Recommender System

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`.

- [ ] 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://bigdata.ices.utexas.edu/project/large-scale-recommender-systems/
- [ ] https://www.jianshu.com/c/e12d7195a9ff
- [ ] https://www.alansaid.com/publications.html
- [ ] http://www.mymedialite.net/links.html
- [ ] 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
- [ ] http://www.deitel.com/ResourceCenters/Web20/RecommenderSystems/RecommenderSystemAlgorithms/tabid/1317/Default.aspx

Collaborative Filtering

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

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

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

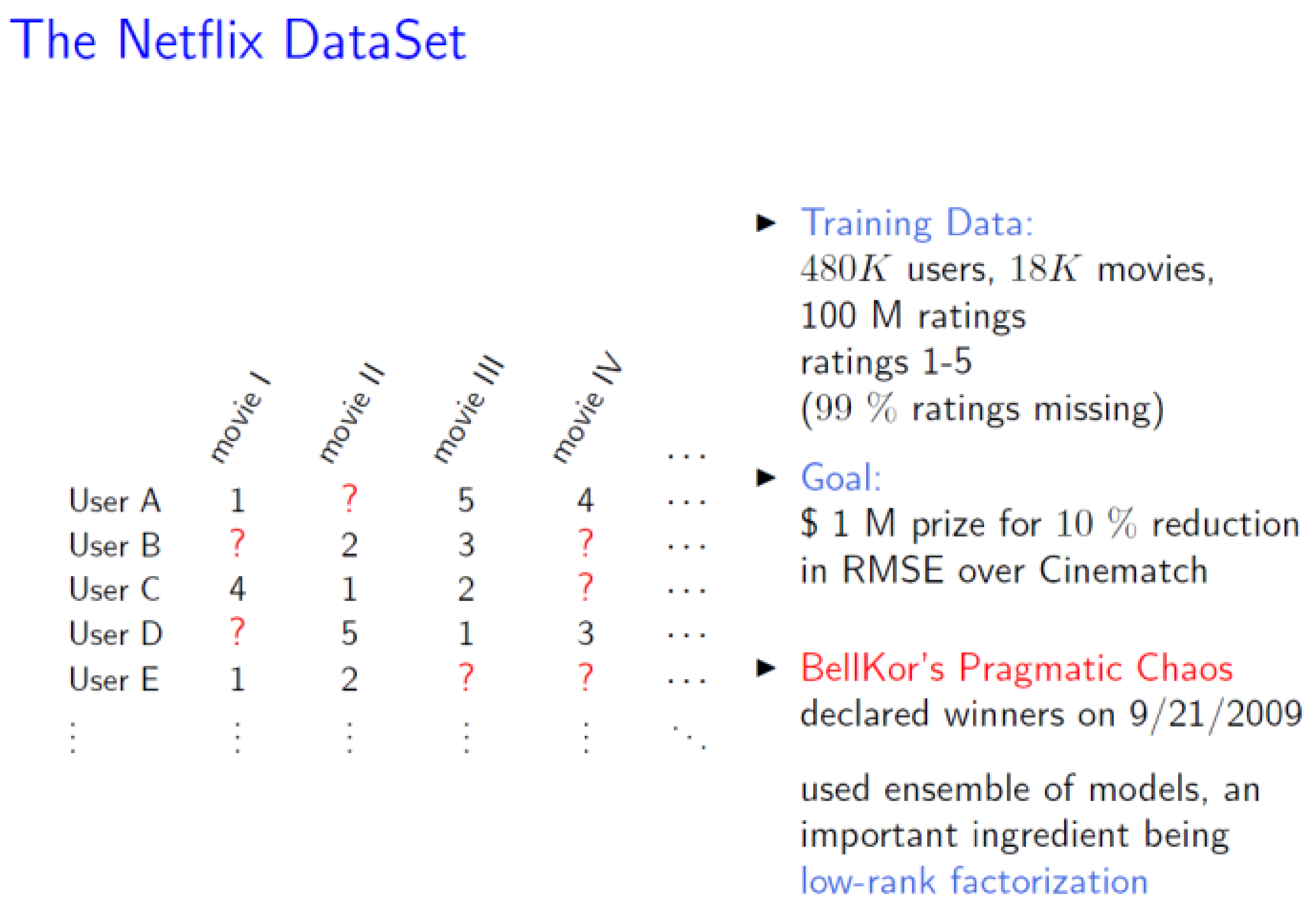
$$L(Z, Y) = \|Z\|_\star - Y(Z_\Omega - X_\Omega).$$

1. Producing Y^{k+1} by

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

2. Producing Z^{k+1} by

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



The Netflix DataSet

	movie I	movie II	movie III	movie IV	...
User A	1	?	5	4	...
User B	?	2	3	?	...
User C	4	1	2	?	...
User D	?	5	1	3	...
User E	1	2	?	?	...
⋮	⋮	⋮	⋮	⋮	⋱

- Training Data:
  $480K$ users, $18K$ movies,
  100 M ratings
  ratings 1-5
  ($99\ \%$ ratings missing)
- Goal:
  \$ 1 M prize for $10\ \%$ reduction
  in RMSE over Cinematch
- BellKor's Pragmatic Chaos
  declared winners on 9/21/2009

  used ensemble of models, an
  important ingredient being
  low-rank factorization

