Tencent AI Lab

RaFM

Rank-Aware Factorization Machines

Yin Zheng

On Behalf of

Xiaoshuang Chen, Yin Zheng, Jiaxing Wang, Wenye Ma, Junzhou Huang

Motivation

Factorization Machines

Factorized embeddings for each feature:

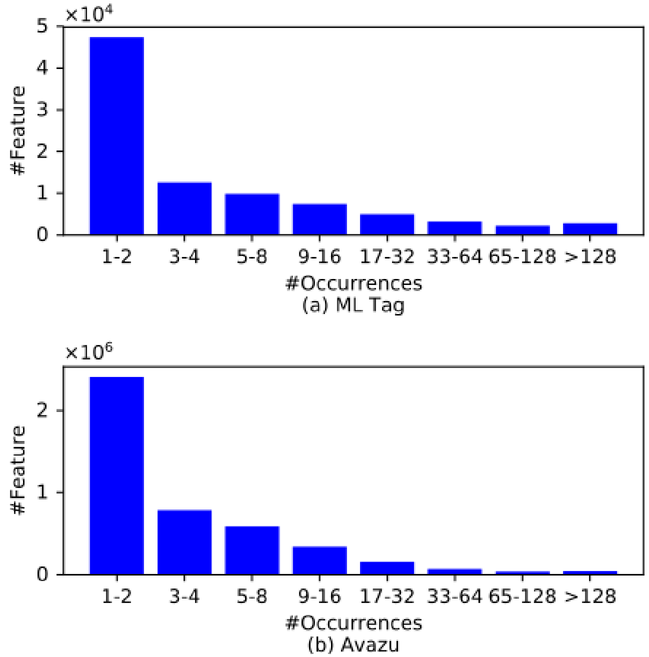
$$\mathrm{V}_i = \boldsymbol{v}_i$$

Modeling pairwise interactions:

$$\hat{y} = \sum_{i,j \in \mathrm{F},\, i<j} \boxed{\langle \mathrm{V}_i, \mathrm{V}_j \rangle} x_i x_j + \sum_{i \in \mathrm{F}} w_i x_i + bias$$

$$\langle \mathrm{V}_i, \mathrm{V}_j \rangle_{FM} = \boldsymbol{v}_i \cdot \boldsymbol{v}_j = \sum_{f=1}^{D} v_{i,f} v_{j,f}$$

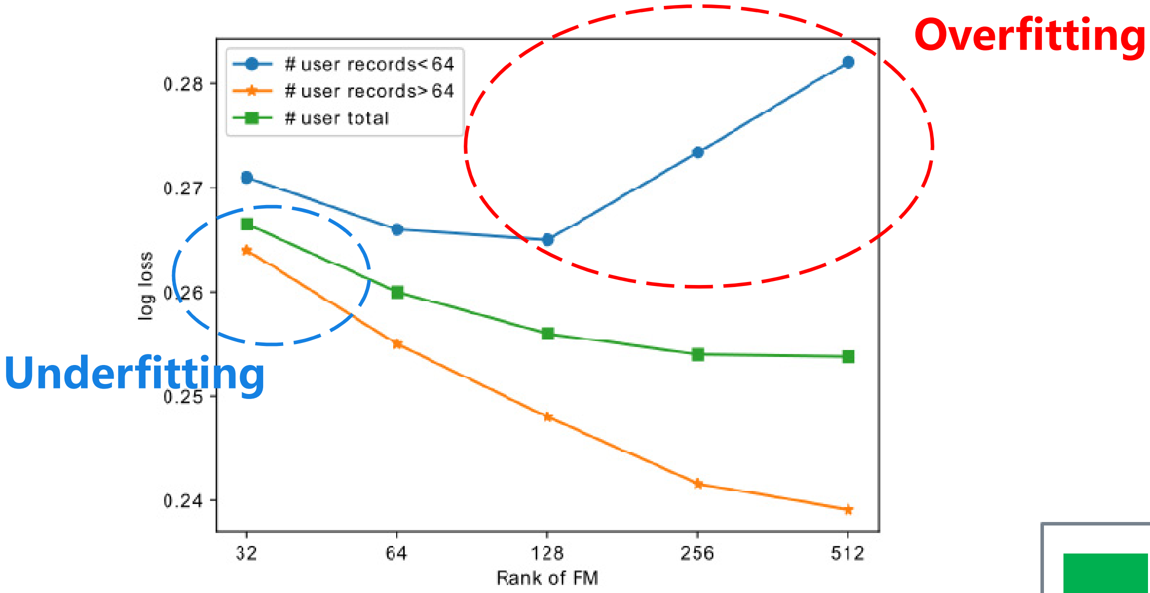
Different features have different frequencies of occurences



