Matrix Factorization and Factorization Machines for Recommender Systems

Chih-Jen Lin
Department of Computer Science
National Taiwan University

Talk at 4th Workshop on Large-Scale Recommender Systems, ACM RecSys, 2016

Outline

Outline

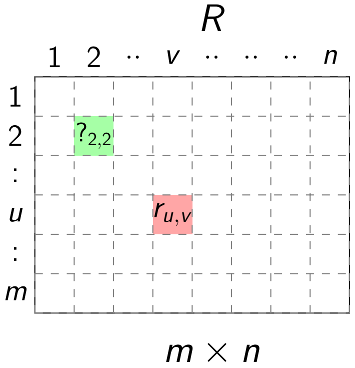
Matrix Factorization

- Matrix Factorization is an effective method for recommender systems (e.g., Netflix Prize and KDD Cup 2011)
- A group of users give ratings to some items

User	Item	Rating
1	5	100
1	13	30
...	...	...
u	v	r
...	...	...

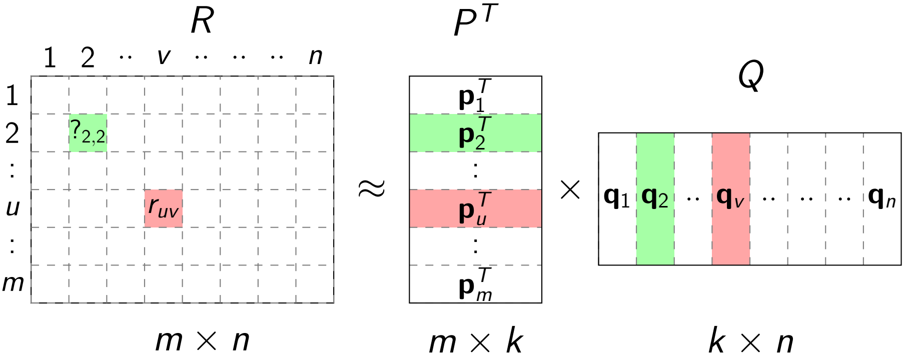
- The information can be represented by a rating matrix R

Matrix Factorization (Cont'd)



- m, n: numbers of users and items
- u, v: index for u_{th} user and v_{th} item
- $r_{u,v}$: u_{th} user gives a rating $r_{u,v}$ to v_{th} item

Matrix Factorization (Cont'd)



- k: number of latent dimensions
- $r_{u,v} = \mathbf{p}_u^T \mathbf{q}_v$
- $?_{2,2} = \mathbf{p}_2^T \mathbf{q}_2$

Matrix Factorization (Cont'd)

- A non-convex optimization problem:

$$\min_{P,Q} \sum_{(u,v)\in R} \left((r_{u,v} - \mathbf{p}_u^T \mathbf{q}_v)^2 + \lambda_P \|\mathbf{p}_u\|_F^2 + \lambda_Q \|\mathbf{q}_v\|_F^2 \right)$$

  λ_P and λ_Q are regularization parameters
- Many optimization methods have been successfully applied
- Overall MF is a mature technique

Outline

MF versus Classification/Regression

- MF solves

$$\min_{P,Q} \sum_{(u,v)\in R} \left(r_{u,v} - \mathbf{p}_u^T \mathbf{q}_v\right)^2$$

  Note that I omit the regularization term
- Ratings are the only given information
- This doesn't sound like a classification or regression problem
- But indeed we can make some interesting connections

Handling User/Item Features

- What if instead of user/item IDs we are given user and item features?
- Assume user u and item v have feature vectors

$$\mathbf{f}_u \in R^U \text{ and } \mathbf{g}_v \in R^V,$$

where

$$U \equiv \text{number of user features}$$
$$V \equiv \text{number of item features}$$

- How to use these features to build a model?

