2016 SIAM
International Conference
on DATA MINING

Kernelized Matrix Factorization for Collaborative Filtering

Xinyue Liu*, Charu Aggarwal† , Yu-Feng Li‡
Xiangnan Kong* , Xinyuan Sun*, Saket Sathe†

* Worcester Polytechnic Institute ‡ Nanjing University
† IBM T.J. Watson Research Center

Collaborative Filtering Problem

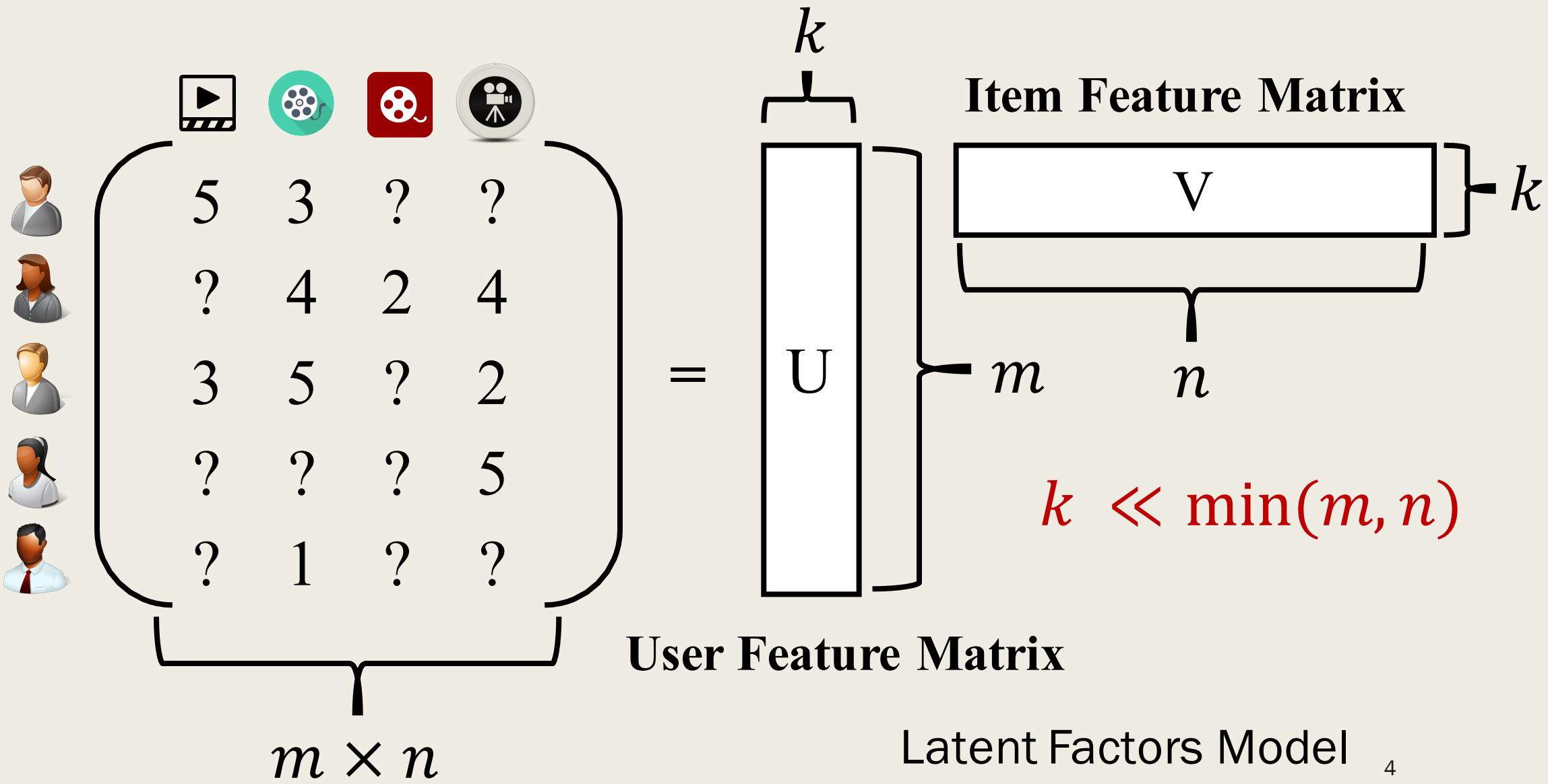
- *Given users and items in a recommender system.*
- *Assume each user can rate each item only once.*
- *Then we can cast the user-item feedbacks into a matrix.*
- *The matrix is always very sparse.*
  - *Netflix Prize dataset (1.18%)*
  - *MovieLens dataset (6.3%)*
- *The task is to estimate the unknown ratings (?) based on the observed ones.*

Applications

- *Netflix: Movie Recommendations*
- *Amazon: Merchandise Recommendations*
- *Pandora: Music Recommendations*
- *Yelp: Restaurant Recommendations*

