Attribute-aware Collaborative Filtering: Survey and Classification

Recommendation system

Goal: discover user preferences over unrated items based on current user preferences

Current user preferences can be interpreted in three ways:

- Pointwise: preserve exact numerical rating that a user gives to an item
- Pairwise: preserve any pairs of item preferences of a particular user
- Listwise: preserve the relative order of all item ratings of a user as a list

Pointwise approach seeks to preserve item rating while pairwise and listwise seek to preserve item ranking, which does not require the recommendation system to produce ratings for each item

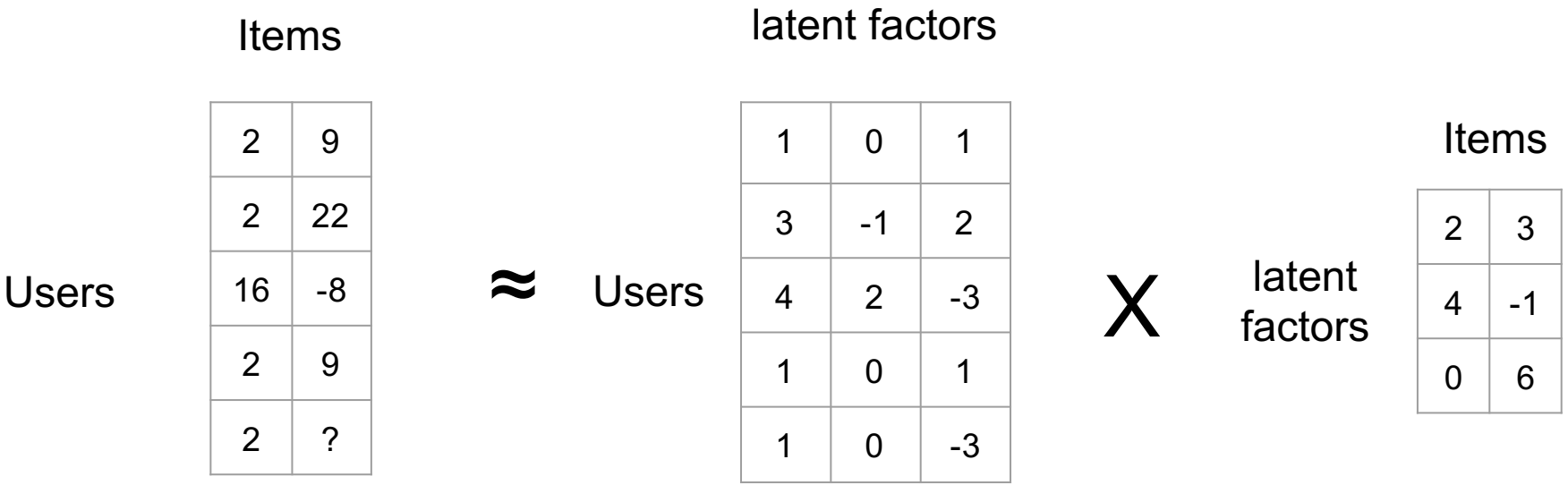
Rating type

- Explicit feedback: real value ratings e.g. [0, 5], [0, 100]
- Implicit feedback: binary ratings e.g. {0, 1}, {-1, 1}
  - e.g. whether a user clicks a webpage, where 0 indicates not clicked and 1 indicates clicked
  - Sigmoid function can be used to constrain real value predictions into [0, 1]

$$\mathrm{sig}(x) = \frac{1}{1+\exp(-x)}$$

  - Negative sampling can be applied when there is only positive examples in training data

Matrix Factorization - Toy example



1. Train user and item latent factors with observed ratings
2. Predict unobserved ratings with trained user and item factors

e.g. ? = <1, 0, -3> * <3, -1, 6> = -15

Matrix factorization - Formal definition

The goal of MF can be seen as finding user and item latent factors which minimize the square error between predicted ratings and true ratings:

$$\boldsymbol{W}^*,\ \boldsymbol{H}^* = \underset{\boldsymbol{W},\boldsymbol{H}}{\operatorname{argmin}} \sum_{(u,i)\in\delta(\boldsymbol{R})} \frac{1}{2}\left(r_{ui} - \boldsymbol{w}_u^\top \boldsymbol{h}_i\right)^2 + \frac{\lambda_W}{2}\sum_{u=1}^{N_u} \|\boldsymbol{w}_u\|_2^2 + \frac{\lambda_H}{2}\sum_{i=1}^{N_i} \|\boldsymbol{h}_i\|_2^2$$

Regularizations are added to prevent overfit of latent factors

Relation of MF to CF

- MF gives each user and item a latent factor which represents its underlying pattern
- Users with similar latent factors would have similar ratings on items
- Users with similar ratings on a same set of items are trained to have similar latent factors
- MF assumes rating matrix to be at most rank K
  - Low rank implies some rows or columns in rating matrix are linear dependent to others
  - Ratings of a user might be obtained by linear combinations of ratings from other users

Biased matrix factorization

The drawback of MF is it only considers the interaction between user and item, which neglects the bias of individual user or item. Bias MF improves it by considering them:

$$\hat{r}_{ui} = \mu + c_u + d_i + \boldsymbol{w}_u^\top \boldsymbol{h}_i$$

The formal definition then becomes:

$$\boldsymbol{W}^*, \boldsymbol{H}^*, \boldsymbol{c}^*, \boldsymbol{d}^*, \mu^* = \underset{\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{c}, \boldsymbol{d}, \mu}{\operatorname{argmin}} \sum_{(u,i) \in \delta(\boldsymbol{R})} (r_{ui} - \mu - c_u - d_i - \boldsymbol{w}_u^\top \boldsymbol{h}_i)^2 + \lambda \left(\|\boldsymbol{W}\|_F^2 + \|\boldsymbol{H}\|_F^2 + \|\boldsymbol{c}\|_2^2 + \|\boldsymbol{d}\|_2^2 \right)$$

Probabilistic Matrix Factorization (1/2)

- An equivalent probabilistic form of MF
- Assume each user and item follows a zero-mean spherical multivariate Gaussian distribution

$$p\left(\boldsymbol{W} \mid \sigma_W^2\right) = \prod_{u=1}^{N_u} \mathcal{N}\left(\boldsymbol{w}_u \mid \mathbf{0}, \sigma_W^2 \boldsymbol{I}\right), \qquad p\left(\boldsymbol{H} \mid \sigma_H^2\right) = \prod_{i=1}^{N_i} \mathcal{N}\left(\boldsymbol{h}_i \mid \mathbf{0}, \sigma_H^2 \boldsymbol{I}\right)$$