Handling User/Item Features (Cont'd)

- We can consider a regression problem where data instances are

value	features
$\vdots$	$\vdots$
r_{uv}	$\begin{bmatrix} \mathbf{f}_u^T & \mathbf{g}_v^T \end{bmatrix}$
$\vdots$	$\vdots$

  and solve

$$\min_{\mathbf{w}} \sum_{u,v \in R} \left(r_{u,v} - \mathbf{w}^T \begin{bmatrix} \mathbf{f}_u \\ \mathbf{g}_v \end{bmatrix} \right)^2$$

Feature Combinations

- However, this does not take the interaction between users and items into account
- Following the concept of degree-2 polynomial mappings in SVM, we can generate new features

$$(f_u)_t(g_v)_s, t = 1, \ldots, U, s = 1, \ldots V$$

and solve

$$\min_{w_{t,s}, \forall t,s} \sum_{u,v \in R} (r_{u,v} - \sum_{t=1}^{U} \sum_{s=1}^{V} w_{t,s}(f_u)_t(g_v)_s)^2$$

Feature Combinations (Cont'd)

- This is equivalent to

$$\min_{W} \sum_{u,v \in R} (r_{u,v} - \mathbf{f}_u^T W \mathbf{g}_v)^2,$$

  where

$$W \in R^{U \times V} \text{ is a matrix}$$

- If we have $\text{vec}(W)$ by concatenating W's columns, another form is

$$\min_{W} \sum_{u,v \in R} \left(r_{u,v} - \text{vec}(W)^T \begin{bmatrix} \vdots \\ (f_u)_t (g_v)_s \\ \vdots \end{bmatrix} \right)^2,$$

Feature Combinations (Cont'd)

- However, this setting fails for extremely sparse features
- Consider the most extreme situation. Assume we have

  user ID and item ID

  as features
- Then

$$U = m, J = n,$$
$$\mathbf{f}_i = [\underbrace{0, \ldots, 0}_{i-1}, 1, 0, \ldots, 0]^T$$

Feature Combinations (Cont'd)

- The optimal solution is

$$W_{u,v} = \begin{cases} r_{u,v}, & \text{if } u, v \in R \\ 0, & \text{if } u, v \notin R \end{cases}$$

- We can never predict

$$r_{u,v}, u, v \notin R$$

Factorization Machines

- The reason why we cannot predict unseen data is because in the optimization problem

$$\# \text{ variables} = mn \gg \# \text{ instances } = |R|$$

- Overfitting occurs
- Remedy: we can let

$$W \approx P^T Q,$$

where P and Q are low-rank matrices. This becomes matrix factorization

Factorization Machines (Cont'd)

- This can be generalized to sparse user and item features

$$\min_{P,Q} \sum_{(u,v)\in R} (r_{u,v} - \mathbf{f}_u^T P^T Q \mathbf{g}_v)^2$$

- That is, we think

$$P\mathbf{f}_u \text{ and } Q\mathbf{g}_v$$

are latent representations of user u and item v, respectively

- We can also consider the interaction between elements in $\mathbf{f}_u$ (or elements in $\mathbf{g}_v$)

Factorization Machines (Cont'd)

- The new formulation is

$$\min_{P,Q} \sum_{(u,v)\in R} \left(r_{u,v} - \begin{bmatrix} \mathbf{f}_u^T & \mathbf{g}_v^T \end{bmatrix} \begin{bmatrix} P^T \\ Q^T \end{bmatrix} \begin{bmatrix} P & Q \end{bmatrix} \begin{bmatrix} \mathbf{f}_u \\ \mathbf{g}_v \end{bmatrix} \right)^2$$

- This becomes factorization machines (Rendle, 2010)

Factorization Machines (Cont'd)

- Similar ideas have been used in other places such as Stern et al. (2009)
- We see that such ideas can be used for not only recommender systems.
- They may be useful for any classification problems with very sparse features

FM for Classification

- In a classification setting assume a data instance is $\mathbf{x} \in R^n$
- Linear model:

$$\mathbf{w}^T \mathbf{x}$$

- Degree-2 polynomial mapping:

$$\mathbf{x}^T W \mathbf{x}$$

FM for Classification (Cont'd)

- FM:

$$\mathbf{x}^T P^T P \mathbf{x}$$

or alternatively

$$\sum_{i,j} \mathbf{x}_i \mathbf{p}_i^T \mathbf{p}_j \mathbf{x}_j,$$

where

$$\mathbf{p}_i, \mathbf{p}_j \in R^k$$

- That is, in FM each feature is associated with a latent factor

Outline

Field-aware Factorization Machines

- We have seen that FM seems to be useful to handle highly sparse features such as user IDs
- What if we have more than two ID fields?
- For example, in CTR (click-through rate) prediction for computational advertising, we may have

clicked	features
⋮	⋮
Yes	user ID, Ad ID, site ID
⋮	⋮

Field-aware Factorization Machines (Cont'd)

- FM can be generalized to handle different interactions between fields

  Two latent matrices for user ID and Ad ID
  Two latent matrices for user ID and site ID
  ⋮