- *Educational system, dating website, everything where you have choices…*

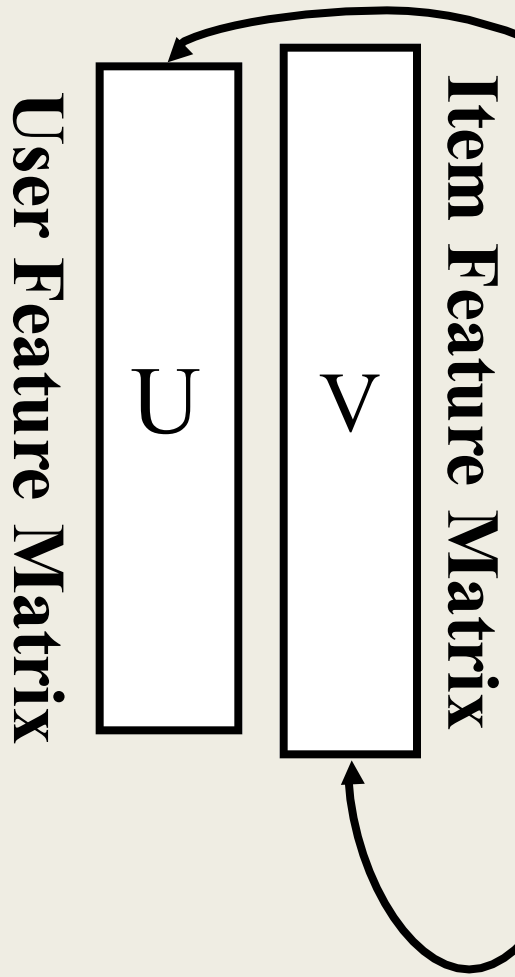
Matrix Factorization



$$k \ll \min(m, n)$$

Latent Factors Model

The Latent Factors



K = 5

	Action	Sci-Fi	Sports	Musicals	Romantic	
Bob	5	3	4	-5	-2	u_1
Alice	2	-1	-2	5	4	u_2
	0.7	0.0	0.1	0.0	0.2	v_1

Will  like  ?

$$u_1 v_1^T = (5 \times 0.7) + (4 \times 0.1) + (-2 \times 0.2) = 3.5$$

Will  like  ?

$$u_2 v_1^T = (2 \times 0.7) + (-2 \times 0.1) + (4 \times 0.2) = 2$$

Find the Latent Factors

- The goal is to find U and V such that:

$$\min_{U,V} \sum_{(i,j)\in R} (r_{ij} - u_i v_j^T)^2$$

Can be solved via gradient descent

users

items

1		3		5				5		4	
		5	4		4				2	1	3
2	4			1	2			4	3	5	
	2	4	5			4			2		
		4	4	3	4				2		5
1		3		3		2			4		

