Recommender System

[最新！五大顶会2019必读的深度推荐系统与CTR预估相关的论文 - 深度传送门的文章 - 知乎]

Recommender Systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user.

RSs are primarily directed towards individuals who lack sufficient personal experience or competence to evaluate the potentially overwhelming number of alternative items that a Web site, for example, may offer.

Xavier Amatriain discusses the traditional definition and its data mining core.

Traditional definition: The **recommender system** is to estimate a utility function that automatically predicts how a user will like an item.

User Interest is **implicitly** reflected in `Interaction history`, `Demographics` and `Contexts`, which can be regarded as a typical example of data mining. Recommender system should match a context to a collection of information objects. There are some methods called `Deep Matching Models for Recommendation`.
It is an application of machine learning, which is in the *representation + evaluation + optimization* form. And we will focus on the `representation and evaluation`.

- [ ] https://github.com/hongleizhang/RSPapers
- [ ] https://github.com/benfred/implicit
- [ ] https://github.com/YuyangZhangFTD/awesome-RecSys-papers
- [ ] https://github.com/daicoolb/RecommenderSystem-Paper
- [ ] https://github.com/grahamjenson/list_of_recommender_systems
- [ ] https://www.zhihu.com/question/20465266/answer/142867207
- [ ] http://www.mbmlbook.com/Recommender.html
- [x] 直接优化物品排序的推荐算法
- [ ] 推荐系统遇上深度学习
- [ ] Large-Scale Recommender Systems@UTexas
- [ ] Alan Said's publication
- [ ] MyMediaLite Recommender System Library
- [ ] Recommender System Algorithms @ deitel.com
- [ ] Workshop on Recommender Systems: Algorithms and Evaluation
- [ ] Semantic Recommender Systems. Analysis of the state of the topic
- [ ] Recommender Systems (2019/1)
- [ ] Recommender systems & ranking

Evolution of the Recommender Problem
Rating
Ranking
Page Optimization

Context-aware Recommendations

Evaluation of Recommendation System

The evaluation of machine learning algorithms depends on the tasks.
The evaluation of recommendation system can be regarded as some machine learning models such as regression, classification and so on.
We only take the mathematical convenience into consideration in the following methods.
`Gini index, covering rate` and more realistic factors are not discussed in the following content.

- Evaluating recommender systems
- Distance Metrics for Fun and Profit

Collaborative Filtering

There are 3 kinds of collaborative filtering: user-based, item-based and model-based collaborative filtering.

The user-based methods are based on the similarities of users. If user u and v are very similar friends, we may recommend the items which user u bought to the user v and explains it that your friends also bought it.

The item-based methods are based on the similarity of items. If one person added a brush to shopping-list, it is reasonable to recommend some toothpaste to him or her. And we can explain that you bought item X and the people who bought X also bought Y.
And we focus on the model-based collaborative filtering.

- 协同过滤详解
- 深入推荐引擎相关算法 - 协同过滤
- http://topgeek.org/blog/2012/02/10/探索推荐引擎内部的秘密，第-1-部分-推荐引擎初探/
- http://topgeek.org/blog/2012/02/13/探索推荐引擎内部的秘密，第-2-部分-深入推荐引擎/

Matrix Completion

Matrix completion is to complete the matrix X with missing elements, such as

$$\min_{Z} Rank(Z)$$
$$s.t. \sum_{(i,j):Observed} (Z_{(i,j)} - X_{(i,j)})^2 \leq \delta$$

Note that the rank of a matrix is not easy or robust to compute.

We can apply customized PPA to matrix completion problem

$$\min\{\|Z\|_\star\}$$
$$s.t.\ Z_\Omega = X_\Omega$$

We let $Y \in \mathbb{R}^{n \times n}$ be the the Lagrangian multiplier to the constraints $Z_\Omega = X_\Omega$
and Lagrange function is

$$L(Z,Y) = \|Z\|_\star - Y(Z_\Omega - X_\Omega).$$

1. Producing Y^{k+1} by

$$Y^{k+1} = \arg\max_Y L([2Z^k - Z^{k-1}], Y) - \frac{s}{2}\|Y - Y^k\|;$$

2. Producing Z^{k+1} by

$$Z^{k+1} = \arg\min_Z L(Z, Y^{k+1}) + \frac{r}{2}\|Z - Z^k\|.$$

Rahul Mazumder, Trevor Hastie, Robert Tibshirani reformulate it as the following:

$$\min f_\lambda(Z) = \frac{1}{2}\|P_\Omega(Z - X)\|_F^2 + \lambda\|Z\|_\star$$

where X is the observed matrix, P_Ω is a projector and $\|\cdot\|_\star$ is the nuclear norm of matrix.

- A SINGULAR VALUE THRESHOLDING ALGORITHM FOR MATRIX COMPLETION
- Matrix and Tensor Decomposition in Recommender Systems
- Low-Rank Matrix Recovery
- ECE 18-898G: Special Topics in Signal Processing: Sparsity, Structure, and Inference Low-rank matrix recovery via nonconvex optimization

- http://people.eecs.berkeley.edu/~yima/
- New tools for recovering low-rank matrices from incomplete or corrupted observations by Yi Ma@UCB
- Matrix Completion/Sensing as NonConvex Optimization Problem
- Exact Matrix Completion via Convex Optimization
- A SINGULAR VALUE THRESHOLDING ALGORITHM FOR MATRIX COMPLETION
- Customized PPA for convex optimization
- Matrix Completion.m

Maximum Margin Matrix Factorization

> *A novel approach to collaborative prediction is presented, using low-norm instead of low-rank factorizations. The approach is inspired by, and has strong connections to, large-margin linear discrimination. We show how to learn low-norm factorizations by solving a semi-definite program, and present generalization error bounds based on analyzing the Rademacher complexity of low-norm factorizations.*

Consider the soft-margin learning, where we minimize a trade-off between the trace norm of Z and its hinge-loss relative to X_O:

$$\min_Z \|Z\|_\Omega + c \sum_{(ui)\in O} \max(0, 1 - Z_{ui}X_{ui}).$$

And it can be rewritten as a semi-definite optimization problem (SDP):

$$\min_{A,B} \frac{1}{2}(tr(A) + tr(B)) + c \sum_{(ui)\in O} \xi_{ui},$$
$$s.t. \begin{bmatrix} A & X \\ X^T & B \end{bmatrix} \geq 0, Z_{ui}X_{ui} \geq 1 - \xi_{ui}, \xi_{ui} > 0 \, \forall ui \in O$$

where c is a trade-off constant.

- Maximum Margin Matrix Factorization
- Fast Maximum Margin Matrix Factorization for Collaborative Prediction
- Maximum Margin Matrix Factorization by Nathan Srebro

This technique is also called **nonnegative matrix factorization**.

The data sets we more frequently encounter in collaborative prediction problem are of `ordinal ratings` $X_{ij} \in \{1, 2, \ldots, R\}$ such as $\{1, 2, 3, 4, 5\}$.
To relate the real-valued Z_{ij} to the discrete X_{ij}. we use $R - 1$ thresholds $\theta_1, \ldots, \theta_{R-1}$.