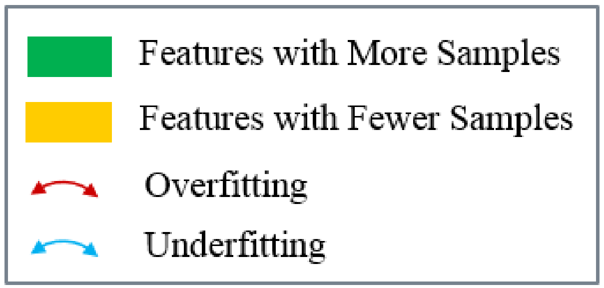
What is the best rank of the embeddings?

Motivation

Performance of FMs with fixed ranks

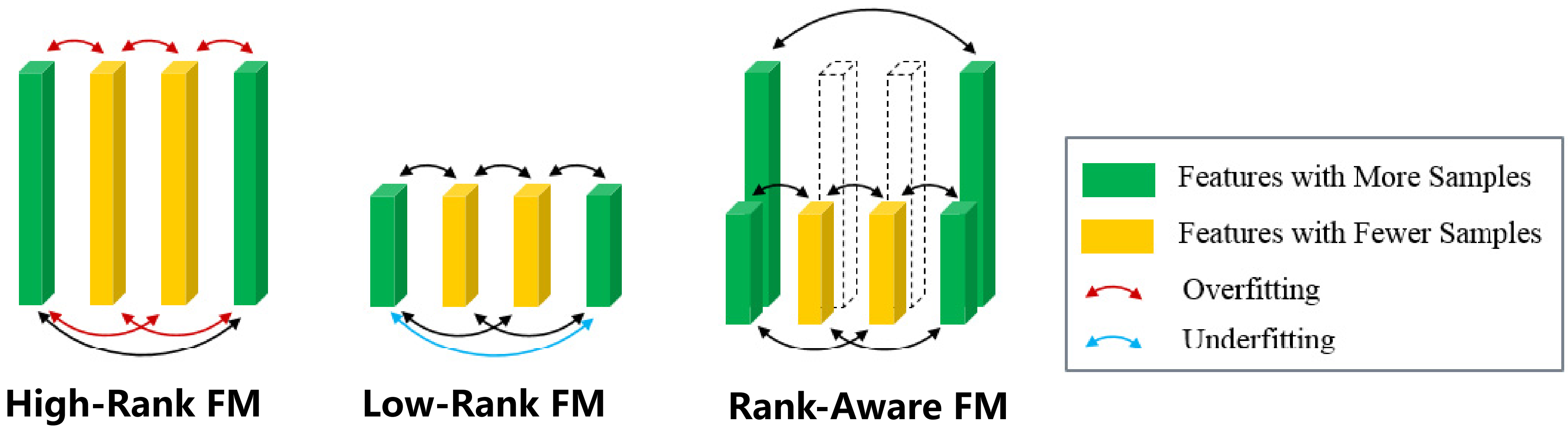


MovieLens Tag



Basic Model

Rank-Aware Factorization Machines



Basic Model

Tencent AI Lab

Rank-Aware Factorization Machines

$$\hat{y} = \sum_{i,j\in \mathrm{F},i<j} \boxed{\langle \mathrm{V}_i, \mathrm{V}_j \rangle} x_i x_j + \sum_{i\in \mathrm{F}} w_i x_i + bias$$