≈

factors

.1	.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
.1	.7	.3

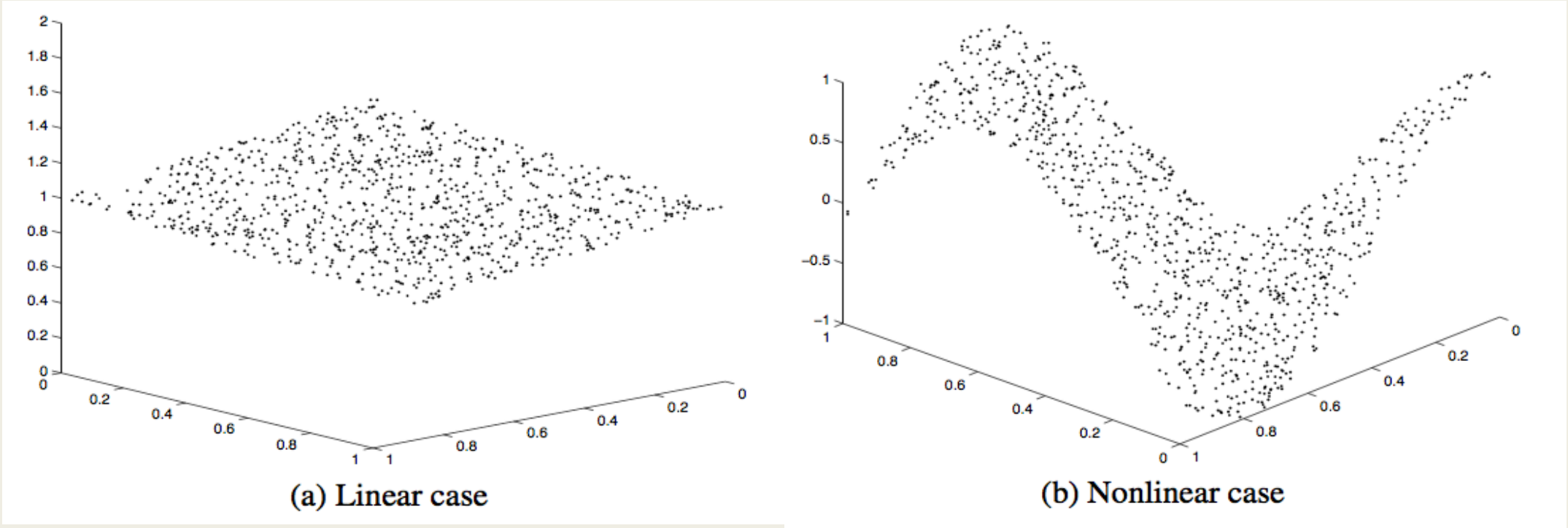
users

factors

items

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

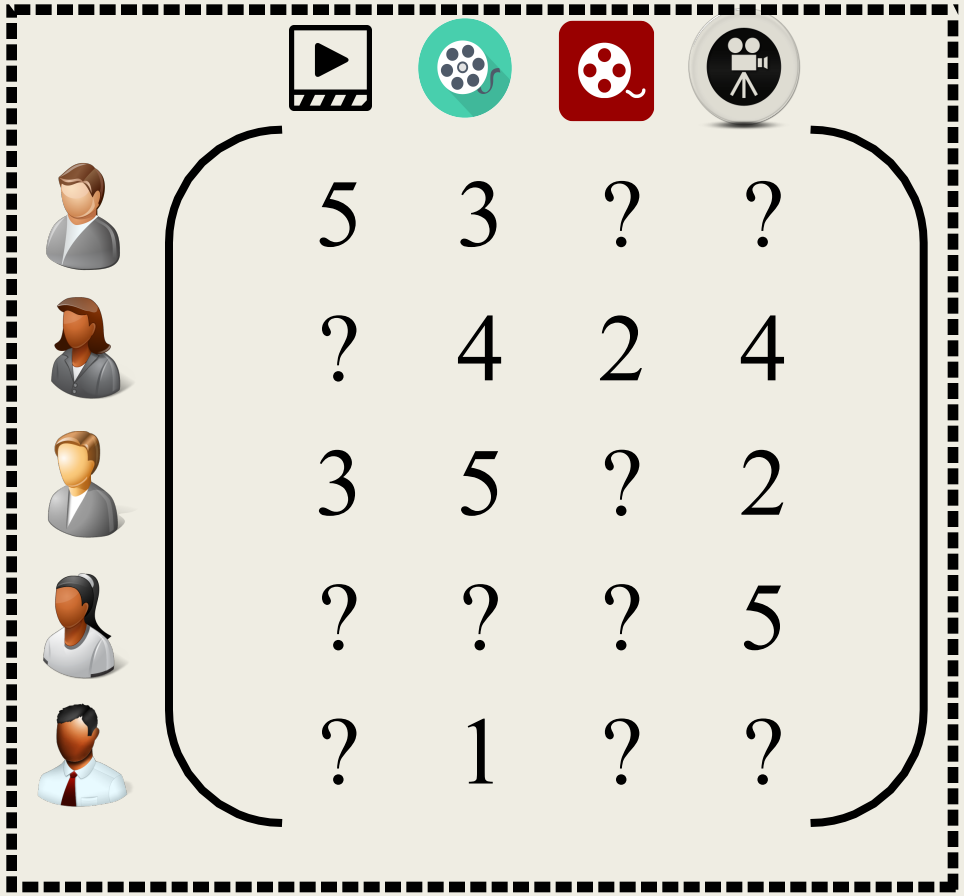
Limitation of Matrix Factorization



$$\min_{U,V} \sum_{(i,j)\in R} (r_{ij} - \underbrace{u_i v_j^T}_{linear})^2$$

Data can distributed on a non-linear hyper-plane.

MF is unable to capture the nonlinearity of the data!



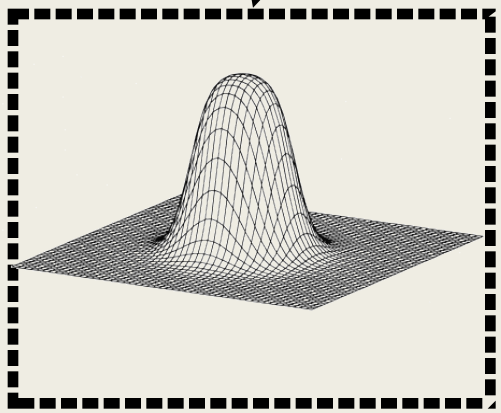
Original Space

=

$\phi(V)$

$\phi(U)$

Hilbert Space
(high-dimensional space)