SVD and Beyond

If we have collected user u's explicit evaluation score to the item i , $R_{[u][i]}$, and all such data makes up a matrix $R = (R_{[u][i]})$ while the user u cannot evaluate all the item so that the matrix is incomplete and missing much data.
SVD is to factorize the matrix into the multiplication of matrices so that

$$\hat{R} = P^T Q.$$

And we can predict the score $R_{[u][i]}$ via

$$\hat{R}_{[u][i]} = \hat{r}_{u,i} = \langle P_u, Q_i \rangle = \sum_f p_{u,f} q_{i,f}$$

where P_u, Q_i is the u-th column of P and the i-th column of Q, respectively.
And we can define the cost function

$$C(P, Q) = \sum_{(u,i):Observed} (r_{u,i} - \hat{r}_{u,i})^2 = \sum_{(u,i):Observed} (r_{u,i} - \sum_f p_{u,f} q_{i,f})^2$$
$$\arg\min_{P_u, Q_i} C(P, Q)$$

where λ_u is always equal to λ_i.

Additionally, we can add regular term into the cost function to void over-fitting

$$C(P,Q) = \sum_{(u,i):Observed} (r_{u,i} - \sum_f p_{u,f} q_{i,f})^2 + \lambda_u \|P_u\|^2 + \lambda_i \|Q_i\|^2.$$

It is called the regularized singular value decomposition or **Regularized SVD**.

Funk-SVD considers the user's preferences or bias.
It predicts the scores by

$$\hat{r}_{u,i} = \mu + b_u + b_i + \langle P_u, Q_i \rangle$$

where μ, b_u, b_i is biased mean, biased user, biased item, respectively.
And the cost function is defined as

$$\min \sum_{(u,i):Observed} (r_{u,i} - \hat{r}_{u,i})^2 + \lambda(\|P_u\|^2 + \|Q_i\|^2 + \|b_i\|^2 + \|b_u\|^2).$$

SVD ++ predicts the scores by

$$\hat{r}_{u,i} = \mu + b_u + b_i + (P_u + |N(u)|^{-0.5} \sum_{i \in N(u)} y_i) Q_i^T$$

where y_j is the implicit feedback of item j and $N(u)$ is user u's item set.
And it can decompose into 3 parts:

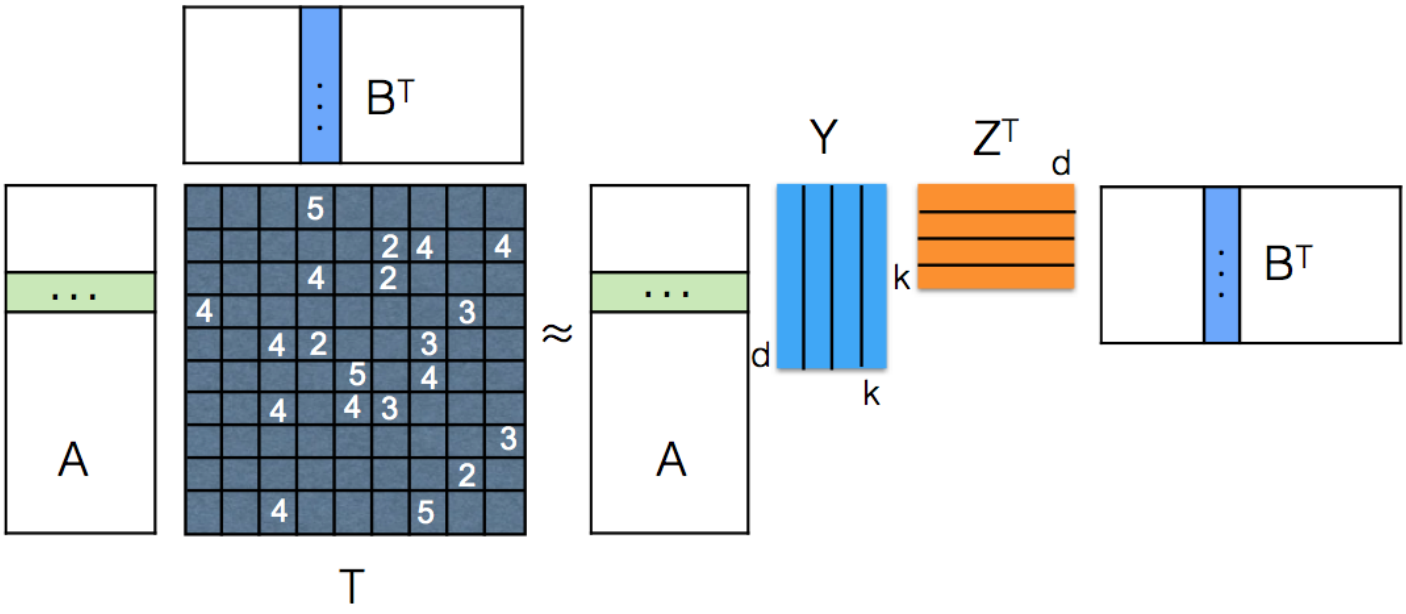
- $\mu + b_u + b_i$ is the base-line prediction;
- $\langle P_u, Q_i \rangle$ is the SVD of rating matrix;
- $\left\langle |N(u)|^{-0.5} \sum_{i \in N(u)} y_i, Q_i \right\rangle$ is the implicit feedback where $N(u)$ is user u's item set, y_j is the implicit feedback of item j.

We learn the values of involved parameters by minimizing the regularized squared error function.

- Biased Regularized Incremental Simultaneous Matrix Factorization@orange3-recommender
- SVD++@orange3-recommender
- 矩阵分解之SVD和SVD++
- SVD++：推荐系统的基于矩阵分解的协同过滤算法的提高
- https://zhuanlan.zhihu.com/p/42269534

Inductive Matrix Completion

One possible improvement of this cost function is that we may design more appropriate loss function other than the squared error function.



utexas.edu

Inductive Matrix Completion (IMC) is an algorithm for recommender systems with side-information of users and items. The IMC formulation incorporates features associated with rows (users) and columns (items) in matrix completion, so that it enables predictions for users or items that were not seen during training, and for which only features are known but no dyadic information (such as ratings or linkages).

IMC assumes that the associations matrix is generated by applying feature vectors associated with
its rows as well as columns to a low-rank matrix Z.
The goal is to recover Z using observations from P.

The inputs x_i, y_j are feature vectors.
The entry $P_{(i,j)}$ of the matrix is modeled as $P_{(i,j)} = x_i^T Z y_j$ and Z is to recover in the form of $Z = WH^T$.

$$\min \sum_{(i,j)\in\Omega} \ell(P_{(i,j)}, x_i^T W H^T y_j) + \frac{\lambda}{2}(\|W\|^2 + \|H\|^2)$$

The loss function ℓ penalizes the deviation of estimated entries from the observations.
And ℓ is diverse such as the squared error $\ell(a,b) = (a-b)^2$, the logistic error $\ell(a,b) = \log(1+\exp(-ab))$.