Multiple embeddings with different ranks:

$$\mathrm{V}_i = \left\{ \boldsymbol{v}_i^{(1)}, \boldsymbol{v}_i^{(2)}, \cdots, \boxed{\boldsymbol{v}_i^{(k_i)}} \right\}$$

The largest rank to avoid overfitting

(hyperparameters)

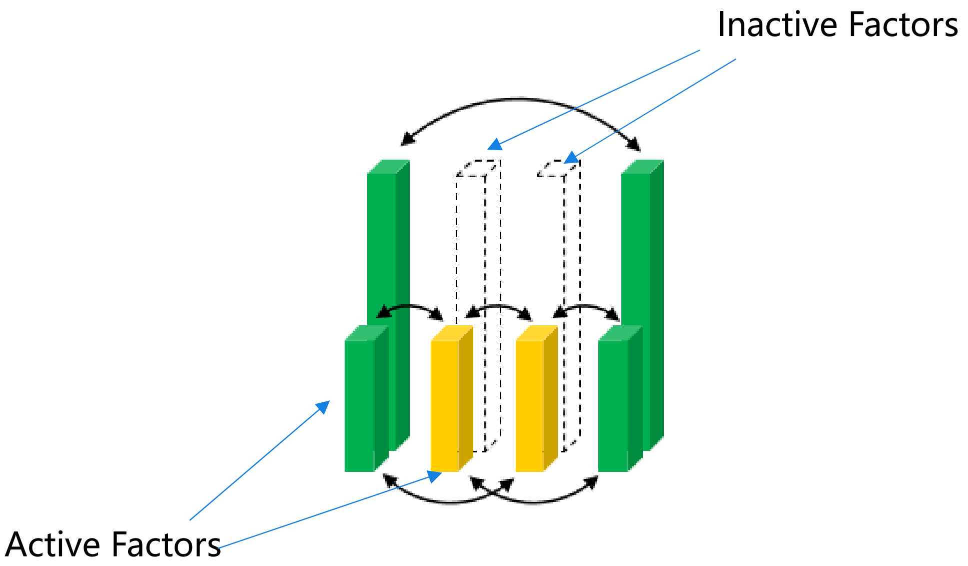
$$\langle \mathrm{V}_i, \mathrm{V}_j \rangle_{RaFM} = \boldsymbol{v}_i^{(k_{ij})} \cdot \boldsymbol{v}_j^{(k_{ij})} \qquad k_{ij} = \min(k_i, k_j)$$

Choose a proper rank for computation of pairwise interaction

- **What is the time and space complexity?**
- **How to efficiently train RaFM?**

Space Complexity

Active and Inactive Factors



Described by Feature Set

$$\mathrm{F}_k = \{ i \in \mathrm{F} : k_i \geq k \}$$

Inactive factors: $\boldsymbol{v}^{(p)}\big|_{\mathrm{F} - \mathrm{F}_p}$ **Need NOT be stored!**

Active factors: $\boldsymbol{v}^{(p)}\big|_{\mathrm{F}_p}$

Space Complexity: $O\left(\sum_{k=1}^{m} D_k \left| \mathrm{F}_k \right| \right)$