Conditional probability over observed ratings:

$$p\left(\boldsymbol{R} \mid \boldsymbol{W}, \boldsymbol{H}, \sigma^2\right) = \prod_{(i,j) \in \delta(\boldsymbol{R})} \mathcal{N}\left(r_{ui} \mid \boldsymbol{w}_u^\top \boldsymbol{h}_i, \sigma_R^2\right)$$

Probabilistic Matrix Factorization (2/2)

Maximum a posteriori (MAP):

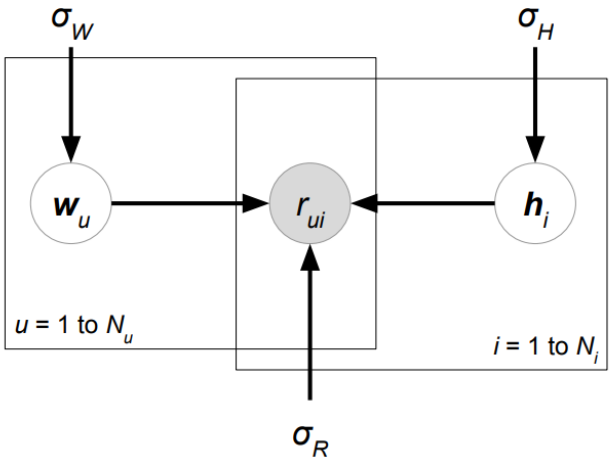
$$\begin{aligned}\log p\left(\boldsymbol{W}, \boldsymbol{H} \mid \boldsymbol{R}, \sigma_R^2, \sigma_W^2, \sigma_H^2\right) &= \log p\left(\boldsymbol{R} \mid \boldsymbol{W}, \boldsymbol{H}, \sigma_R^2\right) + \log p\left(\boldsymbol{W} \mid \sigma_W^2\right) + \log p\left(\boldsymbol{H} \mid \sigma_H^2\right) + C \\ &= -\frac{1}{2\sigma_R^2} \sum_{(u,i)\in\delta(\boldsymbol{R})} \left(r_{ui} - \boldsymbol{w}_u^\top \boldsymbol{h}_i\right)^2 - \frac{1}{2\sigma_W^2} \sum_{u=1}^{N_u} \boldsymbol{w}_u^\top \boldsymbol{w}_u - \frac{1}{2\sigma_H^2} \sum_{i=1}^{N_i} \boldsymbol{h}_i^\top \boldsymbol{h}_i \\ &\quad - \frac{1}{2}\left(|\delta(\boldsymbol{R})| \log \sigma_R^2 + N_u K \log \sigma_W^2 + N_i K \log \sigma_H^2\right) + C\end{aligned}$$

With Gaussian noise observed, it is equivalent to minimize the following objective function:

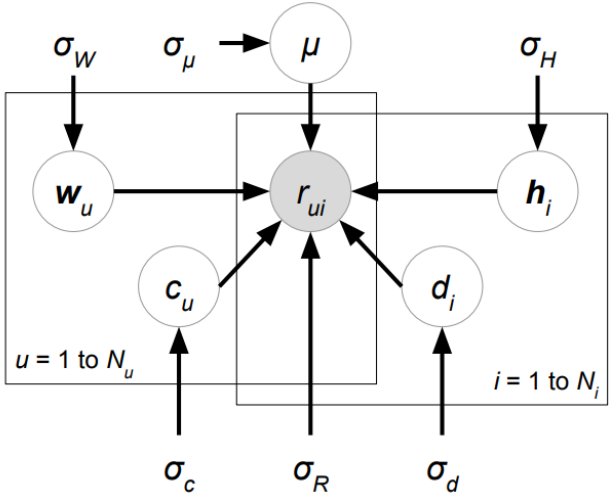
$$\sum_{(u,i)\in\delta(\boldsymbol{R})} \frac{1}{2}(r_{ui} - \boldsymbol{w}_u^\top \boldsymbol{h}_i)^2 + \frac{\lambda_W}{2} \sum_{u=1}^{N_u} \|\boldsymbol{w}_u\|_2^2 + \frac{\lambda_H}{2} \sum_{i=1}^{N_i} \|\boldsymbol{h}_i\|_2^2$$

which is the same as MF

Probabilistic MF - Graphical model



(a) PMF



(b) Biased PMF

Recommendation goal - Rating prediction (1/3)

- Pointwise learning
- The objective function of rating prediction recommendation system is often related to Root Mean Square Error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{|\delta(\boldsymbol{R})|} \sum_{(u,i)|r_{ui} \in \delta(\boldsymbol{R})} (\hat{r}_{ui} - r_{ui})^2}$$

Recommendation goal - Rating prediction (2/3)

Minimize MSE (root is ignored for convenience) or maximize normal likelihood

Explicit feedback:

$$\underset{\hat{r}}{\operatorname{argmin}} \sum_{(u,i)|r_{ui}\in\delta(\boldsymbol{R})} (\hat{r}_{ui} - r_{ui})^2$$

$$\underset{\hat{r}}{\operatorname{argmax}} \prod_{(u,i)|r_{ui}\in\delta(\boldsymbol{R})} \mathcal{N}\left(r_{ui} \mid \mu = \hat{r}_{ui}, \sigma^2\right)$$

Implicit feedback:

$$\underset{\hat{r}}{\operatorname{argmin}} \sum_{(u,i)|r_{ui}\in\delta(\boldsymbol{R})} (\operatorname{sig}(\hat{r}_{ui}) - r_{ui})^2$$

$$\underset{\hat{r}}{\operatorname{argmax}} \prod_{(u,i)|r_{ui}\in\delta(\boldsymbol{R})} \mathcal{N}\left(r_{ui} \mid \mu = \operatorname{sig}(\hat{r}_{ui}), \sigma^2\right)$$

Recommendation goal - Rating prediction (3/3)

If ratings are {0, 1}, the problem can be formulated as a binary classification problem, where logistic regression (Bernoulli likelihood) can be applied:

$$\underset{\hat{r}}{\operatorname{argmax}} \prod_{(u,i)|1=r_{ui}\in\delta(\boldsymbol{R})} \Pr\left(\hat{r}_{ui}=1\right) \prod_{(u,i)|0=r_{ui}\in\delta(\boldsymbol{R})} \Pr\left(\hat{r}_{ui}=0\right)$$

$$= \underbrace{\prod_{(u,i)|1=r_{ui}\in\delta(\boldsymbol{R})} \operatorname{sig}\left(\hat{r}_{ui}\right)}_{\text{Positive set}} \underbrace{\prod_{(u,i)|0=r_{ui}\in\delta(\boldsymbol{R})} \left(1-\operatorname{sig}\left(\hat{r}_{ui}\right)\right)}_{\text{Negative set}}.$$