- Inductive Matrix Completion for Recommender Systems with Side-Information
- Inductive Matrix Completion for Predicting Gene-Diseasev Associations

Probabilistic Matrix Factorization

In linear regression, the least square methods is equivalent to maximum likelihood estimation of the error in standard normal distribution.

Regularized SVD

$C(P,Q) = \sum_{(u,i):Observed}(r_{(u,i)} - \sum_f p_{(u,f)} q_{(i,f)})^2 + \lambda_u \|P_u\|^2 + \lambda_i \|Q_i\|^2$

Probabilistic model

$$r_{u,i} \sim N(\sum_f p_{(u,f)} q_{(i,f)}, \sigma^2), P_u \sim N(0, \sigma_u^2 I), Q_i \sim N(0, \sigma_i^2 I)$$

And σ_u^2 and σ_i^2 is related with the regular term λ_u and λ_u.

So that we can reformulate the optimization problem as maximum likelihood estimation.

- Latent Factor Models for Web Recommender Systems
- Regression-based Latent Factor Models@CS 732 - Spring 2018 - Advanced Machine Learning by Zhi Wei

BellKor's Progamatic Chaos

Until now, we consider the recommendation task as a regression prediction process, which is really common in machine learning.
The boosting or stacking methods may help us to enhance these methods.

> *A key to achieving highly competitive results on the Netflix data is usage of sophisticated blending schemes, which combine the multiple individual predictors into a single final solution. This significant component was managed by our colleagues at the Big Chaos team. Still, we were producing a few blended solutions, which were later incorporated as individual predictors in the final blend. Our blending techniques were applied to three distinct sets of predictors. First is a set of 454 predictors, which represent all predictors of the BellKor's Pragmatic Chaos team for which we have matching Probe and Qualifying results. Second, is a set of 75 predictors, which the BigChaos team picked out of the 454 predictors by forward selection. Finally, a set of 24 BellKor predictors for which we had matching Probe and Qualifying results. from Netflix Prize.*

- https://www.netflixprize.com/community/topic_1537.html
- https://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf
- https://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf

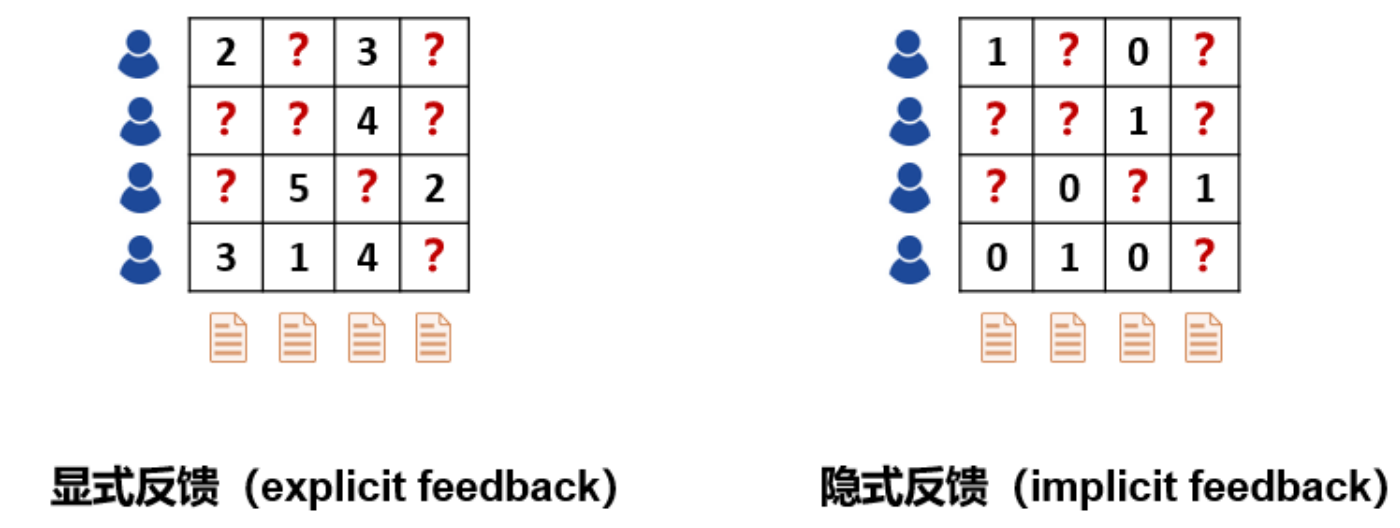
---

Another advantage of collaborative filtering or matrix completion is that even the element of matrix is binary or implicit information such as

- BPR: Bayesian Personalized Ranking from Implicit Feedback,
- Applications of the conjugate gradient method for implicit feedback collaborative filtering,
- Intro to Implicit Matrix Factorization
- a curated list in github.com.

Recommendation with Implicit Information

Explicit and implicit feedback



WRMF is simply a modification of this loss function:

$$C(P,Q)_{WRMF} = \sum_{(u,i):Observed} c_{u,i}(I_{u,i} - \sum_f p_{u,f}q_{i,f})^2 + \lambda_u \|P_u\|^2 + \lambda_i \|Q_i\|^2.$$

We make the assumption that if a user has interacted at all with an item, then $I_{u,i} = 1$. Otherwise, $I_{u,i} = 0$.
If we take $d_{u,i}$ to be the number of times a user u has clicked on an item i on a website, then

$$c_{u,i} = 1 + \alpha d_{u,i}$$

where α is some hyperparameter determined by cross validation.
The new term in cost function $C = (c_{u,i})$ is called confidence matrix.

WRMF does not make the assumption that a user who has not interacted with an item does not like the item. WRMF does assume that that user has a negative preference towards that item, but we can choose how confident we are in that assumption through the confidence hyperparameter.

Alternating least square (**ALS**) can give an analytic solution to this optimization problem by setting the gradients equal to 0s.

- http://nicolas-hug.com/blog/matrix_facto_1
- http://nicolas-hug.com/blog/matrix_facto_2
- http://nicolas-hug.com/blog/matrix_facto_3
- Collaborative Filtering for Implicit Feedback Datasets
- Alternating Least Squares Method for Collaborative Filtering
- Implicit Feedback and Collaborative Filtering

Collaborative Less-is-More Filtering

Sometimes, the information of user we could collect is implicit such as the clicking at some item.

In `CLiMF` the model parameters are learned by directly maximizing the Mean Reciprocal Rank (MRR).

Its objective function is

$$F(U,V) = \sum_{i=1}^{M}\sum_{j=1}^{N} Y_{ij}[\ln g(U_i^T V_j) + \sum_{k=1}^{N} \ln(1 - Y_{ij} g(U_i^T V_k - U_i^T V_j))] - \frac{\lambda}{2}(\|U\|^2 + \|V\|^2)$$

where M, N is the number of users and items, respectively. Additionally, λ denotes the regularization coefficient and Y_{ij} denotes the binary relevance score of item j to user i, i.e.,$Y_{ij} = 1$ if item j is relevant to user j, 0 otherwise. The function g is logistic function $g(x) = \frac{1}{1+\exp(-x)}$.
The vector U_i denotes a d-dimensional latent factor vector for user i, and V_j a d-dimensional latent factor vector for item i.