Kernel

Implicitly define $\phi(\cdot)$

Mapping function

Kernelized MF

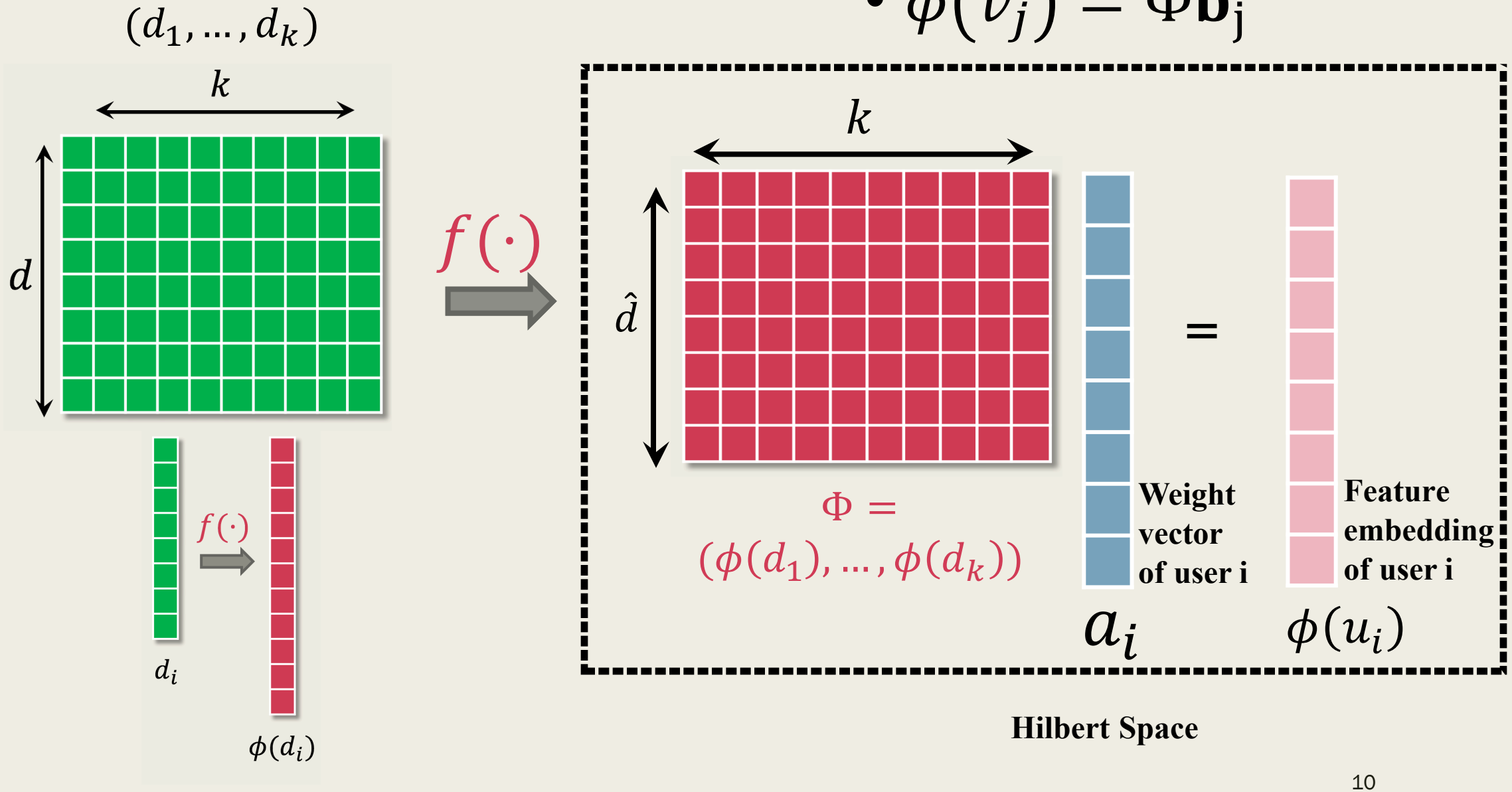
Challenges

$$\min_{\phi(U),\phi(V)} \sum_{(i,j)\in R} \left(r_{ij} - \phi(u_i)^T \phi(v_j)\right)^2$$

1. *It is difficult to minimize the objective function over* ***implicitly defined*** $\phi(U)\ and\ \phi(V)$
2. *It is difficult to choose* ***a good kernel function*** *(and kernel parameters) for the data*
3. *It is difficult to use only* ***a single kernel*** *to capture the complex non-linearity embedded in the data*

Dictionary Vectors

- $\phi(u_i) = \Phi \boldsymbol{a}_i$
- $\phi(v_j) = \Phi \mathbf{b}_j$



Dictionary Vectors Cont.

$$\min_{\phi(U),\phi(V)} \sum_{(i,j)\in R} \left(r_{ij} - \phi(u_i)^T \phi(v_j)\right)^2$$

$$\phi(u_i) = \Phi \boldsymbol{a}_i$$
$$\phi(v_j) = \Phi \mathbf{b}_j$$

$$\min_{\mathbf{A},\mathbf{B}} \sum_{(i,j)\in R} \left(r_{ij} - \boldsymbol{a}_i \Phi^T \Phi \mathbf{b}_j\right)^2$$

$$\mathbf{K} = \Phi^T \Phi$$

$$\min_{\mathbf{A},\mathbf{B}} \sum_{(i,j)\in R} \left(r_{ij} - \boldsymbol{a}_i \mathbf{K} \mathbf{b}_j\right)^2$$

$$\mathbf{K} = \begin{pmatrix} \phi(d_1)^T\phi(d_1) & \cdots & \phi(d_1)^T\phi(d_k) \\ \vdots & \vdots & \vdots \\ \phi(d_k)^T\phi(d_1) & \cdots & \phi(d_k)^T\phi(d_k) \end{pmatrix}$$

Gram Matrix

Kernel function

$$K_{ij} = \phi(d_i)^T \phi(d_j) = f(d_i, d_j)$$

* $\mathbf{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m), \mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$

$$f(x_1, x_2) = \phi(x_1)^T \phi(x_2) = (x_1^T x_2 + c)^d$$

Polynomial Kernel

Optimization

L2-norm regularizations

$$\min_{\mathbf{A},\mathbf{B}} \sum_{(i,j)\in R} \left(r_{ij} - \boldsymbol{a}_i^T \mathbf{K} \boldsymbol{b}_j\right)^2 + \lambda(\boldsymbol{a}_i^T \mathbf{K} \boldsymbol{a}_i + \boldsymbol{b}_j^T \mathbf{K} \boldsymbol{b}_j)$$