- We call this approach FFM (field-aware factorization machines)
- An early study on three fields is Rendle and Schmidt-Thieme (2010)

FFM for CTR Prediction

- It's used by Jahrer et al. (2012) to win the 2nd prize of KDD Cup 2012
- In 2014 my students used FFM to win two Kaggle CTR competitions
- After we used FFM to win the first competition, in the second competition all top teams use FFM
- Note that for CTR prediction, logistic rather than squared loss is used

Practical Use of FFM

- Recently we conducted a detailed study on FFM (Juan et al., 2016)
- Here I briefly discuss some results there

Numerical Features

- For categorical features like IDs, we have

  ID: field ID index: feature
- Each field has many 0/1 features
- But how about numerical features?
- Two possibilities
  - Dummy fields: The field has only one real-valued feature
  - Discretization: transform a numerical feature to a categorical one and then many binary features

Normalization

- After obtaining the feature vector, empirically we find that instance-wise normalization is useful
- Faster convergence and better test accuracy

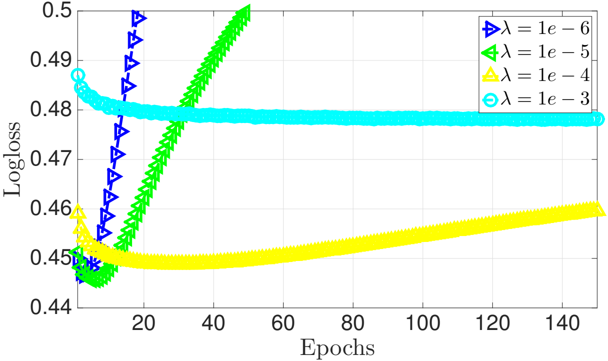
Impact of Parameters

We have the following parameters

- k: number of latent factors
- λ: regularization parameter
- parameters of the optimization methods (e.g., learning rate of stochastic gradient)

Their sensitivity to the performance varies

Example: Regularization Parameter λ



- Too large λ: model not good
- Too small λ: better model but easily overfitting
- Similar situations occur for SG learning rates
- Early stopping by a validation procedure is needed

Experiments: Two CTR Sets

method	test logloss	rank
Linear	0.46224	91
Poly2	0.44956	14
FM	0.44922	14
FM	0.44847	11
FFM	0.44603	3
Linear	0.38833	64
Poly2	0.38347	10
FM	0.38407	11
FM	0.38531	15
FFM	0.38223	6

For same method (e.g., FM), we try different parameters

Experiments: Two CTR Sets (Cont'd)

- For these two sets, FFM is the best
- For winning competitions, some additional tricks are used

Experiments: Other Sets

- Can FFM work well for other sets? Can we identify when it's useful
- We try the following data

Data Set	# instances	# features	# fields
KDD2010-bridge	20,012,499	651,166	9
KDD2012	20,950,284	19,147,857	11
phishing	11,055	100	30
adult	48,842	308	14
cod-rna (dummy fields)	331,152	8	8
cod-rna (discretization)	331,152	2,296	8
ijcnn (dummy fields)	141,691	22	22
ijcnn (discretization)	141,691	69,867	22

Experiments: Other Sets (Cont'd)

Data Set	LM	Poly2	FM	FFM
KDD2010-bridge	0.30910	0.27448	0.28437	<u>0.26899</u>
KDD2012	0.49375	0.49640	0.49292	<u>0.48700</u>
phishing	0.11493	0.09659	0.09461	<u>0.09374</u>
adult	0.30897	<u>0.30757</u>	0.30959	0.30760
cod-rna (dummy fields)	0.13829	0.12874	<u>0.12580</u>	0.12914
cod-rna (discretization)	0.16455	0.17576	0.16570	<u>0.14993</u>
ijcnn (dummy fields)	0.20627	0.09209	0.07696	<u>0.07668</u>
ijcnn (discretization)	0.21686	0.22546	0.22259	<u>0.18635</u>

Best results are underlined

Experiments: Other Sets (Cont'd)

- For data with categorical data, FFM works well
- For some data (e.g., `adult`), feature interactions are not useful
- It's not easy for FFM to handle numerical features

Outline

Solving the Optimization Problem

- MF, FM, and FFM all involve optimization problems
- Optimization techniques for them are related but different due to different problem structures
- With time constraint I will only briefly discuss some optimization techniques for matrix factorization