Numbers		Factors		Others	
M	the number of users	U_i	latent factor vector for user i	Y_{ij}	binary relevance score
N	the number of items	V_j	latent factor vector for item i	f	logistic function

We use stochastic gradient ascent to maximize the objective function.

- Collaborative Less-is-More Filtering@orange3-recommendation
- https://dl.acm.org/citation.cfm?id=2540581
- Collaborative Less-is-More Filtering python Implementation
- CLiMF: Collaborative Less-Is-More Filtering

- https://www.cnblogs.com/Xnice/p/4522671.html
- https://blog.csdn.net/turing365/article/details/80544594
- https://en.wikipedia.org/wiki/Collaborative_filtering
- Matrix_factorization for recommender system
- http://www.cnblogs.com/DjangoBlog/archive/2014/06/05/3770374.html
- https://www.acemap.info/author/page?AuthorID=7E61F31B
- Fast Python Collaborative Filtering for Implicit Feedback Datasets
- Intro to Implicit Matrix Factorization: Classic ALS with Sketchfab Models
- Learning to Rank Sketchfab Models with LightFM
- Finding Similar Music using Matrix Factorization
- Faster Implicit Matrix Factorization
- CUDA Tutorial: Implicit Matrix Factorization on the GPU
- Top-N Recommendations from Implicit Feedback Leveraging Linked Open Data ?
- DiFacto — Distributed Factorization Machines

Hyperbolic Recommender Systems

Many well-established recommender systems are based on representation learning in Euclidean space.
In these models, matching functions such as the Euclidean distance or inner product are typically used for computing similarity scores between user and item embeddings. This paper investigates the notion of learning user and item representations in hyperbolic space.

Given a user u and an item v that are both lying in the Poincare ball B^n,
the distance between two points on *P* is given by

$$d_p(x,y) = cosh^{-1}(1 + 2\frac{\|(x-y\|^2}{(1-\|x\|^2)(1-\|y\|^2)}).$$

`HyperBPR` leverages BPR pairwise learning to minimize the pairwise ranking loss between the positive and negative items.
Given a user u and an item v that are both lying in Poincare ball B^n, we take:

$$\alpha(u,v) = f(d_p(u,v)).$$

The objective function is defined as follows:

$$\arg\min_{\Theta} \sum_{i,j,k} -\ln(\sigma\{\alpha(u_i,v_j) - \alpha(u_i,v_k)\}) + \lambda\|\Theta\|_2^2$$

where (i,j,k) is the triplet that belongs to the set D that
contains all pairs of positive and negative items for each
user; σ is the logistic sigmoid function; Θ represents the model parameters; and λ is the regularization parameter.

The parameters of our model are learned by using `RSGD`.

- https://arxiv.org/abs/1111.5280
- https://arxiv.org/abs/1809.01703
- https://arxiv.org/abs/1902.0864

Deep Learning and Recommender System

Deep learning is powerful in processing visual and text information so that it helps to find the interests of users such as Deep Interest Network, xDeepFM and more.

Deep learning models for recommender system may come from the restricted Boltzman machine.
And deep learning models are powerful information extractors.
Deep learning is really popular in recommender system such as spotlight.

Factorization Machines(FM)

The matrix completion used in recommender system are linear combination of some features such as regularized SVD.
The model equation for a factorization machine of degree $d = 2$ is defined as

$$\hat{y} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$$
$$= w_0 + \langle w, x \rangle + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$$

where the model parameters that have to be estimated are

$$w_0 \in \mathbb{R}, w \in \mathbb{R}^n, V \in \mathbb{R}^{n \times k}.$$

And $\langle \cdot, \cdot \rangle$ is the dot (inner) product of two vectors so that $\langle v_i, v_j \rangle = \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}$.
A row v_i within V describes the i-th latent variable with k factors for x_i.

And the linear regression $w_0 + \sum_{i=1}^{n} w_i x_i$ is called `the first order part`; the pair-wise interactions between features $\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$ is called the `second order part`.

- https://blog.csdn.net/g11d111/article/details/77430095
- Factorization Machines for Recommendation Systems
- http://www.52caml.com/head_first_ml/ml-chapter9-factorization-family/
- https://www.cnblogs.com/pinard/p/6370127.html

Field-aware Factorization Machine(FFM)

In FMs, every feature has only one latent vector to learn the latent effect with any other features.
In FFMs, each feature has several latent vectors. Depending on the field of other features, one of them is used to do the inner product.
Mathematically,

$$\hat{y} = \sum_{j_1=1}^{n} \sum_{j_2=i+1}^{n} \langle v_{j_1,f_2}, v_{j_2,f_1} \rangle x_{j_1} x_{j_2}$$

where f_1 and f_2 are respectively the fields of j_1 and j_2.

- https://www.csie.ntu.edu.tw/~cjlin/papers/ffm.pdf
- https://blog.csdn.net/mmc2015/article/details/51760681

Wide & Deep Model

The output of this model is

$$P(Y = 1|x) = \sigma(W_{wide}^T [x, \phi(x)] + W_{deep}^T \alpha^{(lf)} + b)$$

where the `wide` part deal with the categorical features such as user demographics and the `deep` part deal with continuous features.

- https://arxiv.org/pdf/1606.07792.pdf
- Wide & Deep Learning: Better Together with TensorFlow, Wednesday, June 29, 2016
- https://www.jianshu.com/p/dbaf2d9d8c94
- https://www.sohu.com/a/190148302_115128

Deep FM

`DeepFM` ensembles FM and DNN and to learn both second order and higher-order feature interactions:

$$\hat{y} = \sigma(y_{FM} + y_{DNN})$$

where σ is the sigmoid function so that $\hat{y} \in [0, 1]$ is the predicted CTR, y_{FM} is the output of FM component, and y_{DNN} is the output of deep component.

The **FM component** is a factorization machine and the output of FM is the summation of an `Addition` unit and a number of `Inner Product` units:

$$\hat{y} = \langle w, x \rangle + \sum_{j_1=1}^{n} \sum_{j_2=i+1}^{n} \langle v_i, v_j \rangle x_{j_1} x_{j_2}.$$

The **deep component** is a `feed-forward neural network`, which is used to learn high-order feature interactions. There is a personal guess that the component function in activation function e^x can expand in the polynomials form $e^x = 1 + x + \frac{x^2}{2!} + \ldots, + \frac{x^n}{n!} + \ldots$, which include all the order of interactions.

We would like to point out the two interesting features of this network structure:

1. while the lengths of different input field vectors can be different, their embeddings are of the same size (k);
2. the latent feature vectors (V) in FM now server as network weights which are learned and used to compress the input field vectors to the embedding vectors.

It is worth pointing out that FM component and deep component share the same feature embedding, which brings two important benefits:

1. it learns both low- and high-order feature interactions from raw features;
2. there is no need for expertise feature engineering of the input.