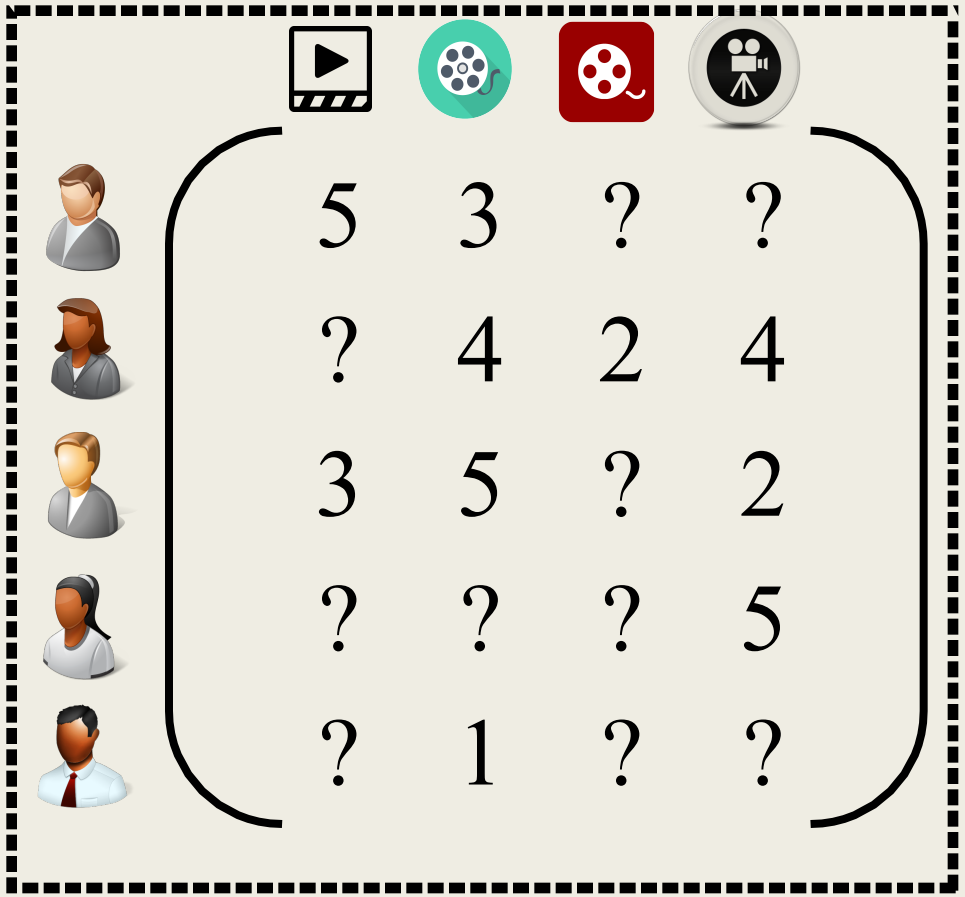
$$\min_{B} \sum_{(i,j)\in R} \left(r_{ij} - \boldsymbol{a}_i^T \mathbf{K} \boldsymbol{b}_j\right)^2 + \lambda \boldsymbol{b}_j^T \mathbf{K} \boldsymbol{b}_j$$

Close Form Solution: $\widehat{\boldsymbol{b}_j} = (\mathbf{K}^T \mathbf{A}\mathbf{A}^T \mathbf{K} + \lambda \mathbf{K})^{-1} \mathbf{K}^T \mathbf{A} r_j$

Challenges

$$\min_{\phi(U),\phi(V)} \sum_{(i,j)\in R} \left(r_{ij} - \phi(u_i)^T \phi(v_j)\right)^2$$

1. *It is difficult to minimize the objective function over implicitly defined* $\phi(U)\ and\ \phi(V)$
2. *It is difficult to choose* ***a good kernel function*** *(and kernel parameters) for the data*
3. *It is difficult to use only* ***a single kernel*** *to capture all the non-linearity embedded in the data*



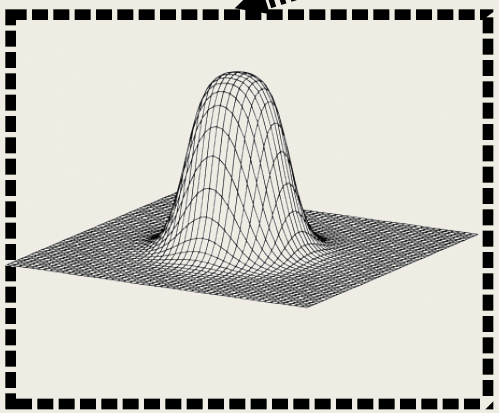
=

U

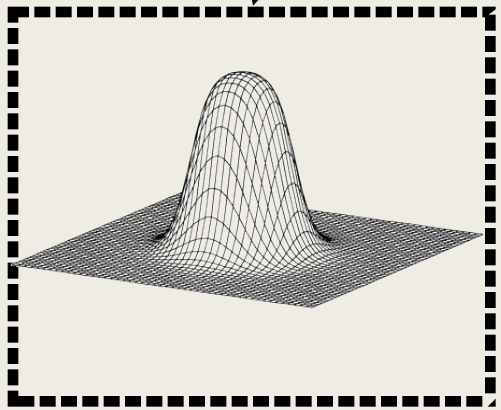
V

Hilbert Space
(high-dimensional space)

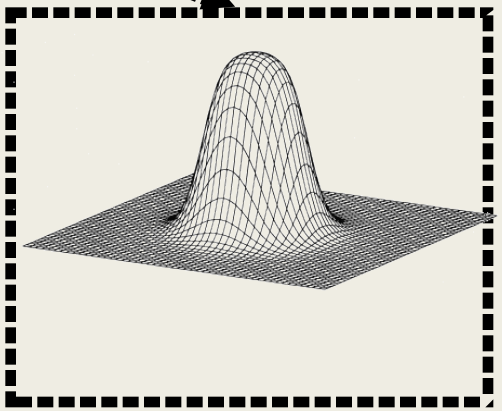
Original Space



+



+



Kernel 1 **Kernel 2** **Kernel 3**

Multi-Kernel MF

Combine Multiple Kernels

- Given p different kernels, we can compute p Gram matrices:

$$\{\mathbf{K}_1, \dots, \mathbf{K}_\mathrm{p}\}$$

- We want to learn the kernel weight vector for p kernels:

$$\boldsymbol{\mu} = (\mu_1, \dots, \mu_p)^T$$

$$\min_{\boldsymbol{\mu}} \sum_{(i,j)\in R} \left(r_{ij} - \sum \mu_u \boldsymbol{a}_i^T \mathbf{K}_u \boldsymbol{b}_j\right)^2 + \lambda(\boldsymbol{b}_j^T \sum \mu_u \mathbf{K}_u \boldsymbol{b}_j + \boldsymbol{a}_i^T \sum \mu_u \mathbf{K}_u \boldsymbol{a}_i)$$