Matrix Factorization

- Recall we have a non-convex optimization problem:

$$\min_{P,Q} \sum_{(u,v)\in R} \left((r_{u,v} - \mathbf{p}_u^T \mathbf{q}_v)^2 + \lambda_P \|\mathbf{p}_u\|_F^2 + \lambda_Q \|\mathbf{q}_v\|_F^2 \right)$$

- Existing optimization techniques include
  - ALS: Alternating Least Squares (ALS)
  - CD : Coordinate Descent
  - SG : Stochastic Gradient

Complexity in Training MF

To update P, Q once

- ALS: $O(|R|k^2 + (m+n)k^3)$
- CD: $O(|R|k)$

To go through $|R|$ elements once

- SG: $O(|R|k)$

I don't discuss details, but this indicates that CD and SG are generally more efficient

Stochastic Gradient for Matrix Factorization

- SG update rule:

$$\mathbf{p}_u \leftarrow \mathbf{p}_u + \gamma \left(e_{u,v} \mathbf{q}_v - \lambda_P \mathbf{p}_u \right),$$
$$\mathbf{q}_v \leftarrow \mathbf{q}_v + \gamma \left(e_{u,v} \mathbf{p}_u - \lambda_Q \mathbf{q}_v \right)$$

  where

$$e_{u,v} \equiv r_{u,v} - \mathbf{p}_u^T \mathbf{q}_v$$

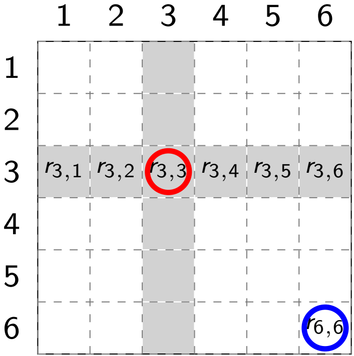
- Two issues:
  - SG is sensitive to learning rate
  - SG is inherently sequential

SG's Learning Rate

- We can apply advanced settings such as ADAGRAD (Duchi et al., 2011)
- Each element of latent vectors $\mathbf{p}_u$, $\mathbf{q}_v$ has its own learning rate
- Maintaining so many learning rates can be quite expensive
- How about a modification to let the whole $\mathbf{p}_u$ (or the whole $\mathbf{q}_v$) associates with a rate? (Chin et al., 2015b)
- This is an example that we take MF's property into account

SG for Parallel MF

After $r_{3,3}$ is selected, ratings in gray blocks cannot be updated



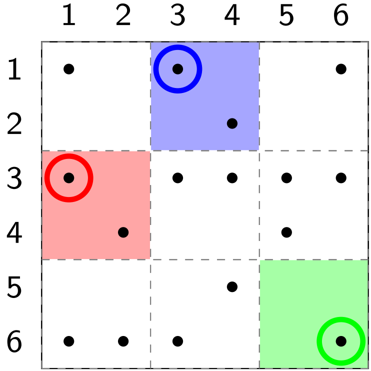
But $r_{6,6}$ can be used

- $r_{3,1} = \mathbf{p_3}^T \mathbf{q_1}$
- $r_{3,2} = \mathbf{p_3}^T \mathbf{q_2}$
- ..
- $r_{3,6} = \mathbf{p_3}^T \mathbf{q_6}$

- $r_{3,3} = \mathbf{p_3}^T \mathbf{q_3}$
  $r_{6,6} = \mathbf{p_6}^T \mathbf{q_6}$

SG for Parallel MF (Cont'd)

We can split the matrix to blocks and update those which don't share **p** or **q**



This concept is simple, but there are many issues to have a right implementation under the given architecture

SG for Parallel MF (Cont'd)

- Past developments of SG for parallel MF include Gemulla et al. (2011); Chin et al. (2015a); Yun et al. (2014)
- However, the idea of block splits applies to MF only
- We haven't seen an easy way to extend it to FM or FFM
- This is another example where we take problem structure into account

Outline

Discussion and Conclusions

- In this talk we briefly discuss three models for recommender systems

MF, FM, and FFM

- They are related, but are useful in different situations
- Different algorithms may be needed due to different properties of the optimization problems

Acknowledgments

Past and current students who have contributed to this work:

- Wei-Sheng Chin
- Yu-Chin Juan
- Meng-Yuan Yang
- Bo-Wen Yuan
- Yong Zhuang

We thank the support from Ministry of Science and Technology in Taiwan