- https://zhuanlan.zhihu.com/p/27999355
- https://zhuanlan.zhihu.com/p/25343518
- https://zhuanlan.zhihu.com/p/32127194

- https://arxiv.org/pdf/1703.04247.pdf
- https://blog.csdn.net/John_xyz/article/details/78933253#deep-fm

Neural Factorization Machines

$$\hat{y} = w_0 + \langle w, x \rangle + f(x)$$

where the first and second terms are the linear regression part similar to that for FM, which models global bias of data and weight
of features. The third term $f(x)$ is the core component of NFM
for modelling feature interactions, which is a `multi-layered feedforward neural network`.

`B-Interaction Layer` including `Bi-Interaction Pooling` is an innovation in artificial neural network.

https://i.ooxx.ooo

https://i.ooxx.ooo

- https://www.comp.nus.edu.sg/~xiangnan/papers/sigir17-nfm.pdf
- http://staff.ustc.edu.cn/~hexn/
- https://github.com/hexiangnan/neural_factorization_machine

Attentional Factorization Machines

Attentional Factorization Machine (AFM) learns the importance of each feature interaction from data via a neural attention network.

We employ the attention mechanism on feature interactions by performing a weighted sum on the interacted vectors:

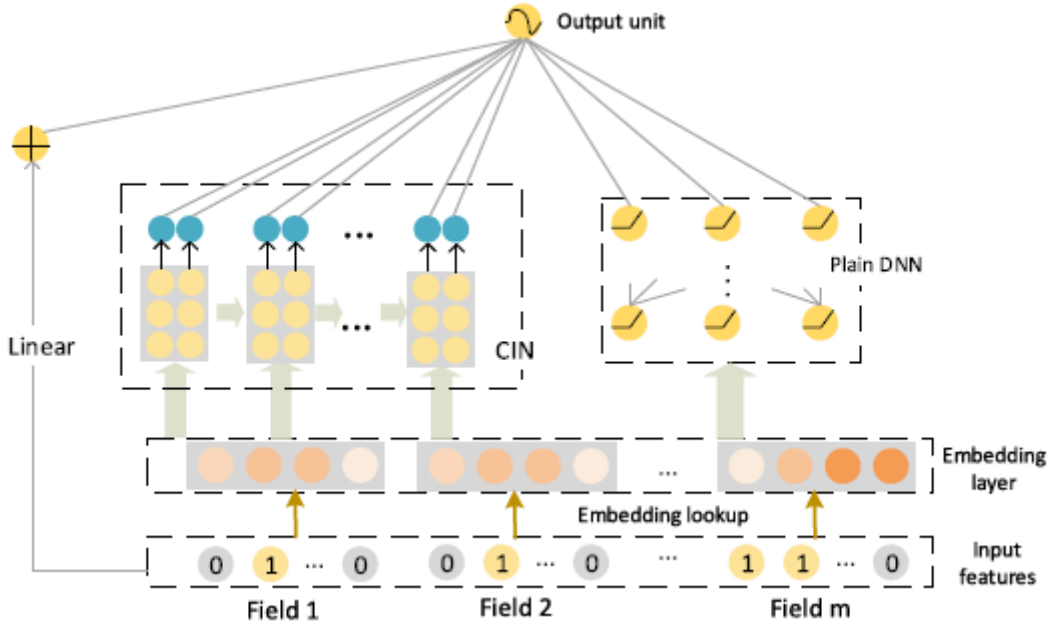
$$\sum_{(i,j)} a_{(i,j)}(V_i \odot V_j)x_i x_j$$

where $a_{i,j}$ is the attention score for feature interaction.

- https://www.comp.nus.edu.sg/~xiangnan/papers/ijcai17-afm.pdf
- http://blog.leanote.com/post/ryan_fan/Attention-FM（AFM）

xDeepFM

`Compressed Interaction Network(CIN)`



- [x] KDD 2018 | 推荐系统特征构建新进展：极深因子分解机模型
- [ ] https://arxiv.org/abs/1803.05170
- [ ] http://kubicode.me/2018/09/17/Deep Learning/eXtreme-Deep-Factorization-Machine/
- [ ] 推荐系统遇上深度学习(二十二)--DeepFM升级版XDeepFM模型强势来袭！

Restricted Boltzmann Machines for Collaborative Filtering(RBM)

Let V be a $K \times m$ observed binary indicator matrix with $v_i^k = 1$ if the user rated item i as k and 0 otherwise.
We also let h_j, $j = 1, \ldots, F$, be the binary values of hidden (latent) variables, that can be thought of as representing stochastic binary features that have different values for different users.

We use a conditional multinomial distribution (a “softmax”) for modeling each column of the observed "visible" binary rating matrix V and a conditional
Bernoulli distribution for modeling "hidden" user features h:

$$p(v_i^k = 1 \mid h) = \frac{\exp(b_i^k + \sum_{j=1}^{F} h_j W_{i,j}^k)}{\sum_{l=1}^{K} \exp(b_i^k + \sum_{j=1}^{F} h_j W_{i,j}^l)}$$

$$p(h_j = 1 \mid V) = \sigma(b_j + \sum_{i=1}^{m} \sum_{k=1}^{K} v_i^k W_{i,j}^k)$$

where $\sigma(x) = \frac{1}{1+exp(-x)}$ is the logistic function, $W_{i,j}^k$ is is a symmetric interaction parameter between feature j and rating k of item i, b_i^k is the bias of rating k for item i, and b_j is the bias of feature j.

The marginal distribution over the visible ratings V is

$$p(V) = \sum_h \frac{\exp(-E(V,h))}{\sum_{V',h'} \exp(-E(V',h'))}$$

with an "energy" term given by:

$$E(V,h) = -\sum_{i=1}^{m}\sum_{j=1}^{F}\sum_{k=1}^{K} W_{i,j}^{k} h_j v_i^k - \sum_{i=1}^{m}\sum_{k=1}^{K} v_i^k b_i^k - \sum_{j=1}^{F} h_j b_j.$$

The items with missing ratings do not make any contribution to the energy function

The parameter updates required to perform gradient ascent in the log-likelihood over the visible ratings V can be obtained

$$\Delta W_{i,j}^{k} = \epsilon \frac{\partial \log(p(V))}{\partial W_{i,j}^{k}}$$

where ϵ is the learning rate.
The authors put a `Contrastive Divergence` to approximate the gradient.

We can also model “hidden” user features h as Gaussian latent variables:

$$p(v_i^k = 1|h) = \frac{\exp(b_i^k + \sum_{j=1}^{F} h_j W_{i,j}^{k})}{\sum_{l=1}^{K} \exp(b_i^k + \sum_{j=1}^{F} h_j W_{i,j}^{l})}$$

$$p(h_j = 1|V) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(\frac{(h - b_j - \sigma_j \sum_{i=1}^{m}\sum_{k=1}^{K} v_i^k W_{i,j}^{k})^2}{2\sigma_j^2}\right)$$

where σ_j^2 is the variance of the hidden unit j.

- https://www.cnblogs.com/pinard/p/6530523.html
- https://www.cnblogs.com/kemaswill/p/3269138.html
- https://www.cs.toronto.edu/~rsalakhu/papers/rbmcf.pdf
- http://www.cs.toronto.edu/~fritz/absps/cdmiguel.pdf
- http://deeplearning.net/tutorial/rbm.html
- RBM notebook form Microsoft

AutoRec

AutoRec is a novel `autoencoder` framework for collaborative filtering (CF). Empirically, AutoRec’s compact and efficiently trainable model outperforms state-of-the-art CF techniques (biased matrix factorization, RBMCF and LLORMA) on the Movielens and Netflix datasets.

