Discrete Factorization Machines for Fast Feature-based Recommendation

Han Liu[1], Xiangnan He[2], Fuli Feng[2], Liqiang Nie[1],
Rui Liu[3], Hanwang Zhang[4]

1.Shandong University
2.National University of Singapore
3.University of Electronic Science and Technology of China
4.Nanyang Technological University

Motivation

Accurate Recommender System

Quality of Service & Profit of the Service Provider

side information

content-based : *e.g.*, item descriptions

context-based : *e.g.*, when and where a purchase is made

session-based : e.g., recent browsing history of users

user

item

Factorization Machines (FM)

FM is a score prediction function for a (user, item) pair feature **x**.

$\mathbf{x} \in \mathbb{R}^n$ is a feature representation of the side-information, concatenated

$\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ models the interaction between the i-th and j-th feature dimensions.

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

Model bias parameter w_i

k–dim latent factor vector $\mathbf{v}_i$

FM models the interaction between **each pair** of nonzero features

Motivation

yelp

1,300,000 users	174,000 business	1,200,000 attributes

here n = 1,300,000+174,000+1,200,000 = 2,674,000

$\mathbf{V} \in \mathbb{R}^{k \times n}$

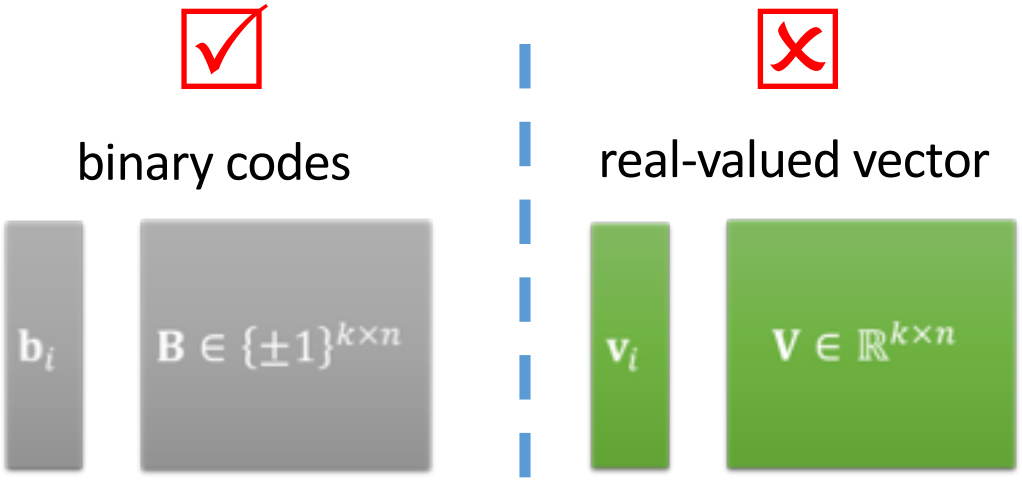
On-device storage?

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$$

Computation cost?

Existing FM framework is not suitable for fast recommendation, especially for mobile users.

Discrete Factorization Machines



Storing: Easily Store | Impossible

Computing: XOR Bit Operations | Float Multiplications

$$\mathrm{DFM}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{b}_i, \mathbf{b}_j \rangle x_i x_j.$$