Recommendation goal - Item ranking (1/2)

- Pairwise, listwise learning
- The objective function is to preserve user preferences over items

$$\underset{\hat{r}}{\operatorname{argmax}} \prod_{(u,i,j)|^{\{r_{ui},r_{uj}\}\subseteq\delta(\boldsymbol{R}),}_{r_{ui}>r_{uj}}} \Pr\left(\hat{r}_{ui} > \hat{r}_{uj}\right)$$

$$= \prod_{(u,i,j)|^{\{r_{ui},r_{uj}\}\subseteq\delta(\boldsymbol{R}),}_{r_{ui}>r_{uj}}} \operatorname{sig}\left(\hat{r}_{ui} - \hat{r}_{uj}\right)$$

Recommendation goal - Item ranking (2/2)

Most item ranking recommendation systems have objective function related to:

- AUC

$$\text{AUC} = \frac{1}{T} \sum_{(u,i,j)|^{\{r_{ui}, r_{uj}\} \subseteq \delta(\boldsymbol{R}),}_{r_{ui} > r_{uj}}} \mathbb{I}\left(\hat{r}_{ui} > \hat{r}_{uj}\right)$$

- Hinge loss

$$\underset{\hat{r}}{\text{argmin}} \sum_{(u,i,j)|^{\{r_{ui}, r_{uj}\} \subseteq \delta(\boldsymbol{R}),}_{r_{ui} > r_{uj}}} \max\left\{0, \hat{r}_{uj} - \hat{r}_{ui}\right\}$$

Item ranking - Bayesian Personalized Ranking

Bayesian Personalized Ranking (BPR)

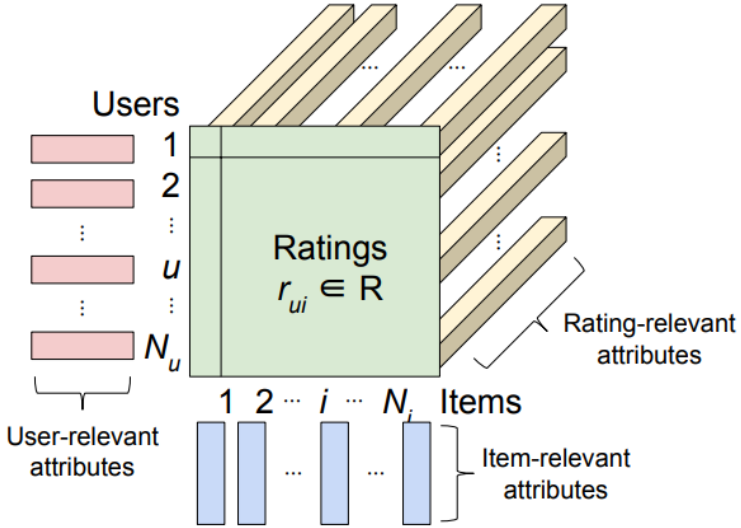
- Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI '09). AUAI Press, Arlington, Virginia, United States, 452-461.
- The objective function of BPR is to maximize AUC

$$
\begin{aligned}
\underset{\hat{r}}{\operatorname{argmax}} \ \mathrm{AUC} &= \sum_{(u,i,j)|^{\{r_{ui},r_{uj}\}\subseteq\delta(\boldsymbol{R}),}_{r_{ui}>r_{uj}}} \mathbb{I}\left(\hat{r}_{ui}-\hat{r}_{uj}>0\right) \\
&\approx \sum_{(u,i,j)|^{\{r_{ui},r_{uj}\}\subseteq\delta(\boldsymbol{R}),}_{r_{ui}-r_{uj}>0}} \operatorname{sig}\left(\hat{r}_{ui}-\hat{r}_{uj}\right) \\
&= \sum_{(u,i,j)|^{\{r_{ui},r_{uj}\}\subseteq\delta(\boldsymbol{R}),}_{r_{ui}-r_{uj}>0}} \log \operatorname{sig}\left(\hat{r}_{ui}-\hat{r}_{uj}\right) \\
&= \log \prod_{(u,i,j)|^{\{r_{ui},r_{uj}\}\subseteq\delta(\boldsymbol{R}),}_{r_{ui}-r_{uj}>0}} \operatorname{sig}\left(\hat{r}_{ui}-\hat{r}_{uj}\right)
\end{aligned}
$$

What are attributes?

Attribute source:

- User: age, job, gender etc.
- Item: location, price etc.
- Rating: upvotes, downvotes etc.



We refer to user/item attributes as side information and rating attributes as context

Attribute type

- Numerical
  - Real value, integer
  - e.g. Age, # of likes
- Categorical
  - Hard to be digitalized
  - e.g. Types of food, words

Attribute type - Numerical

Three common uses of numerical attributes

- Attributes fit latent factors

$$\underset{\theta, W}{\operatorname{argmin}} \| f_\theta(X) - W \| \text{ or } W = f_\theta(X) \text{ for user-relevant attributes}$$

- Latent factors fit attributes

$$\underset{\theta, W}{\operatorname{argmin}} \| f_\theta(W) - X \| \text{ or } X = f_\theta(W) \text{ for user-relevant attributes}$$

- Independent of latent factors

$$\underset{\theta, W, H}{\operatorname{argmin}} \left\| f_\theta(X) + W^\top H - R \right\|$$

Attribute type - Categorical

Categorical attributes are inappropriate to be directly transformed into numerical attributes because there is no ordering

=> Each dimension of attributes can be transformed into an one-hot encoding vector first and then be treated the same as numerical attributes

Attribute transformation (1/2)

Despite the sources of attributes are different (side information and context), they are actually interchangeable

- Side information can be convert to context by simply concatenate the original context with the user and item side information

$$\boldsymbol{z}'_{\pi(u,i)} = \begin{bmatrix} \boldsymbol{z}_{\pi(u,i)} \\ \boldsymbol{x}_u \\ \boldsymbol{y}_i \end{bmatrix} \in \mathbb{R}^{K_Z+K_X+K_Y}$$

Attribute transformation (2/2)

- Context can be converted to side information by appending all context of users/items to its original side information
  - If a particular feature is absent, the value should be all zeros. This is to ensure the dimension of user/item attributes to remain the same, as this is a requirement of most recommendation systems

$$\boldsymbol{x}'_u = \left[\boldsymbol{x}_u^\top \quad \boldsymbol{z}_{\pi(u,1)}^\top \quad \boldsymbol{z}_{\pi(u,2)}^\top \quad \cdots \quad \boldsymbol{z}_{\pi(u,i)}^\top \quad \cdots \quad \boldsymbol{z}_{\pi(u,N_i)}^\top\right]^\top \in \mathbb{R}^{K_X + K_Z N_i}$$

Why using attributes?

