reduce_sum(tf.math.multiply(self.w, inputs), axis=1)

        interactions = 0.5 * tf.reduce_sum(tf.math.pow(tf.matmul(inputs, self.V), 2)
        - tf.matmul(tf.math.pow(inputs, 2), tf.math.pow(self.V, 2)),1,keepdims=False)

        y_hat = tf.math.sigmoid(self.w_0 + linear_terms + interactions)

        return y_hat
```

```python
def train_on_batch(model, optimizer, accuracy, inputs, targets):
    with tf.GradientTape() as tape:
        y_pred = model(inputs)
        loss = tf.keras.losses.binary_crossentropy(from_logits=False,
                                                   y_true=targets,
                                                   y_pred=y_pred)

    # loss를 모델의 파라미터로 편미분하여 gradients를 구한다.
    grads = tape.gradient(target=loss, sources=model.trainable_variables)

    # apply_gradients()를 통해 processed gradients를 적용한다.
    optimizer.apply_gradients(zip(grads, model.trainable_variables))

    # accuracy: update할 때마다 정확도는 누적되어 계산된다.
    accuracy.update_state(targets, y_pred)

    return loss
```

7. Codes

```python
def train(epochs):
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y)

    train_ds = tf.data.Dataset.from_tensor_slices(
        (tf.cast(X_train, tf.float32), tf.cast(Y_train, tf.float32))).shuffle(500).batch(8)

    test_ds = tf.data.Dataset.from_tensor_slices(
        (tf.cast(X_test, tf.float32), tf.cast(Y_test, tf.float32))).shuffle(200).batch(8)

    model = FM()
    optimizer = tf.keras.optimizers.SGD(learning_rate=0.01)
    accuracy = BinaryAccuracy(threshold=0.5)
    loss_history = []
```

```
스텝 048에서 누적 정확도: 0.9222
FM 테스트 정확도: 0.8721
SVM 테스트 정확도: 0.8509
```

7. Codes

```
ratings_df = pd.read_csv("ratings.csv", encoding='utf-8')
ratings_df.drop('timestamp', inplace=True, axis=1) # timestamp 필요 없어보임
movies_df = pd.read_csv("movies.csv", encoding='utf-8')
movies_df = movies_df.set_index("movieId") # movieId를 index로 변환함
dummy_genre_df =  movies_df['genres'].str.get_dummies(sep='|')
movies_df['year'] = movies_df["title"].str.extract('(\(\d\d\d\d\))')

movies_df['year'] = movies_df['year'].astype('str')
movies_df['year'] = movies_df['year'].map(lambda x: x.replace("(", "").replace(")", ""))
movies_df['year'] = movies_df['year'].astype("float32")
movies_df.dropna(axis=0)
movies_df.drop('title',axis=1,inplace=True)


bins = list(range(1900, 2021, 20))
labels = [x for x in range(len(bins) - 1)] #[0, 1, 2, 3, 4, 5]

movies_df['year_level'] = pd.cut(movies_df['year'], bins, right=False, labels=labels) # movies_df['year']을 bins
movies_df.drop('year', inplace=True, axis=1)

threshold = 10
over_threshold = ratings_df.groupby('movieId').size() >= threshold # 만약 영화가 10개 이상의 rating이 달리면 true, 아니면

ratings_df['over_threshold'] = ratings_df['movieId'].map(lambda x: over_threshold[x])

ratings_df = ratings_df[ratings_df["over_threshold"] == True]
ratings_df.drop("over_threshold", axis=1, inplace=True)
# print(ratings_df)
random_idx = np.random.permutation(len(ratings_df))# 순서를 무작위로 섞는다
shuffled_df = ratings_df.iloc[random_idx] # 섞는 순서로 배치한다

X = pd.concat([
            pd.get_dummies(shuffled_df['userId'], prefix="user"), # 모든 user에 대한 차원
            pd.get_dummies(shuffled_df['movieId'], prefix="movie"), # 모든 영화에 대한 차원
            shuffled_df['movieId'].apply(lambda x: dummy_genre_df.loc[x]), # 모든 장르에 대한 차원
            shuffled_df['movieId'].apply(lambda x: movies_df.loc[x]["year_level"]).rename('year_level'), # 연도에
], axis=1)
```

Thank you