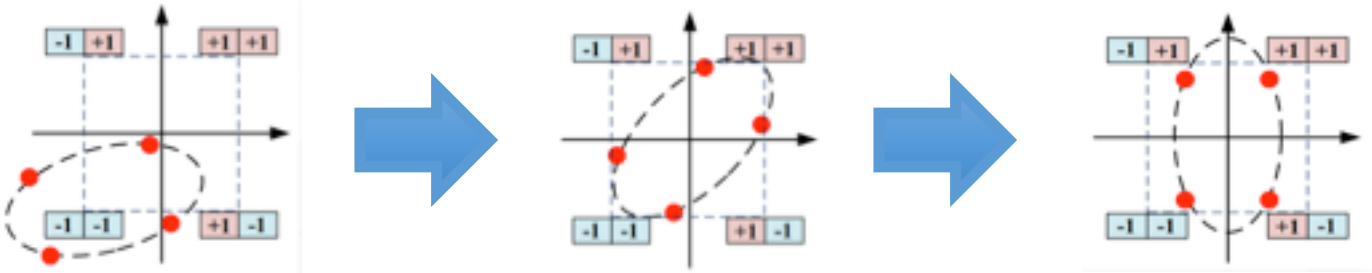
Solution with the Constraints

Observed score

Binary codes

$$\arg\min_{w_0,\mathbf{w},\mathbf{B}} \sum_{(\mathbf{x},y)\in\mathcal{V}} (y - w_0 - \sum_{i=1}^{n} w_i x_i - \sum_{i=1}^{n}\sum_{j=i+1}^{n} \langle \mathbf{b}_i, \mathbf{b}_j \rangle x_i x_j)^2$$

$$+ \alpha \sum_{i=1}^{n} w_i^2, \text{s.t. } \mathbf{B} \in \{\pm 1\}^{k\times n}, \underbrace{\mathbf{B1} = \mathbf{0}}_{\text{Balance}}, \underbrace{\mathbf{BB}^T = n\mathbf{I}}_{\text{De-correlation}}$$



Without any constraints | Balanced | De-correlated

Balance Constraint: each bit should split the dataset evenly

De-Correlation Constraint: each bit should be as independent as possible

However, the hard constraints of zero-mean and orthogonality may not be satisfied in Hamming space!

Our DFM Formulation

Objective Function:

$$\arg\min_{w_0,\mathbf{w},\mathbf{B}} \sum_{(\mathbf{x},y)\in\mathcal{V}} (y - w_0 - \sum_{i=1}^{n} w_i x_i - \sum_{i=1}^{n}\sum_{j=i+1}^{n} \langle \mathbf{b}_i, \mathbf{b}_j \rangle x_i x_j)^2 + \alpha \sum_{i=1}^{n} w_i^2 + \beta \|\mathbf{B} - \mathbf{D}\|^2$$

Score Prediction

Constraint Trade-off

Binary Constraint: $\mathbf{B} \in \{\pm 1\}^{k \times n}$

Delegate Code Quality Constraint: $\mathbf{D1} = \mathbf{0}, \ \mathbf{D}\mathbf{D}^T = n\mathbf{I}$

Balance Constraint

De-correlation Constraint

Our Solution: Alternating Optimization

Alternative Procedure

B-Subproblem

$$\arg\min_{\mathbf{B}} \sum_{(\mathbf{x},y)\in\mathcal{V}} (y - w_0 - \sum_{i=1}^{n} w_i x_i - \sum_{i=1}^{n}\sum_{j=i+1}^{n} \langle \mathbf{b}_i, \mathbf{b}_j \rangle x_i x_j)^2 - 2\beta tr(\mathbf{B}^T\mathbf{D}),\ \text{s.t. } \mathbf{B} \in \{\pm 1\}^{k\times n}$$

D-Subproblem

$$\arg\max_{\mathbf{D}} tr(\mathbf{B}^T\mathbf{D}), s.t.\ \mathbf{D1} = \mathbf{0}, \mathbf{D}\mathbf{D}^T = m\mathbf{I}.$$

w-Subproblem

$$\arg\min_{w_0,\mathbf{w}} \sum_{(\mathbf{x},y)\in\mathcal{V}} (\phi - w_0 - \sum_{i=1}^{n} w_i x_i)^2 + \alpha \sum_{i=1}^{n} w_i^2,\ \ \phi = y - \sum_{i=1}^{n}\sum_{j=i+1}^{n} \langle \mathbf{b}_i, \mathbf{b}_j \rangle x_i x_j$$

B-Subproblem for Binary Codes

Objective Function

$$\arg\min_{\mathbf{B}} \sum_{(\mathbf{x},y)\in\mathcal{V}} (y - w_0 - \sum_{i=1}^{n} w_i x_i - \sum_{i=1}^{n}\sum_{j=i+1}^{n} \langle \mathbf{b}_i, \mathbf{b}_j \rangle x_i x_j)^2 - 2\beta tr(\mathbf{B}^T\mathbf{D}), \text{ s.t. } \mathbf{B} \in \{\pm 1\}^{k\times n}$$