- More contraints can be added
  - Users with similar attributes should have similar latent factors
- Similarity between users or items can be pre-computed and further utilized
- Cold-start problem
  - Sometimes users or items have few observed ratings
  - Additional attributes are used as auxiliary information

How to use attributes?

- Discriminative matrix factorization (DMF)
- Generative matrix factorization
- Generalized factorization
- Heterogeneous graph

DMF - framework

Main idea: attributes model latent factors

$$\underset{\boldsymbol{W},\boldsymbol{H}}{\operatorname{argmax}} \underbrace{p\left(\boldsymbol{W},\boldsymbol{H} \mid \boldsymbol{R},\boldsymbol{X}\right)}_{\text{Posterior}} = \underset{\boldsymbol{W},\boldsymbol{H}}{\operatorname{argmax}} \frac{p\left(\boldsymbol{R} \mid \boldsymbol{W},\boldsymbol{H},\boldsymbol{X}\right) p\left(\boldsymbol{W},\boldsymbol{H} \mid \boldsymbol{X}\right)}{p\left(\boldsymbol{R} \mid \boldsymbol{X}\right)}$$

$$= \underset{\boldsymbol{W},\boldsymbol{H}}{\operatorname{argmax}}\, p\left(\boldsymbol{R} \mid \boldsymbol{W},\boldsymbol{H},\boldsymbol{X}\right) p\left(\boldsymbol{W},\boldsymbol{H} \mid \boldsymbol{X}\right)$$

$$= \underset{\boldsymbol{W},\boldsymbol{H}}{\operatorname{argmax}} \underbrace{p\left(\boldsymbol{R} \mid \boldsymbol{W},\boldsymbol{H},\boldsymbol{X}\right)}_{\text{Likelihood}} \underbrace{p\left(\boldsymbol{W} \mid \boldsymbol{X}\right) p\left(\boldsymbol{H} \mid \boldsymbol{X}\right)}_{\text{Prior}}$$

DMF - Linear model

Weight vectors are applied on attributes (like linear regression)

$$\underset{\boldsymbol{W},\boldsymbol{H},\theta}{\operatorname{argmax}} \prod_{(u,i)|r_{ui}\in\delta(\boldsymbol{R})} \mathcal{N}\left(r_{ui} \mid \mu_R = \boldsymbol{w}_u^\top \boldsymbol{h}_i + \boldsymbol{a}_u^\top \boldsymbol{x}_u + \boldsymbol{b}_i^\top \boldsymbol{y}_i + \boldsymbol{c}^\top \boldsymbol{z}_{\pi(u,i)}, \sigma_R^2\right) p(\boldsymbol{W} \mid \boldsymbol{X}) p(\boldsymbol{H} \mid \boldsymbol{Y})$$

- Pros: flexible; can be applied when some resources of attributes are absent, e.g. only user attributes are given
- Cons: user attributes and item attributes are considered separately

Example: Bayesian Matrix Factorization with Side Information (BMFSI)

$$\underset{\boldsymbol{W},\boldsymbol{H},\theta}{\operatorname{argmax}} \underbrace{p(\boldsymbol{R} \mid \boldsymbol{W}, \boldsymbol{H}, \theta)}_{\text{Likelihood}} \underbrace{p(\boldsymbol{W}) p(\boldsymbol{H})}_{\text{Priors}}$$

$$= \underset{\boldsymbol{W},\boldsymbol{H},\theta}{\operatorname{argmax}} \underbrace{\prod_{(u,i)|r_{ui}\in\delta(\boldsymbol{R})} \mathcal{N}\left(r_{ui} \mid \boldsymbol{w}_u^\top \boldsymbol{h}_i + \boldsymbol{a}_u^\top \boldsymbol{x}_u + \boldsymbol{b}_i^\top \boldsymbol{y}_i, \sigma_R^2\right)}_{\text{Matrix factorization using attributes}} \underbrace{\prod_u \mathcal{N}\left(\boldsymbol{w}_u \mid \boldsymbol{\mu}_u, \boldsymbol{\Sigma}_u\right) \prod_i \mathcal{N}\left(\boldsymbol{h}_i \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)}_{\text{Regularization}}$$

Ian Porteous, Arthur Asuncion, and Max Welling. 2010. Bayesian matrix factorization with side information and dirichlet process mixtures. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*(AAAI'10)

DMF - Bilinear model (1/2)

In addition to weight vectors, a matrix is presented to model attributes interactions (usually between user and item)

$$\underset{\boldsymbol{W},\boldsymbol{H},\theta}{\operatorname{argmax}} \prod_{(u,i)|r_{ui}\in\delta(\boldsymbol{R})} \mathcal{N}\left(r_{ui} \mid \mu_R = \boldsymbol{x}_u^\top \boldsymbol{A}\boldsymbol{y}_i + \boldsymbol{c}_u^\top \boldsymbol{x}_u + \boldsymbol{d}_i^\top \boldsymbol{y}_i + \boldsymbol{b}, \sigma_R^2\right) p(\boldsymbol{W}|\boldsymbol{X}) p(\boldsymbol{H}|\boldsymbol{Y})$$

The likelihood can be reformed to the following:

$$\mu_R = \boldsymbol{x}_u^\top \boldsymbol{A}\boldsymbol{y}_i + \boldsymbol{c}_u^\top \boldsymbol{x}_u + \boldsymbol{d}_i^\top \boldsymbol{y}_i + \boldsymbol{b} = \widetilde{\boldsymbol{x}}_u^\top \widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{y}}_i$$

which can be seen as learning the matrix only

DMF - Bilinear model (2/2)

Some bilinear models are implicit, for example:

learning $\boldsymbol{w}_u^\top \boldsymbol{h}_i$ where $\boldsymbol{w}_u = \boldsymbol{S}\boldsymbol{x}_u$ and $\boldsymbol{h}_i = \boldsymbol{T}\boldsymbol{y}_i$ is the same as learning

$$\boldsymbol{x}_u^\top (\boldsymbol{S}^\top \boldsymbol{T}) \boldsymbol{y}_i \text{ where } \boldsymbol{A} = \boldsymbol{S}^\top \boldsymbol{T}$$