Formally, the objective function for the Item-based AutoRec (I-AutoRec) model is, for regularisation strength $\lambda > 0$,

$$\min_{\theta} \sum_{i=1}^{n} \|r^i - h(r^i|\theta)\|_O^2 + \frac{1}{2}(\|W\|_F^2 + \|V\|_F^2)$$

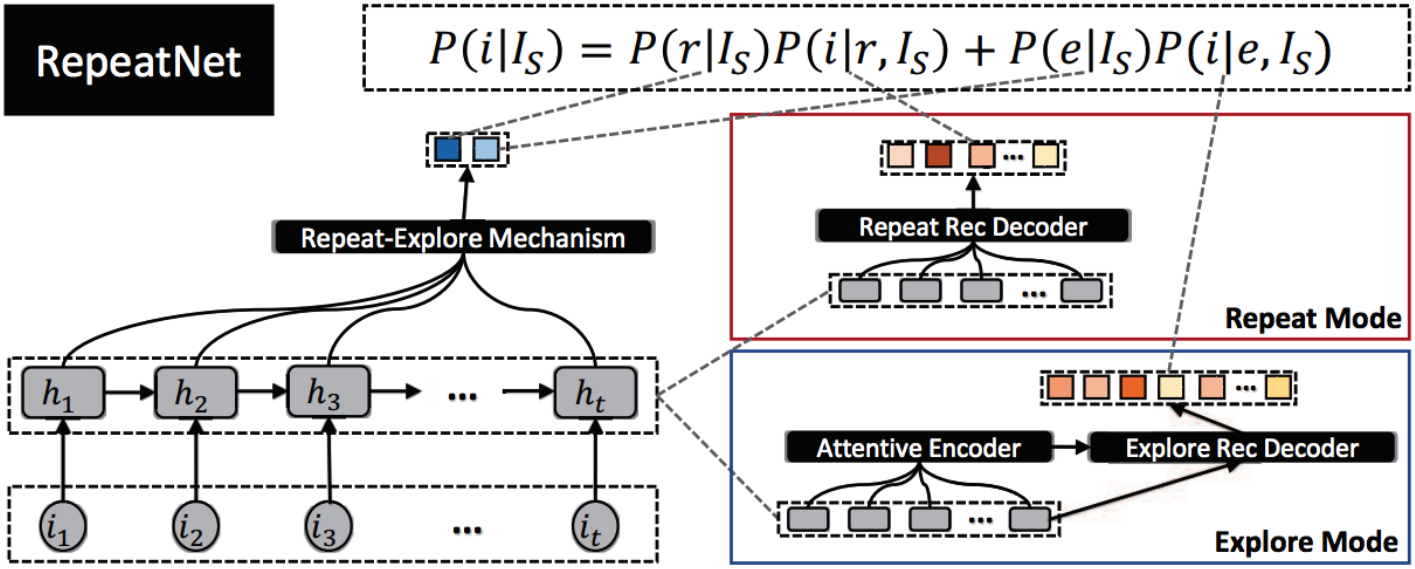
where $\{r^i \in \mathbb{R}^d, i = 1, 2, \ldots, n\}$ is partially observed vector and $\|\cdot\|_o^2$ means that we only consider the contribution of observed ratings.
The function $h(r|\theta)$ is the reconstruction of input $r \in \mathbb{R}^d$:

$$h(r|\theta) = f(W \cdot g(Vr + \mu) + b)$$

for for activation functions f, g as described in dimension reduction. Here $\theta = \{W, V, r, b\}$.

- https://blog.csdn.net/studyless/article/details/70880829
- http://users.cecs.anu.edu.au/~akmenon/papers/autorec/autorec-paper.pdf

---



http://kubicode.me

- https://github.com/hwwang55/DKN
- https://www.csie.ntu.edu.tw/~b97053/paper/Rendle2010FM.pdf
- https://www.cnblogs.com/pinard/p/6370127.html
- https://www.jianshu.com/p/6f1c2643d31b
- https://blog.csdn.net/John_xyz/article/details/78933253
- https://zhuanlan.zhihu.com/p/38613747
- https://www.infosec-wiki.com/?p=394011
- Recommender Systems with Deep Learning
- 深度学习在序列化推荐中的应用
- 深入浅出 Factorization Machine 系列
- 论文快读 - Deep Neural Networks for YouTube Recommendations

Deep Geometric Matrix Completion

It’s easy to observe how better matrix completions can be achieved by considering the sparse matrix as defined over two different graphs:
a user graph and an item graph. From a signal processing point of view, the matrix X
can be considered as a bi-dimensional signal defined over two distinct domains.
Instead of recurring to multigraph convolutions realized over the entire matrix X, two
independent single-graph GCNs (graph convolution networks) can be applied on matrices W and H.

Given the aforementioned multi-graph convolutional layers,
the last step that remains concerns the choice of the architecture to use for reconstructing the missing information.
Every (user, item) pair in the multi-graph approach and every user/item in the separable
one present in this case an independent state, which is updated (at every step) by means of the features produced by the selected GCN.

- graph convolution network有什么比较好的应用task？ - superbrother的回答 - 知乎
- https://arxiv.org/abs/1704.06803
- Deep Geometric Matrix Completion: a Geometric Deep Learning approach to Recommender Systems
- Talk: Deep Geometric Matrix Completion

Deep Matching Models for Recommendation

It is essential for the recommender system to find the item which matches the users' demand. Its difference from web search is that recommender system provides item information even if the users' demands or generally interests are not provided. It sounds like modern crystal ball to read your mind.

In A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems the authors propose to extract rich features from user’s browsing
and search histories to model user’s interests. The underlying assumption is that, users’ historical online activities
reflect a lot about user’s background and preference, and
therefore provide a precise insight of what items and topics users might be interested in.

- http://sonyis.me/dnn.html
- https://akmenon.github.io/
- https://sigir.org/sigir2018/program/tutorials/
- http://staff.ustc.edu.cn/~hexn/
- Framework and Principles of Matching Technologies
- A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems
- Collaborative Deep Learning for Recommender Systems

Social Recommendation

We present a novel framework for studying recommendation algorithms in terms of the
‘jumps’ that they make to connect people to artifacts. This approach emphasizes reachability via an algorithm within the `implicit graph structure` underlying a recommender
dataset and allows us to consider questions relating algorithmic parameters to properties of the datasets.