For each feature code $\mathbf{b}_{\downarrow i}$, optimize ***bit by bit***

for $r = 1$ **to** n **do** — **for** loop over ***n*** features

  repeat

    for $t = 1$ **to** k **do** — **for** loop over ***k*** bits

      $\hat{b}_{rt} = \sum_{\mathcal{V}_r} (x_r \psi - x_r^2 \hat{\mathbf{x}}^T \mathbf{Z}_{\bar{t}} \mathbf{b}_{r\bar{t}}) \hat{\mathbf{x}}^T \mathbf{z}_t + \beta d_{rt}$;

      $b_{rt} \leftarrow \text{sgn}\big(K(\hat{b}_{rt}, b_{rt})\big)$

    end

  until *converge*;

end

D-Subproblem for Code Delegate

Objective Function

$$\arg\max_{\mathbf{D}} tr(\mathbf{B}^T\mathbf{D}), s.t.\ \mathbf{D1} = \mathbf{0},\ \mathbf{DD}^T = n\mathbf{I}$$

Small SVD $k \times n$ $\quad ([\mathbf{P}\widehat{\mathbf{P}}], \mathbf{Q}) \leftarrow \mathrm{SVD}(\overline{\mathbf{B}})$

$k \times n$ row-centered code matrix

Orthogonalization $\quad \widehat{\mathbf{Q}} \leftarrow \mathrm{GramSchmidt}([\mathbf{Q1}])$

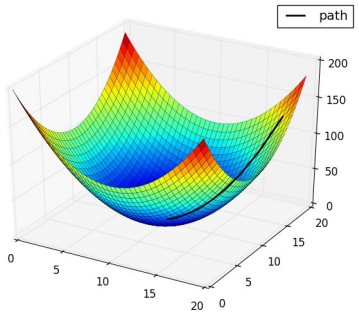
$$\mathbf{D} \leftarrow \sqrt{n} \begin{bmatrix} \mathbf{P} & \widehat{\mathbf{P}} \end{bmatrix} \begin{bmatrix} \mathbf{Q} & \widehat{\mathbf{Q}} \end{bmatrix}^T$$

w-Subproblem for Bias

Objective Function

$$\arg\min_{w_0,\mathbf{w}} \sum_{(\mathbf{x},y)\in\mathcal{V}} (\phi - w_0 - \sum_{i=1}^{n} w_i x_i)^2 + \alpha \sum_{i=1}^{n} w_i^2,\ \phi = y - \sum_{i=1}^{n}\sum_{j=i+1}^{n} \langle \mathbf{b}_i, \mathbf{b}_j \rangle x_i x_j$$

It is the standard multivariate linear regression problem, use Coordinate Descent algorithm



Experiment Settings

- Datasets:

Datasets	#users	#items	#ratings	Density
Yelp	13,679	12,922	640,143	0.36%
Amazon	35,151	33,195	1,732,060	0.15%

- Split: randomly split 50% training and 50% testing
  move items in the testing set that haven't occurred in the training set to the training set.
- Evaluation Protocol: rank the testing items of a user and evaluate the ranked list with NDCG@K

Compared to the state-of-the-art

- **libFM**: Factorization Machines with ***libFM*** [Rendle et al.,TIST'12]

  original implementation of FM

- **DCF**: ***D***iscrete ***C***ollaborative ***F***iltering [Zhang et al.,SIGIR'16]