Combine Multiple Kernels Cont.

$$\min_{\mu} \sum_{(i,j)\in R} \left(r_{ij} - \sum \mu_u \boldsymbol{a}_i^T \mathbf{K}_u \boldsymbol{b}_j\right)^2 + \lambda\left(\boldsymbol{b}_j^T \sum \mu_u \mathbf{K}_u \boldsymbol{b}_j + \boldsymbol{a}_i^T \sum \mu_u \mathbf{K}_u \boldsymbol{a}_i\right)$$

$$\boldsymbol{v}_{ij} = \left(\boldsymbol{a}_i^T \mathbf{K}_1 \boldsymbol{b}_j, \ldots, \boldsymbol{a}_i^T \mathbf{K}_p \boldsymbol{b}_j\right)^T$$

$$\boldsymbol{y}_{ij} = \left(\boldsymbol{a}_i^T \mathbf{K}_1 \boldsymbol{a}_i + \boldsymbol{b}_j^T \mathbf{K}_1 \boldsymbol{b}_j, \ldots, \boldsymbol{a}_i^T \mathbf{K}_1 \boldsymbol{a}_i + \boldsymbol{b}_j^T \mathbf{K}_1 \boldsymbol{b}_j\right)^T$$

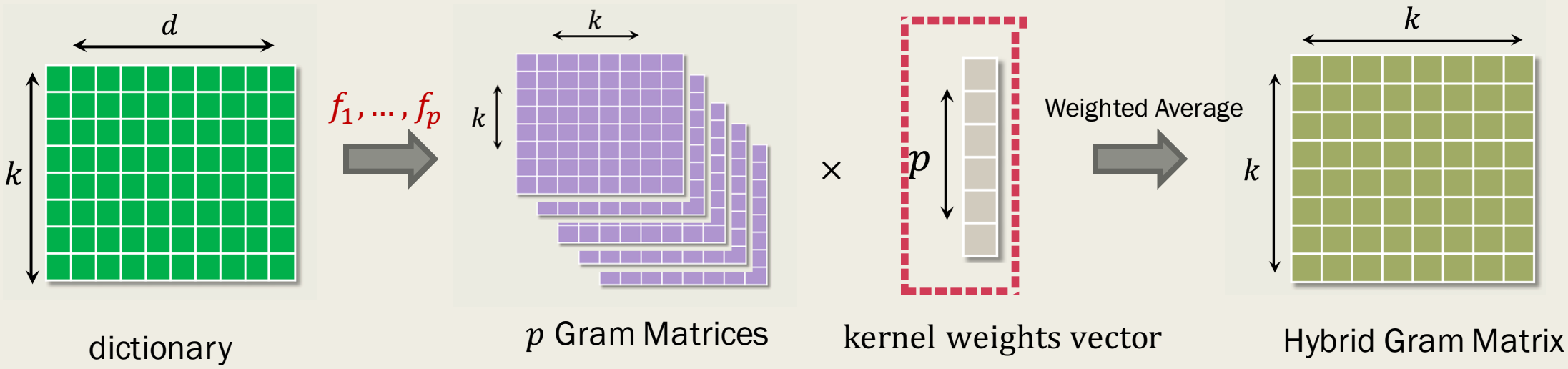
$$\mathbf{Y} = \sum_{u,i} \boldsymbol{v}_{ij} \boldsymbol{v}_{ij}^T$$

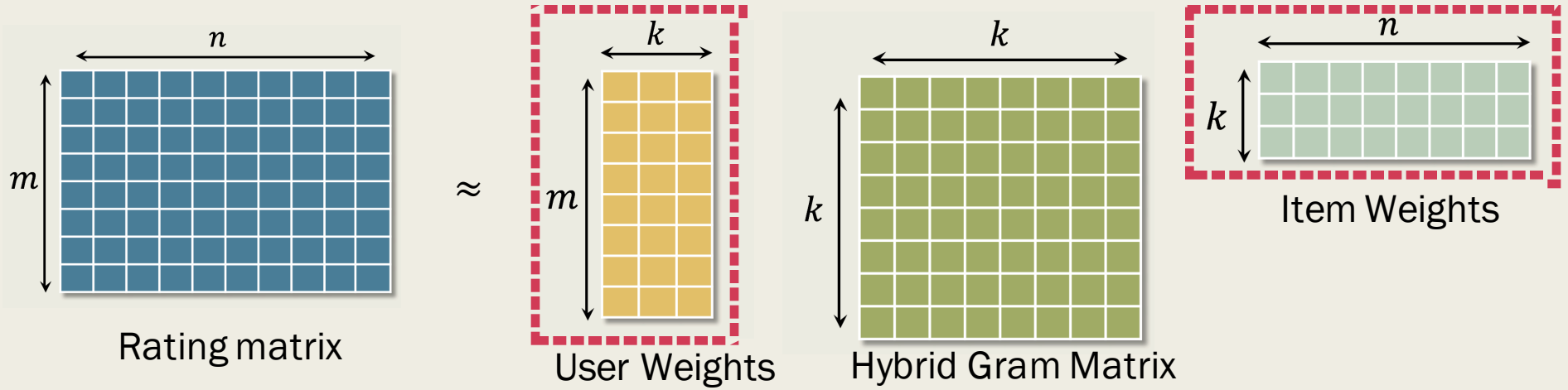
$$\boldsymbol{Z} = \sum_{u,i} \left(\lambda - 2r_{ij}\right) \boldsymbol{y}_{ij}$$