Time Complexity

$$F_k = \{i \in F : k_i \geq k\}$$

$$k_{ij}\big|_{[l,k]} = \max\left[l, \min\left(k, k_{ij}\right)\right]$$

Auxiliary Variables

$$A_{l,k} = \sum_{i,j \in F_k, i<j} \boldsymbol{v}_i^{(l)} \cdot \boldsymbol{v}_j^{(l)} x_i x_j = \frac{1}{2}\left(\left\|\sum_{i \in F_k} \boldsymbol{v}_i^{(l)} x_i\right\|_2^2 - \sum_{i \in F_k} \left\|\boldsymbol{v}_i^{(l)} x_i\right\|_2^2\right) \Rightarrow O\left(D_l \left|F_k\right|\right)$$

$$B_{l,k} = \sum_{i<j} \boldsymbol{v}_i^{\left(k_{ij}|_{[l,k]}\right)} \cdot \boldsymbol{v}_j^{\left(k_{ij}|_{[l,k]}\right)} x_i x_j$$

$$RaFM = B_{1,m}$$

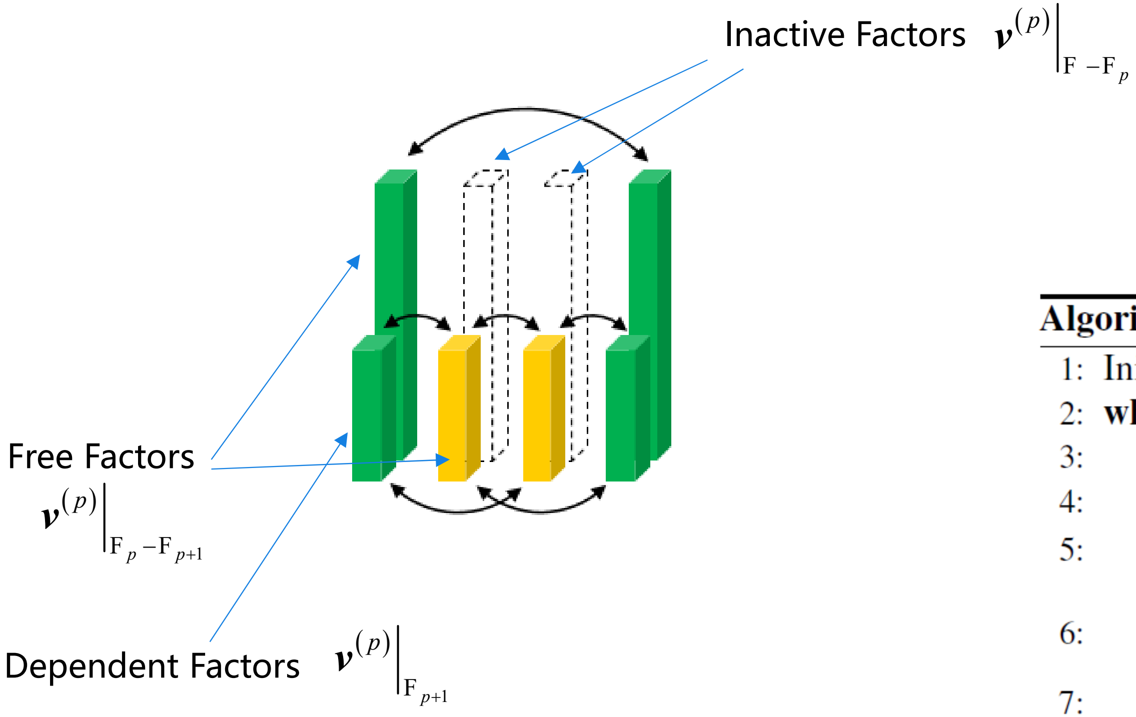
It is easy to prove that

$$B_{l,k+1} = B_{l,k} - A_{k,k+1} + A_{k+1,k+1} \Rightarrow O\left(\sum_{k=1}^{m} D_k \left|F_k\right|\right)$$

Learning Algorithm

$$F_k = \{ i \in F : k_i \geq k \}$$

Free and Dependent Factors



Bi-Level Optimization

$$\min \frac{1}{N} \sum_{\boldsymbol{x}} L\left(B_{1,m}, y\right)$$

$$\boldsymbol{v}^{(p)}\Big|_{F_{p+1}} = \arg\min \frac{1}{N} \sum_{\boldsymbol{x}} L\left(B_{1,p}, B_{1,p+1}\right), \forall 1 \leq p < m$$