- Pros: interactions between user attributes and item attributes are captured
- Cons: requires at least two sources of attributes, otherwise it is reduced to linear model

DMF - Bilinear model (3/3)

Example: Regression-based Latent Factor Model (RLFM)

$$\underset{\boldsymbol{W},\boldsymbol{H},\theta}{\operatorname{argmax}} \underbrace{p(\boldsymbol{R} \mid \boldsymbol{W}, \boldsymbol{H}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{\gamma}, \boldsymbol{Z})}_{\text{Likelihood}} \underbrace{p(\boldsymbol{W} \mid \boldsymbol{A}, \boldsymbol{X}) p(\boldsymbol{H} \mid \boldsymbol{B}, \boldsymbol{Y}) p(\boldsymbol{c} \mid \boldsymbol{\alpha}, \boldsymbol{X}) p(\boldsymbol{d} \mid \boldsymbol{\beta}, \boldsymbol{Y})}_{\text{Prior}}$$

$$= \underset{\boldsymbol{W},\boldsymbol{H},\theta}{\operatorname{argmax}} \underbrace{\prod_{(u,i)|r_{ui} \in \delta(\boldsymbol{R})} \mathcal{N}\left(r_{ui} \mid \boldsymbol{w}_u^\top \boldsymbol{h}_i + c_u + d_i + \boldsymbol{\gamma}^\top \boldsymbol{z}_{\pi(u,i)}, \sigma_R^2\right)}_{\text{Matrix factorization using attributes}}$$

$$\underbrace{\prod_u \mathcal{N}\left(\boldsymbol{w}_u \mid \boldsymbol{A}\boldsymbol{x}_u, \boldsymbol{\Sigma}_W\right) \mathcal{N}\left(c_u \mid \boldsymbol{\alpha}^\top \boldsymbol{x}_u, \sigma_c^2\right) \prod_i \mathcal{N}\left(\boldsymbol{h}_i \mid \boldsymbol{B}\boldsymbol{y}_i, \boldsymbol{\Sigma}_H\right) \mathcal{N}\left(d_i \mid \boldsymbol{\beta}^\top \boldsymbol{y}_i, \sigma_d^2\right)}_{\text{Regularization using attributes}}$$

Deepak Agarwal and Bee-Chung Chen. 2009. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (KDD '09)

DMF - Similarity Matrix (1/2)

A matrix is pre-computed to capture similarities between users or items.

The usage of the matrix varies. For example, $S_{ij}||W_i - W_j||_2^2$ regularizes user/item latent factors based on their attribute similarity

- Pros:
  - Human knowledge involves in collaborative filtering (explicitly defines how similarity should be measured)
  - Various choices of similarity measure, e.g. kernel methods
- Cons:
  - High space complexity for similarity matrix (quadratic to number of users/items)

DMF - Similarity Matrix (2/2)

Example: Kernelized Probabilistic Matrix Factorization (KPMF)

$$\underset{\boldsymbol{W},\boldsymbol{H}}{\operatorname{argmax}} \underbrace{p(\boldsymbol{R} \mid \boldsymbol{W}, \boldsymbol{H})}_{\text{Likelihood}} \underbrace{p(\boldsymbol{W} \mid \boldsymbol{X}) p(\boldsymbol{H} \mid \boldsymbol{Y})}_{\text{Prior}}$$

$$= \underset{\boldsymbol{W},\boldsymbol{H}}{\operatorname{argmax}} \underbrace{\prod_{(u,i)|r_{ui} \in \delta(\boldsymbol{R})} \mathcal{N}\left(r_{ui} \mid \boldsymbol{w}_u^\top \boldsymbol{h}_i, \sigma_R^2\right)}_{\text{Matrix factorization}} \underbrace{\prod_k \mathcal{N}\left(\boldsymbol{w}^k \mid \boldsymbol{0}, \boldsymbol{S_X}\right) \prod_l \mathcal{N}\left(\boldsymbol{h}^l \mid \boldsymbol{0}, \boldsymbol{S_Y}\right)}_{\text{Regularization using attributes}}.$$

Kernelized Probabilistic Matrix Factorization: Exploiting Graphs and Side Information Tinghui Zhou, Hanhuai Shan, Arindam Banerjee, and Guillermo Sapiro Proceedings of the 2012 SIAM International Conference on Data Mining

How to use attributes?

- Discriminative matrix factorization
- Generative matrix factorization (GMF)
- Generalized factorization
- Heterogeneous graph

GMF - framework

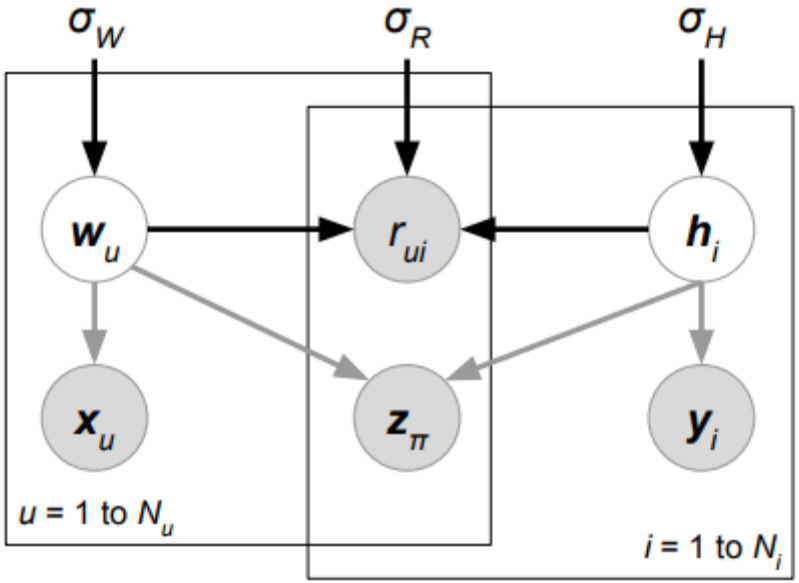
Main idea: distribution of attributes are modeled by latent factors