  CF+binarization+direct optimization

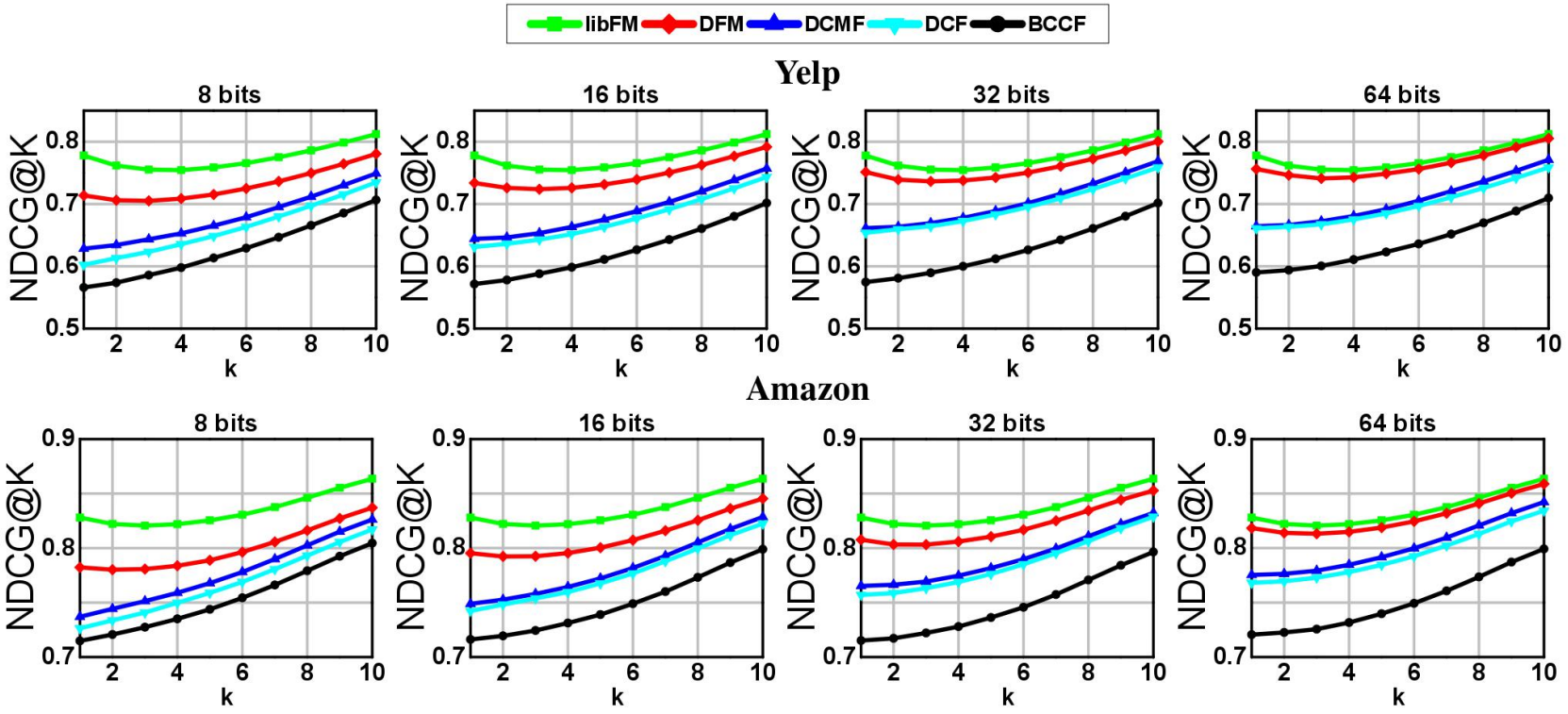
- **DCMF**: ***D***iscrete ***C***ontent-aware ***M***atrix ***F***actorization [Lian et al.,KDD'17]

  CF+binarization+direct optimization+constraint

- **BCCF**: ***B***inary ***C***ode learning for ***C***ollaborative ***F***iltering [Zhou&Zha,KDD'12]

  MF+binarization+two-stage optimization

Performance Comparison



In figure, we show the recommendation performance (NDCG@1 to NDCG@10) of DFM and the baseline methods on the two datasets. The code length varies from 8 to 64.

Efficiency Study

Efficiency comparison between DFM and libFM regarding Testing Time Cost (TTC) on the two datasets.

Yelp

Code Length	**8**	**16**	**32**	**64**
libFM (TTC)	27.18	56.77	114.10	217.64
DFM (TTC)	2.06	3.56	6.60	12.43
Acceleration Ratio	13.19	15.95	17.29	17.51

Amazon

Code Length	8	16	32	64
libFM (TTC)	177.03	357.46	716.83	1,414.67
DFM (TTC)	12.67	22.50	42.56	81.04
Acceleration Ratio	13.97	15.89	16.84	17.46

DFM is an operable solution for many large-scale Web service to reduce the computation cost of their recommender systems.

Conclusion & Future Work

- We propose DFM to enable fast feature-based recommendation.
- We develop an efficient algorithm to address the challenging optimization problem of DFM.
- We will extend binary technique to neural recommender models such as Neural FM.

Q&A

Thank you.

https://github.com/hanliu95/DFM