Quadratic Programming

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{\mu}^T \boldsymbol{Y} \boldsymbol{\mu} + \boldsymbol{Z} \boldsymbol{\mu} \\ \text{subject to} \quad & \boldsymbol{\mu} \succcurlyeq 0 \\ & \mathbf{1}_{\mathrm{p}}^{\mathrm{T}} \boldsymbol{\mu} = 1 \end{aligned}$$

Multiple Kernel Matrix Factorization





Datasets

Table 2: Summary of Datasets

Dataset	# of users	# of items	Density (%)
MovieLens	6,040	3,900	6.3
Jester	73,421	100	55.8
Flixster	147,612	48,784	0.11
Dating Agency	135,359	168,791	0.76
Yahoo Music	1,948,882	98,211	0.006
ASSISTments	46,627	179,084	0.073

Experiments

1. Sample a 1000×1000 matrix for each dataset
2. Hold 1 rating for each user for test, the remaining as training data
3. Repeat this partition 5 times for each dataset
4. Average of RMSE on test data are reported

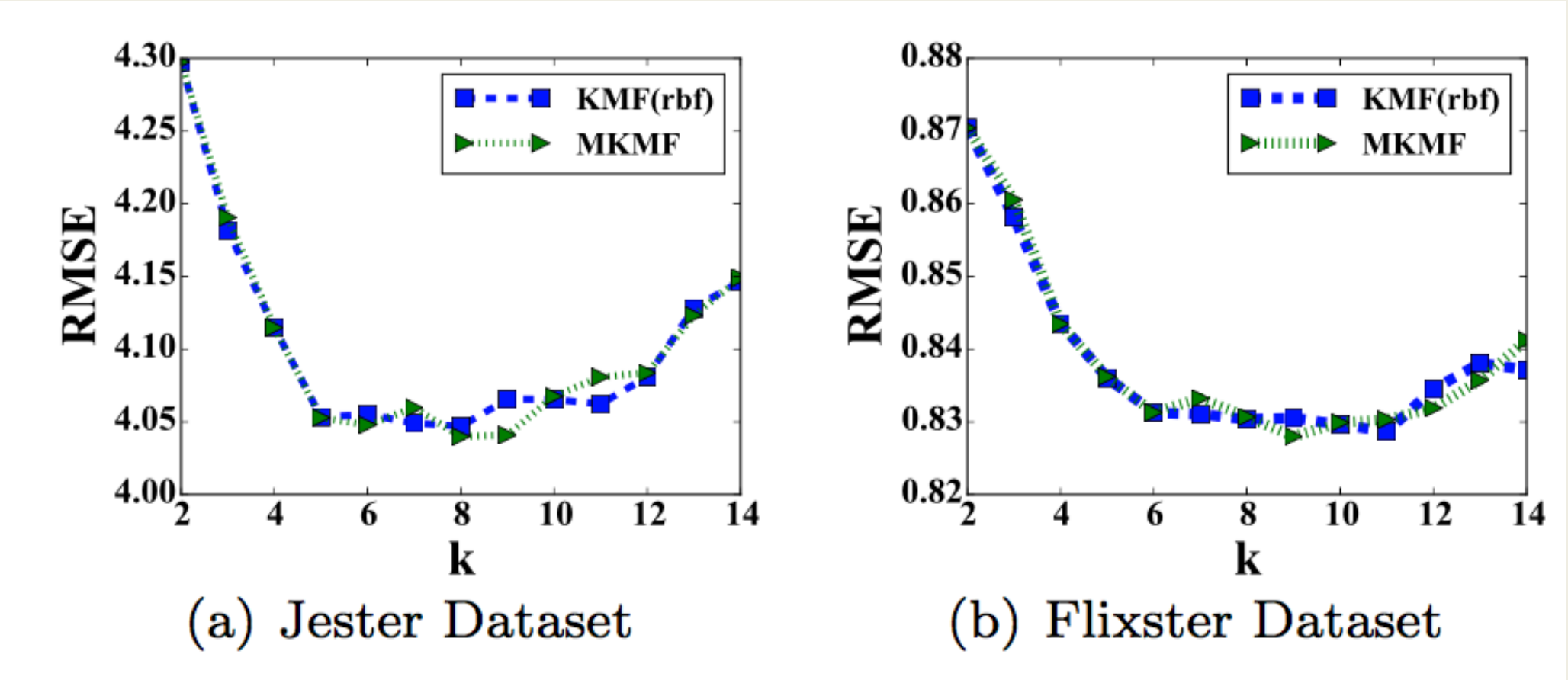
$$\text{RMSE} = \sqrt{\frac{\sum_{(u,i)\in\Omega} \left(r_{ui} - \hat{r}_{ui}\right)^2}{|\Omega|}}$$