$$\underset{\boldsymbol{W},\boldsymbol{H}}{\operatorname{argmax}} \underbrace{p\left(\boldsymbol{W},\boldsymbol{H} \mid \boldsymbol{R},\boldsymbol{X}\right)}_{\text{Posterior}} = \underset{\boldsymbol{W},\boldsymbol{H}}{\operatorname{argmax}} \frac{p\left(\boldsymbol{R},\boldsymbol{X} \mid \boldsymbol{W},\boldsymbol{H}\right) p\left(\boldsymbol{W},\boldsymbol{H}\right)}{p\left(\boldsymbol{R},\boldsymbol{X}\right)}$$

$$= \underset{\boldsymbol{W},\boldsymbol{H}}{\operatorname{argmax}}\, p\left(\boldsymbol{R},\boldsymbol{X} \mid \boldsymbol{W},\boldsymbol{H}\right) p\left(\boldsymbol{W},\boldsymbol{H}\right)$$

$$= \underset{\boldsymbol{W},\boldsymbol{H}}{\operatorname{argmax}} \underbrace{p\left(\boldsymbol{R} \mid \boldsymbol{W},\boldsymbol{H}\right) p\left(\boldsymbol{X} \mid \boldsymbol{W},\boldsymbol{H}\right)}_{\text{Likelihood}} \underbrace{p\left(\boldsymbol{W}\right) p\left(\boldsymbol{H}\right)}_{\text{Prior}}.$$

GMF - graphical model



GMF - Matrix Factorization (1/2)

Attribute matrice are factorized by latent matrice

$$\boldsymbol{R} \approx \boldsymbol{W}^\top \boldsymbol{H} \qquad \boldsymbol{X} \approx \boldsymbol{A}^\top \boldsymbol{W}, \boldsymbol{Y} \approx \boldsymbol{B}^\top \boldsymbol{H}$$

$$\underset{\boldsymbol{W},\boldsymbol{H},\boldsymbol{A},\boldsymbol{B}}{\operatorname{argmax}} \prod_{(u,i)|r_{ui}\in\delta(\boldsymbol{R})} \mathcal{N}\left(r_{ui} \mid \mu_R = \boldsymbol{w}_u^\top \boldsymbol{h}_i, \sigma_R^2\right) \prod_{(j,u)} \mathcal{N}\left(x_{ju} \mid \boldsymbol{a}_j^\top \boldsymbol{w}_u, \sigma_X^2\right) \prod_{(v,i)} \mathcal{N}\left(y_{vi} \mid \boldsymbol{b}_v^\top \boldsymbol{h}_i, \sigma_Y^2\right)$$
$$\prod_{(u,i)|r_{ui}\in\delta(\boldsymbol{R})} \mathcal{N}\left(\boldsymbol{z}_{ui} \mid \boldsymbol{w}_u^\top \boldsymbol{C}\boldsymbol{h}_i, \sigma_Z^2\right) p(\boldsymbol{W}) p(\boldsymbol{H}),$$

- Pros: can be easily extended when multiple attribute sources are available
- Cons: difficult to train (jointly learn rating and attribute distribution)

GMF - Matrix Fatorization (2/2)

Example: Collective Matrix Factorization (CMF)

$$\underset{\boldsymbol{W},\boldsymbol{H},\boldsymbol{A},\boldsymbol{B}}{\operatorname{argmax}} \underbrace{p(\boldsymbol{R} \mid \boldsymbol{W},\boldsymbol{H})p(\boldsymbol{X} \mid \boldsymbol{W},\boldsymbol{A})p(\boldsymbol{Y} \mid \boldsymbol{H},\boldsymbol{B})}_{\text{Likelihood}} \underbrace{p(\boldsymbol{W})p(\boldsymbol{H})p(\boldsymbol{A})p(\boldsymbol{B})}_{\text{Prior}}$$

$$= \underset{\boldsymbol{W},\boldsymbol{H},\boldsymbol{A},\boldsymbol{B}}{\operatorname{argmax}} \underbrace{\prod_{(u,i)|r_{ui}\in\delta(\boldsymbol{R})} \mathcal{N}\left(r_{ui} \mid \boldsymbol{w}_u^\top \boldsymbol{h}_i, \sigma_R^2\right)}_{\text{Matrix factorization of } R} \underbrace{\prod_{(j,u)} \mathcal{N}\left(x_{ju} \mid \boldsymbol{a}_j^\top \boldsymbol{w}_u, \sigma_X^2\right)}_{\text{Matrix factorization of } X} \underbrace{\prod_{(v,i)} \mathcal{N}\left(y_{vi} \mid \boldsymbol{b}_v^\top \boldsymbol{h}_i, \sigma_Y^2\right)}_{\text{Matrix factorization of } Y}$$

$$\underbrace{\prod_u \mathcal{N}\left(\boldsymbol{w}_u \mid \boldsymbol{0}, \boldsymbol{\Sigma}_W\right) \prod_i \mathcal{N}\left(\boldsymbol{h}_i \mid \boldsymbol{0}, \boldsymbol{\Sigma}_H\right) \prod_j \mathcal{N}\left(\boldsymbol{a}_j \mid \boldsymbol{0}, \boldsymbol{\Sigma}_A\right) \prod_v \mathcal{N}\left(\boldsymbol{b}_v \mid \boldsymbol{0}, \boldsymbol{\Sigma}_B\right)}_{\text{Regularization}}$$

Ajit P. Singh and Geoffrey J. Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (KDD '08)

GMF - DNN (1/2)

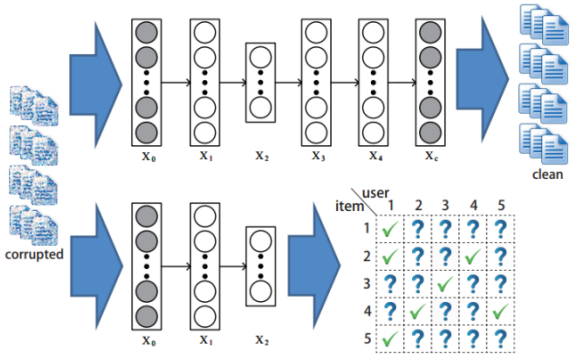
Use autoencoder to model attribute distribution and generate latent factors