- [ ] Social Media Mining: An Introduction
- [ ] http://dmml.asu.edu/smm/slide/SMM-Slides-ch9.pdf
- [ ] Do Social Explanations Work? Studying and Modeling the Effects of Social Explanations in Recommender Systems

- [ ] Social Recommendation With Evolutionary Opinion Dynamics
- [ ] Workshop on Responsible Recommendation
- [ ] A Probabilistic Model for Using Social Networks in Personalized Item Recommendation
- [ ] Product Recommendation and Rating Prediction based on Multi-modal Social Networks

Knowledge Graph and Recommender System

- [ ] https://www.msra.cn/zh-cn/news/features/embedding-knowledge-graph-in-recommendation-system-i
- [ ] https://www.msra.cn/zh-cn/news/features/embedding-knowledge-graph-in-recommendation-system-ii
- [ ] https://www.msra.cn/zh-cn/news/features/explainable-recommender-system-20170914
- [ ] 深度学习与知识图谱在美团搜索广告排序中的应用实践

Reinforcement Learning and Recommender System

- Deep Reinforcement Learning for Page-wise Recommendations
- A Reinforcement Learning Framework for Explainable Recommendation

- Generative Adversarial User Model for Reinforcement Learning Based Recommendation System
- Adversarial Personalized Ranking for Recommendation
- Adversarial Training Towards Robust Multimedia Recommender System
- xplore, Exploit, and Explain: Personalizing Explainable Recommendations with Bandits

---

- [ ] Deep Learning Meets Recommendation Systems
- [ ] Using Keras' Pretrained Neural Networks for Visual Similarity Recommendations
- [ ] Recommending music on Spotify with deep learning

Traditional Approaches	Beyond Traditional Methods
Collaborative Filtering	Tensor Factorization & Factorization Machines
Content-Based Recommendation	Social Recommendations
Item-based Recommendation	Learning to rank
Hybrid Approaches	MAB Explore/Exploit

Ensemble Methods for Recommender System

The RecSys can be considered as some regression or classification tasks, so that we can apply the ensemble methods to these methods as `BellKor's Progamatic Chaos` used the blended solution to win the prize.
In fact, its essence is bagging or blending, which is one sequential ensemble strategy in order to avoid over-fitting or reduce the variance.

In this section, the boosting is the focus, which is to reduce the error and boost the performance from a weaker learner.

There are two common methods to construct a stronger learner from a weaker learner: (1) reweight the samples and learn from the error: AdaBoosting; (2) retrain another learner and learn to approximate the error: Gradient Boosting.

- General Functional Matrix Factorization Using Gradient Boosting

Gradient Boosting Factorization Machines

`Gradient Boosting Factorization Machine (GBFM)` model is to incorporate feature selection algorithm with Factorization Machines into a unified framework.

Gradient Boosting Factorization Machine Model

- *Input: Training Data* $S = \{(\mathbf{x}_i, y_i)\}$.
- *Output:* $\hat{y}_S = y_0(x) + \sum_{s=1}^{S} \langle v_{si}, v_{sj} \rangle$.
- *Initialize rating prediction function as* $\hat{y}_0(x)$
- *for* $s = 1 \rightarrow S$ *do*
- *1. Select interaction feature* C_p *and* C_q *from Greedy Feature Selection Algorithm;*
- *2. Estimate latent feature matrices* V_p *and* V_q*;*
- *3. Update* $\hat{y}_s(\mathrm{x}) = \hat{y}_{s-1}(\mathrm{x}) + \sum_{i \in C_p} \sum_{j \in C_q} \mathbb{I}[i, j \in \mathrm{x}] \left\langle V_p^i, V_q^j \right\rangle$
- *end for*

where s is the iteration step of the learning algorithm. At step s, we greedily select two interaction features C_p and C_q where $\mathbb{I}$ is the indicator function, the value is 1 if the condition holds otherwise 0.

Greedy Feature Selection Algorithm

From the view of gradient boosting machine, at each step s, we would like to search a function f in the function space F that minimize the objective function:

$$L = \sum_i \ell(\hat{y}_s(\mathrm{x}_i), y_i) + \Omega(f)$$

where $\hat{y}_s(\mathrm{x}) = \hat{y}_{s-1}(\mathrm{x}) + \alpha_s f_s(\mathrm{x})$.

We heuristically assume that the function f has the following form:

$$f_\ell(\mathrm{x}) = \prod_{t=1}^{\ell} q_{C_i(t)}(\mathrm{x})$$

where the function q maps latent feature vector x to real value domain

$$q_{C_i(t)}(\mathrm{x}) = \sum_{j \in C_i(t)} \mathbb{I}[j \in \mathrm{x}] w_{tj}.$$

It is hard for a general convex loss function ℓ to search function f to optimize the objective function: $L = \sum_i \ell(\hat{y}_s(\mathrm{x}_i), y_i) + \Omega(f)$.

The most common way is to approximate it by least-square minimization, i.e., $\ell = \|\cdot\|_2^2$. Like in `xGBoost`, it takes second order Taylor expansion of the loss function ℓ and problem is finalized to find the i(t)-th feature which:

$$\arg\min_{i(t)\in\{0,\ldots,m\}} \sum_{i=1}^{n} h_i \left(\frac{g_i}{h_i} - f_{t-1}(\mathrm{x}_i) q_{C_i(t)}(\mathrm{x}_i)\right)^2 + \|\theta\|_2^2$$

where the negative first derivative and the second derivative at instance i as g_i and h_i.

- Gradient boosting factorization machines

BoostFM

`BoostFM` integrates boosting into factorization models during the process of item ranking.
Specifically, BoostFM is an adaptive boosting framework that linearly combines multiple homogeneous component recommender system,
which are repeatedly constructed on the basis of the individual FM model by a re-weighting scheme.

BoostFM

- *Input: The observed context-item interactions or Ttraining Data $S = \{(\mathbf{x}_i, y_i)\}$ parameters E and T.*
- *Output: The strong recommender g^T.*
- *Initialize $Q_{ci}^{(t)} = 1/|S|, g^{(0)} = 0, \forall (c,i) \in S$.*
- *for $s = 1 \rightarrow T$ do*
- *1. Create component recommender;*
- *2. Compute the ranking accuracy;*
- *3. Compute the coefficient;*
- *4. Create the strong recommender;*
- *5. Update weight distribution;*
- *end for*

- BoostFM: Boosted Factorization Machines for Top-N Feature-based Recommendation
- http://wnzhang.net/
- https://fajieyuan.github.io/
- https://www.librec.net/luckymoon.me/

- The author’s final accepted version.

Adaptive Boosting Personalized Ranking (AdaBPR)

`AdaBPR (Adaptive Boosting Personalized Ranking)` is a boosting algorithm for top-N item recommendation using users' implicit feedback.
In this framework, multiple homogeneous component recommenders are linearly combined to achieve more accurate recommendation.
The component recommenders are learned based on a re-weighting strategy that assigns a dynamic weight to each observed user-item interaction.

Here explicit feedback refers to users' ratings to items while implicit feedback is derived
from users' interactions with items, e.g., number of times a user plays a song.

The primary idea of applying boosting for item recommendation is to learn a set of homogeneous component recommenders and then create an ensemble of the component recommenders to predict users' preferences.