Results

Methods	Dataset												Ave. Rank
	Flixster		Yahoo Music		MovieLens		Jester		Dating		ASSISTments		
MKMF	0.8270	(1)	18.3036	(1)	0.8223	(1)	4.0398	(1)	1.6697	(1)	0.7920	(2)	1.1667
KMF (LINEAR)	0.8290	(4)	18.3734	(3)	0.8241	(3.5)	4.0471	(2.5)	1.6804	(4)	0.7933	(4)	3.5
KMF (POLY)	0.8289	(3)	18.3766	(4)	0.8241	(3.5)	4.0472	(4)	1.6797	(3)	0.7934	(5)	3.75
KMF (RBF)	0.8292	(5)	18.3883	(5)	0.8246	(5)	4.0471	(2.5)	1.6701	(2)	0.7927	(3)	3.75
MF	0.8286	(2)	18.3214	(2)	0.8235	(2)	4.0549	(5)	1.6849	(5)	0.8001	(6)	3.6667
SVD	0.9441	(9)	26.2255	(9)	0.9406	(7)	4.2794	(6)	1.7920	(7)	0.8919	(9)	7.8333
IVC-COS	0.9223	(7)	22.9477	(7)	1.0016	(8)	4.6137	(8)	2.7052	(8)	0.8106	(7)	7.5
IVC-PEARSON	0.9226	(8)	22.9486	(8)	1.0020	(9)	4.6142	(9)	2.8209	(9)	0.8446	(8)	8.5
AVG	0.9006	(6)	22.4159	(6)	0.8887	(6)	4.3867	(7)	1.7349	(6)	0.7658	(1)	5.3333

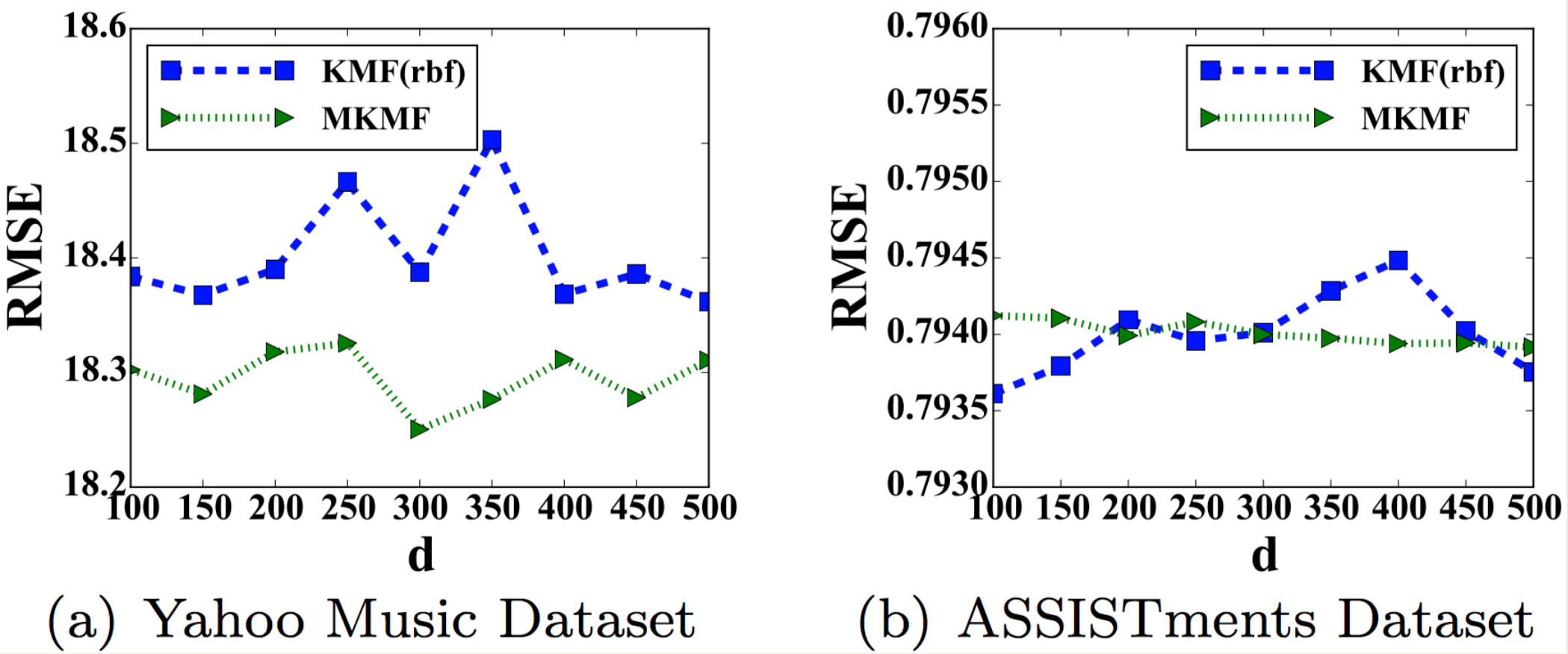
- Our methods are in red box (KMF & MKMF)
- Kernelized MF is better than other baselines (green box)
- Multi-Kernel MF is better than single kernel (blue box)

Parameter Study (k, # of latent factors)



- Small k (<=5) could not fit the data very well
- Large k (>12) could be over-fitting
- Cross-validate on each dataset for the best choice of k

Parameter Study (d, the dimension of dictionary vectors)



- Not sensitive on d
- Any $d > 100$ works just fine

Conclusion & Future Works

- Conclusion:
  - Our proposed KMF and MKMF methods both exploit **the underlying nonlinear correlations** among rows (users) and columns (items) in the rating matrix.
  - KMF/MKMF **improves the overall performance** of predicting the unobserved ratings compared to state-of-art baselines.

- Future Works:
  - Alternative regularizations (e.g. **L2 norm**) can be applied on the weight learning process.
  - The kernel (Gram) matrix might be **learned directly** from the data without the help of dictionary.
  - **Scale** to big data (e.g. Netflix dataset)