$$
\begin{aligned}
\underset{\boldsymbol{W},\boldsymbol{H}}{\operatorname{argmax}}\, p\left(\boldsymbol{W},\boldsymbol{H} \mid \boldsymbol{R},\boldsymbol{X},\tilde{\boldsymbol{X}}\right) &= \underset{\boldsymbol{W},\boldsymbol{H}}{\operatorname{argmax}} \frac{p\left(\boldsymbol{R},\boldsymbol{X} \mid \boldsymbol{W},\boldsymbol{H},\tilde{\boldsymbol{X}}\right) p\left(\boldsymbol{W},\boldsymbol{H} \mid \tilde{\boldsymbol{X}}\right)}{p\left(\boldsymbol{R},\boldsymbol{X} \mid \tilde{\boldsymbol{X}}\right)} \\
&= \underset{\boldsymbol{W},\boldsymbol{H}}{\operatorname{argmax}}\, p\left(\boldsymbol{R},\boldsymbol{X} \mid \boldsymbol{W},\boldsymbol{H},\tilde{\boldsymbol{X}}\right) p\left(\boldsymbol{W},\boldsymbol{H} \mid \tilde{\boldsymbol{X}}\right) \\
&= \underset{\boldsymbol{W},\boldsymbol{H}}{\operatorname{argmax}} \underbrace{\underbrace{p\left(\boldsymbol{R} \mid \boldsymbol{W},\boldsymbol{H},\tilde{\boldsymbol{X}}\right)}_{\text{Matrix factorization}} \underbrace{p\left(\boldsymbol{X} \mid \boldsymbol{W},\boldsymbol{H},\tilde{\boldsymbol{X}}\right)}_{\text{Decoder}\mathcal{D}}}_{\text{Likelihood}} \underbrace{p\left(\boldsymbol{H} \mid \tilde{\boldsymbol{X}}\right) p\left(\boldsymbol{W} \mid \tilde{\boldsymbol{X}}\right)}_{\text{Priors, i.e, Encoder}\mathcal{E}}
\end{aligned}
$$

- Pros:
  - Non-linear mapping is available (with appropriate activation function)
  - Stacked denoising autoencoder makes it more robust to attribute corruption
- Cons:
  - Much more parameters to be tuned

GMF - DNN (2/2)

Example: Collaborative Deep Learning (CDL)



$$\underset{\boldsymbol{W},\boldsymbol{H},\boldsymbol{\theta},\boldsymbol{\phi}}{\operatorname{argmax}} \underbrace{p\left(\boldsymbol{R},\mid \boldsymbol{W},\boldsymbol{H}\right) p\left(\boldsymbol{Y} \mid \boldsymbol{H},\tilde{\boldsymbol{Y}}\right)}_{\text{Likelihood}} \underbrace{p\left(\boldsymbol{H} \mid \tilde{\boldsymbol{Y}}\right) p\left(\boldsymbol{W}\right)}_{\text{Prior}}$$

$$= \underset{\boldsymbol{W},\boldsymbol{H},\boldsymbol{\theta},\boldsymbol{\phi}}{\operatorname{argmax}} \underbrace{\prod_{(u,i)|r_{ui}\in\delta(\boldsymbol{R})} \mathcal{N}\left(r_{ui} \mid \boldsymbol{w}_u^\top \boldsymbol{h}_i, \sigma_R^2\right)}_{\text{Matrix factorization}} \underbrace{\prod_i \mathcal{N}\left(\boldsymbol{y}_i \mid \mathcal{D}_{\boldsymbol{\phi}}(\mathcal{E}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{y}}_i)), \boldsymbol{\Sigma}_Y\right) \mathcal{N}\left(\boldsymbol{h}_i \mid \mathcal{E}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{y}}_i), \boldsymbol{\Sigma}_H\right)}_{\text{Stacked denoising auto-encoder for } \boldsymbol{Y}} \underbrace{\prod_u \mathcal{N}\left(w_u \mid \boldsymbol{0}, \boldsymbol{\Sigma}_W\right)}_{\text{Regularization}}.$$

- The embeddings of autoencoder regularize item latent factors, i.e. $\mathcal{N}\left(\boldsymbol{h}_i \mid \mathcal{E}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{y}}_i), \boldsymbol{\Sigma}_H\right)$
- User latent factors are generated from Gaussian distribution
- Rating prediction is generated by product of user and item latent factors, i.e. $\mathcal{N}\left(r_{ui} \mid \boldsymbol{w}_u^\top \boldsymbol{h}_i, \sigma_R^2\right)$
- The model can be easily extended if user attributes are given

Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD '15)

Comparison of DMF and GMF

	DMF	GMF
Learning target	Rating	Rating, Attributes
Training difficulty	Less difficult	More difficult
Number of models	Many	Few

How to use attributes?

- Discriminative matrix factorization
- Generative matrix factorization
- Generalized factorization (GF)
- Heterogeneous graph

GF - framework (1/2)

Main idea: a more general form of DMF

1. Consider user/item as a kind of categorical attribute (one-hot encoding vector)
2. All attributes (including user, item) can be concatenated as a vector x
   - e.g. given a <u, i, r> pair and let Eu, Ei be the one-hot encoding of user u and item i respectively, the concatenated vector x is <Eu, Ei, Xu, Yi, Zr>
3. The vector is then a typical training example of traditional classification or regression problem, with rating as the label

$$\underset{w}{\operatorname{argmax}} \prod_{r \in \delta(\boldsymbol{R})} \mathcal{N}\left(r \mid \mu_R = \sum_{d=D_m}^{D_M} \sum_{j_1=1}^{N} \sum_{j_2=j_1+1}^{N} \cdots \sum_{j_d=j_{d-1}+1}^{N} w_{j_1 j_2 \ldots j_d} \left(x_{j_1} x_{j_2} \ldots x_{j_d}\right), \sigma_R^2\right)$$

GF - framework (2/2)

Due to large number of parameters in w, which often results in overfitting, w is often a function of latent factors

$$w_{j_1 j_2 \ldots j_d} = f_d\left(\boldsymbol{v}_{j_1}, \boldsymbol{v}_{j_2}, \ldots, \boldsymbol{v}_{j_d}\right)$$

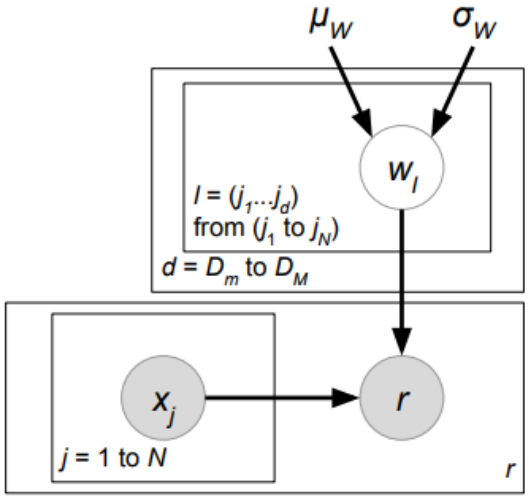
where v_j represents the latent factor for attribute xj