Here, we use a linear combination of component recommenders as the final recommendation model

$$f = \sum_{t=1}^{T} \alpha_t f_t.$$

In the training process, AdaBPR runs for T rounds, and the component recommender f_t is created at t-th round by

$$\arg\min_{f_t \in \mathbb{H}} \sum_{(u,i) \in \mathbb{O}} \beta_u \exp\{-E(\pi(u, i, \sum_{n=1}^{t} \alpha_n f_n))\}$$

where the notations are listed as follows:

- $\mathbb{H}$ is the set of possible component recommenders such as collaborative ranking algorithms;
- $E(\pi(u, i, f))$ denotes the ranking accuracy associated with each observed interaction pair;
- $\pi(u, i, f)$ is the rank position of item i in the ranked item list of u, resulted by a learned ranking model f;
- $\mathbb{O}$ is the set of all observed user-item interactions;
- β_u is defined as reciprocal of the number of user u's historical items $\beta_u = \frac{1}{|V_u^+|}$ (V_u^+ is the historical items of u).

- A Boosting Algorithm for Item Recommendation with Implicit Feedback
- The review @Arivin's blog

Explainable Recommendations

- Explainable Recommendation and Search @ rutgers
- Explainable Recommendation: A Survey and New Perspectives
- Explainable Entity-based Recommendations with Knowledge Graphs
- 2018 Workshop on ExplainAble Recommendation and Search (EARS 2018)
- EARS 2019

- ExplainAble Recommendation and Search (EARS)
- TEM: Tree-enhanced Embedding Model for Explainable Recommendation
- EARS 2019
- https://ears2019.github.io/
- Explainable Recommendation for Self-Regulated Learning
- Dynamic Explainable Recommendation based on Neural Attentive Models
- https://github.com/fridsamt/Explainable-Recommendation
- Explainable Recommendation for Event Sequences: A Visual Analytics Approach by Fan Du
- https://wise.cs.rutgers.edu/code/
- http://www.cs.cmu.edu/~rkanjira/thesis/rose_proposal.pdf

- [ ] https://wsdm2019-dapa.github.io/#section-ketnotes
- [ ] https://github.com/robi56/Deep-Learning-for-Recommendation-Systems
- [ ] https://github.com/wzhe06/Reco-papers
- [ ] https://github.com/hongleizhang/RSPapers
- [ ] https://github.com/hongleizhang/RSAlgorithms
- [ ] https://github.com/cheungdaven/DeepRec
- [ ] https://github.com/cyhong549/DeepFM-Keras
- [ ] https://github.com/grahamjenson/list_of_recommender_systems
- [ ] https://zhuanlan.zhihu.com/p/26977788
- [ ] https://zhuanlan.zhihu.com/p/45097523
- [ ] https://www.zhihu.com/question/20830906
- [ ] https://www.zhihu.com/question/56806755/answer/150755503

- DLRS 2018 : 3rd Workshop on Deep Learning for Recommender Systems
- Deep Learning based Recommender System: A Survey and New Perspectives
- 5^{th} International Workshop on Machine Learning Methods for Recommender Systems
- MoST-Rec 2019: Workshop on Model Selection and Parameter Tuning in Recommender Systems
- 2018 Personalization, Recommendation and Search (PRS) Workshop
- WIDE & DEEP RECOMMENDER SYSTEMS AT PAPI
- Interdisciplinary Workshop on Recommender Systems
- 2nd FATREC Workshop: Responsible Recommendation

Implementation

- [ ] https://github.com/maciejkula/spotlight
- [ ] https://github.com/Microsoft/Recommenders
- [ ] https://github.com/cheungdaven/DeepRec
- [ ] https://github.com/alibaba/euler
- [ ] https://github.com/alibaba/x-deeplearning/wiki/
- [ ] https://github.com/lyst/lightfm

- [ ] Surprise: a Python scikit building and analyzing recommender systems
- [ ] Orange3-Recommendation: a Python library that extends Orange3 to include support for recommender systems.
- [ ] MyMediaLite: a recommender system library for the Common Language Runtime
- [ ] http://www.mymediaproject.org/

Workshop: Building Recommender Systems w/ Apache Spark 2.x

A Leading Java Library for Recommender Systems

lenskit: Python Tools for Recommender Experiments

Samantha - A generic recommender and predictor server

Computational Advertising

Online advertising has grown over the past decade to over $26 billion in recorded revenue in 2010. The revenues generated are based on different pricing models that can be fundamentally grouped into two types: cost per (thousand) impressions (CPM) and cost per action (CPA), where an action can be a click, signing up with the advertiser, a sale, or any other measurable outcome. A web publisher generating revenues by selling advertising space on its site can offer either a CPM or CPA contract. We analyze the conditions under which the two parties agree on each contract type, accounting for the relative risk experienced by each party.

The information technology industry relies heavily on the on-line advertising such as [Google ， Facebook or Alibaba]. Advertising is nothing except information.

GBRT+LR

Practical Lessons from Predicting Clicks on Ads at Facebook or the blog use the GBRT to select proper features and LR to map these features into the interval $[0, 1]$ as a ratio.
Once we have the right features and the right model (decisions trees plus logistic regression), other factors play small roles (though even small improvements are important at scale).

When the feature vector x are given, the tree split the features by GBRT then we transform and input the features to the logistic regression.

Hongliang Jie shares 3 challenges of computational advertising in Etsy,
which will be the titles of the following subsections.

- ONLINE VIDEO ADVERTISING: All you need to know in 2019
- 计算广告
- https://headerbidding.co/category/adops/
- Deep Learning Based Modeling in Computational Advertising: A Winning Formula

Click-Through Rate Modeling

- 聊聊CTR预估的中的深度学习
- Deep Models at DeepCTR
- CTR预估算法之FM, FFM, DeepFM及实践
- Turning Clicks into Purchases
- https://github.com/shenweichen/DeepCTR

- https://github.com/wzhe06/CTRmodel
- https://github.com/cnkuangshi/LightCTR

Conversion Rate Modeling

Bid Optimization

- A collection of research and survey papers of real-time bidding (RTB) based display advertising techniques.

- Papers on Computational Advertising
- CAP 6807: Computational Advertising and Real-Time Data Analytics
- Computational Advertising Contract Preferences for Display Advertising
- Machine Learning for Computational Advertising, UC Santa Cruz, April 22, 2009, Alex Smola, Yahoo Labs, Santa Clara, CA
- Computational Advertising and Recommendation
- Practical Lessons from Predicting Clicks on Ads at Facebook
- http://yelp.github.io/MOE/
- http://www.hongliangjie.com/talks/AICon2018.pdf
- https://sites.google.com/view/tsmo2018/invited-talks

Labs

- https://libraries.io/github/computational-class
- http://www.52caml.com/
- 洪亮劼，博士 – Etsy工程总监
- Data Mining Machine Learning @The University of Texas at Austin
- Center for Big Data Analytics@The University of Texas at Austin
- Multimedia Computing Group@tudelft.nl
- knowledge Lab@Uchicago
- DIGITAL TECHNOLOGY CENTER@UMN
- The Innovation Center for Artificial Intelligence (ICAI)
- Data Mining and Machine Learning lab (DMML)@ASU
- Next Generation Personalization Technologies
- Recommender systems & ranking
- Secure Personalization: Building Trustworthy Recommender Systems
- Similar grants of Next Generation Personalization Technologies