Pushing dependent factors to approximate free factors

Algorithm 1 Training the RaFM

Initialize all the parameters
while not convergent **do**
  Sample a data point $(\boldsymbol{x}, y)$ randomly
  for $1 \leq p < m$ **do**
    $\boldsymbol{v}^{(p)}\big|_{\mathcal{F}_{p+1}} \leftarrow \boldsymbol{v}^{(p)}\big|_{\mathcal{F}_{p+1}} \boxed{- \rho_d \frac{\partial L(\mathcal{B}_{1,p}, \mathcal{B}_{1,p+1})}{\partial \boldsymbol{v}^{(p)}|_{\mathcal{F}_{p+1}}}}$
    $\boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p - \mathcal{F}_{p+1}} \leftarrow \boxed{\boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p - \mathcal{F}_{p+1}} - \rho_f \frac{\partial L(\mathcal{B}_{1,m}, y)}{\partial \boldsymbol{v}^{(p)}|_{\mathcal{F}_p - \mathcal{F}_{p+1}}}}$
  end for
  $\boldsymbol{v}^{(m)}\big|_{\mathcal{F}_m} \leftarrow \boldsymbol{v}^{(m)}\big|_{\mathcal{F}_m} - \rho_f \frac{\partial L(\mathcal{B}_{1,m}, y)}{\partial \boldsymbol{v}^{(m)}|_{\mathcal{F}_m}}$
end while

Proved by Thm. 6

Experiment

Table 3. Results on Regression Tasks

	ML 10M			ML 20M			AMovie		
	square loss	#param	train/test time	square loss	#param	train/test time	square loss	#param	train/test time
FM	0.8016 ±0.0010	2.66M	1×	0.8002 ±0.0008	5.45M	1×	1.0203 ±0.0046	3.25M	1×
DiFacto	0.7950 ±0.0011	1.79M	0.82×/ 0.95×	0.7948 ±0.0005	3.22M	0.70×/ 0.80×	1.0268 ±0.0051	1.76M	0.75×/ 0.75×
MRMA	0.7952 ±0.0006	4.11M	1.27×/ 1.43×	0.7855 ±0.0011	8.43M	1.19×/ 1.38×	1.0071 ±0.0039	5.02M	1.27×/ 1.27×
RaFM	**0.7870 ±0.0008**	1.57M	0.95×/ 1.12×	**0.7807 ±0.0009**	3.63M	0.74×/ 0.85×	**0.9986 ±0.0035**	1.76M	0.75×/ 0.75×

Table 4. Results on Classification Tasks

	Frappe				ML Tag			
	log loss	AUC	#param	train/test time	log loss	AUC	#param	train/test time
FM	0.1702 ±0.0023	0.9771 ±0.0008	1.38M	1×	0.2538 ±0.0009	0.9503 ±0.0006	46.40M	1×
DiFacto	0.1711 ±0.0023	0.9771 ±0.0004	0.61M	0.63×/ 0.85×	0.2529 ±0.0007	0.9450 ±0.004	16.97M	0.42×/ 0.83×
RaFM	**0.1447 ±0.0015**	**0.9811 ±0.0002**	0.71M	0.73×/ 0.85×	**0.2387 ±0.0005**	**0.9526 ±0.0006**	9.23M	0.24×/ 0.71×
	Avazu				**Criteo**			
	log loss	AUC	#param	train/test time	log loss	AUC	#param	train/test time
FM	0.3817 ±0.0001	0.7761 ±0.0003	18.81M	1×	0.4471 ±0.0002	0.8030 ±0.0002	35.87M	1×
DiFacto	0.3823 ±0.0003	0.7778 ±0.0003	10.83M	0.82×/ 1.79×	0.4470 ±0.0002	0.8030 ±0.0004	19.70M	0.63×/ 0.80×
RaFM	**0.3801 ±0.0002**	**0.7826 ±0.0003**	10.17M	0.85×/ 1.20×	**0.4451 ±0.0001**	**0.8060 ±0.0002**	20.88M	0.67×/ 0.84×