Matrix factorization is a special case of GF:

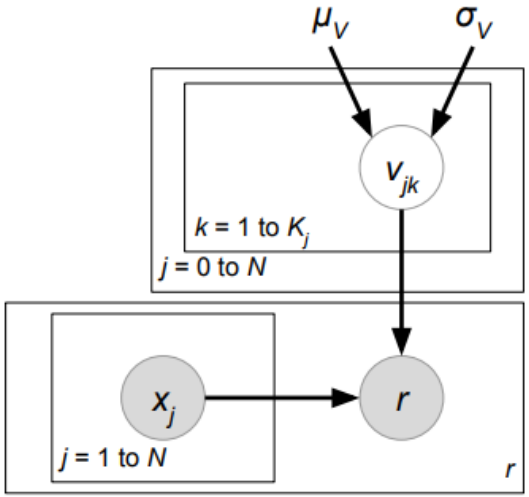
$$\underset{\boldsymbol{v}}{\operatorname{argmax}} \prod_{r_{ui} \in \delta(\boldsymbol{R})} \mathcal{N}\left(\hat{r}_{ui} \mid \mu_R = \sum_{j_1=1}^{N} \sum_{j_2=j_1+1}^{N} \boldsymbol{v}_{j_1}^{\top} \boldsymbol{v}_{j_2} \left(x_{j_1} x_{j_2}\right) = \boldsymbol{v}_u^{\top} \boldsymbol{v}_{N_u+i}, \sigma_R^2\right)$$

where x is the concatenation of user and item one-hot encoding vector

GF - graphical model



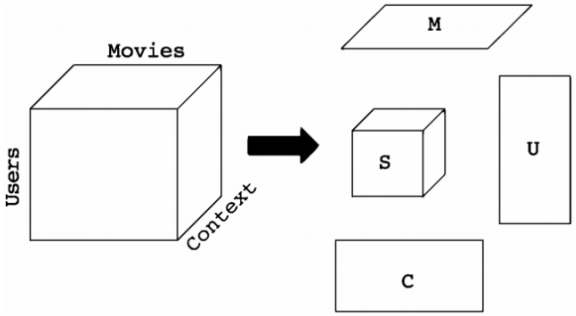
(a) w-weighted generalization



(b) v-approximate generalization

GF - Tensor Factorization

An N-dimensional rating tensor is factorized into a core tensor and several matrice, where each dimension is (user, item, attribute1, attribute2, …)



e.g. 3-dimension tensor (user, item, attr1)

$$U \in \mathbb{R}^{n \times d_U},\ M \in \mathbb{R}^{m \times d_M} \text{ and } C \in \mathbb{R}^{c \times d_C}$$

$$S \in \mathbb{R}^{d_U \times d_M \times d_C}$$

Prediction: $F_{ijk} = S \times_U U_{i*} \times_M M_{j*} \times_C C_{k*}$

Tensor multiplication:

$$T = Y \times_U U$$

=> $T_{ljk} = \sum_{i=1}^{n} Y_{ijk} U_{ij}$

- Pros: highly extendable
- Cons:
  - High space complexity for core tensor (exponential to number of attributes)
  - Attributes are limited to be categorical

Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. 2010. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems* (RecSys '10)

GF - Factorization Machine

The objective function of FM can be simplified from GF framework, which is

$$\mu_R = \underbrace{w_0}_{d=0} + \underbrace{\sum_{l=1}^{N} w_l x_l}_{d=1} + \underbrace{\sum_{j_1=1}^{N} \sum_{j_2=j_1+1}^{N} w_{j_1 j_2} (x_{j_1} x_{j_2})}_{d=2}$$

$$= w_0 + \sum_{l=1}^{N} w_l x_l + \sum_{j_1=1}^{N} \sum_{j_2=j_1+1}^{N} \boldsymbol{v}_{j_1}^{\top} \boldsymbol{v}_{j_2} (x_{j_1} x_{j_2})$$

- Pros: attributes can be numberical
- Cons: higher-order attribute interactions (more than 2) is not considered
- Extension:
  - Multi-view machines (consider higher-oder interactions)
  - Neural Factorization Machines (consider non-linear attribute interactions)

Steffen Rendle. 2010. Factorization Machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining* (ICDM '10)

More on FM

- Lets start with a simple linear regression solution: X= user features f_u and item features g_v, Y= ratings:

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

- Let's try degree-2 polynomial mappings to solve:

$$\min_{w_{t,s}, \forall t,s} \sum_{u,v \in R} \left(r_{u,v} - \sum_{t'=1}^{U} \sum_{s'=1}^{V} w_{t',s'} (f_u)_t (g_v)_s \right)^2$$

- It is likely to overfit since we have many parameters to learn (#Para>#Ratings)

More on FM

- Why not using P^TQ to model W, where P and Q are low-rank matrices

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

How to use attributes?

- Discriminative matrix factorization
- Generative matrix factorization
- Generalized factorization
- Heterogeneous graph (HG)

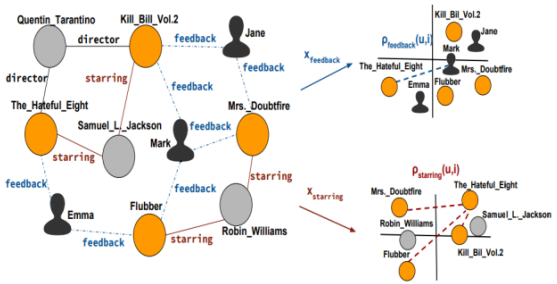
HG

User, item and attributes are represented as nodes in a heterogeneous graph, and the recommendation problem becomes a link prediction problem

- Each user, item, attribute is a node
  a. since nodes are discrete, attributes must be categorical
- Ratings are weights for links ({0, 1} for implicit feedback while real value for explicit feedback)
- Random walk or meta-path algorithms extract similarity from the graph
- MF or other supervised ML algorithms extracts features from similarity information

HG (2/2)

Example: entity2rec



- A knowledge graph with different types of nodes and edges are given
- For each edge type, a subgraph which consists of nodes and edges associated with this edge type only is formed
- For each subgraph, node2vec is applied to extract embeddings
- Similarity can be computed for each <u, i> pair, e.g. cosine similarity
- Similarities belonging to a user from different subgraphs are features for any supervised learning-to-rank algorithm (e.g. Adarank) where labels are observed ratings, i.e. <u, i, r> pair

Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. 2017. entity2rec: Learning User-Item Relatedness from Knowledge Graphs for Top-N Item Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems* (RecSys '17)