- **RaFM outperforms FM.**
- **RaFM is also more computational efficient than FM.**

Improvement: 0.5%~15%

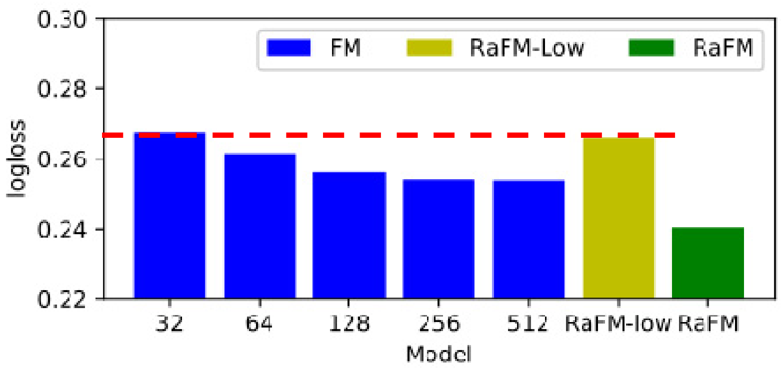
Model Size: 20%~66%

Training Time: 24%~95%

Experiment

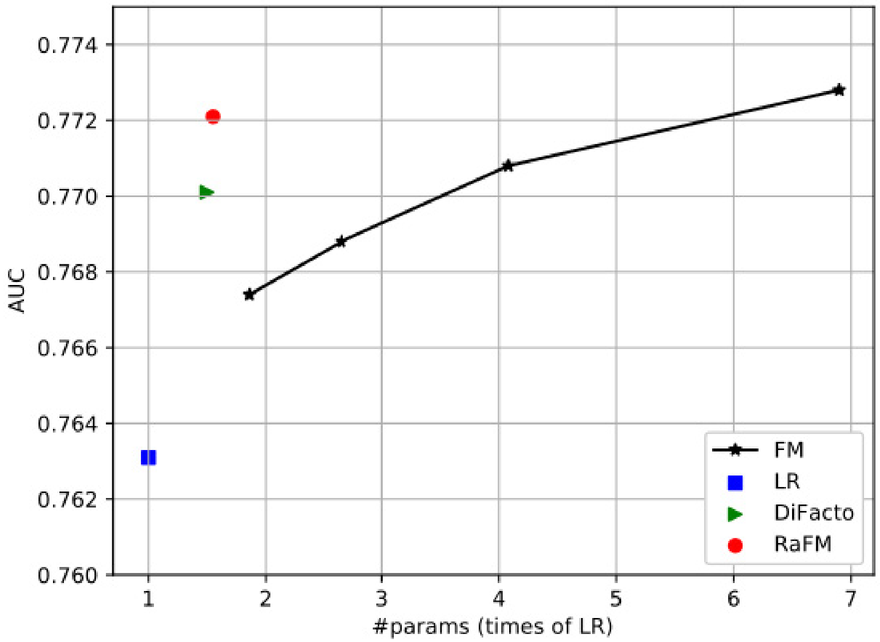
RaFM vs. FM

RaFM: 32 + 512



RaFM-low has similar performance as FM-32.

Results on Tencent CTR Dataset



Pacific Ballroom

Jun 13th 6:30PM~9:00PM

PosterID 220

Thanks!

Code	https://github.com/cxsmarkchan/RaFM
Xiaoshuang Chen	https://cxsmarkchan.github.io
Yin Zheng	https://sites.google.com/site/zhengyin1126