- http://statweb.stanford.edu/~candes/papers/SVT.pdf
- Customized PPA for convex optimization

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

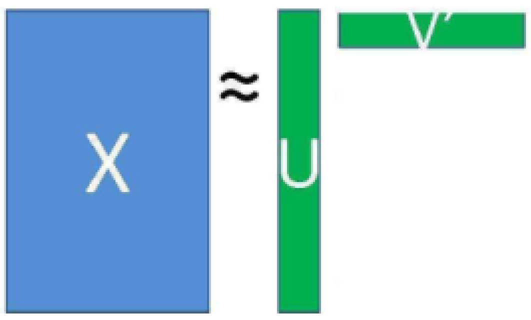
$$\min f_\lambda(Z) = \frac{1}{2}\|P_\Omega(Z-X)\|_F^2 + \lambda\|Z\|_\star$$

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

- https://www.zhihu.com/question/47716840/answer/110843844
- http://www.convexoptimization.com/wikimization/index.php/Matrix_Completion.m
- http://delab.csd.auth.gr/papers/RecSys2016s.pdf
- http://www.princeton.edu/~yc5/ele538b_sparsity/lectures/matrix_recovery.pdf
- https://users.ece.cmu.edu/~yuejiec/ece18898G_notes/ece18898g_nonconvex_lowrank_recovery.pdf

Fix $Rank(Z) = r$ (small) ie $Z = U_{m\times r}V'_{n\times r}$

$$\underset{U,V}{\text{minimize}} \sum_{(i,j):\text{Observed}} ([UV']_{ij} - X_{ij})^2 + \lambda(\|U\|_F^2 + \|V\|_F^2)$$



- Maximum-Margin Matrix Factorization (MMMF), Regularized SVD
- bi-convex criterion, alternating ridge regression... (Non-convex)

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

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

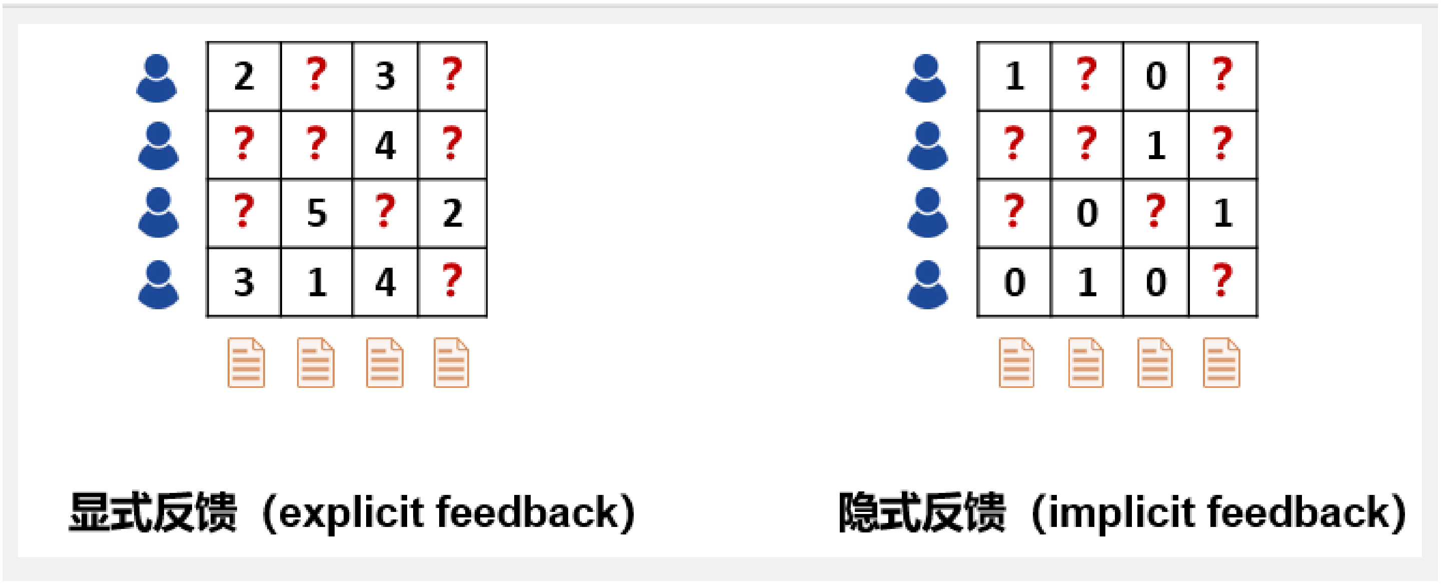
It is called the regularized singular value decomposition.

And the evaluation score is always positive and discrete such as $\{2,4,6,8.10\}$. This technique is also called **nonnegative matrix factorization**.

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,
- https://www.ethanrosenthal.com/2016/10/19/implicit-mf-part-1/
- a curated list in github.com,
- https://zhuanlan.zhihu.com/p/42269534

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
- http://yifanhu.net/PUB/cf.pdf

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:

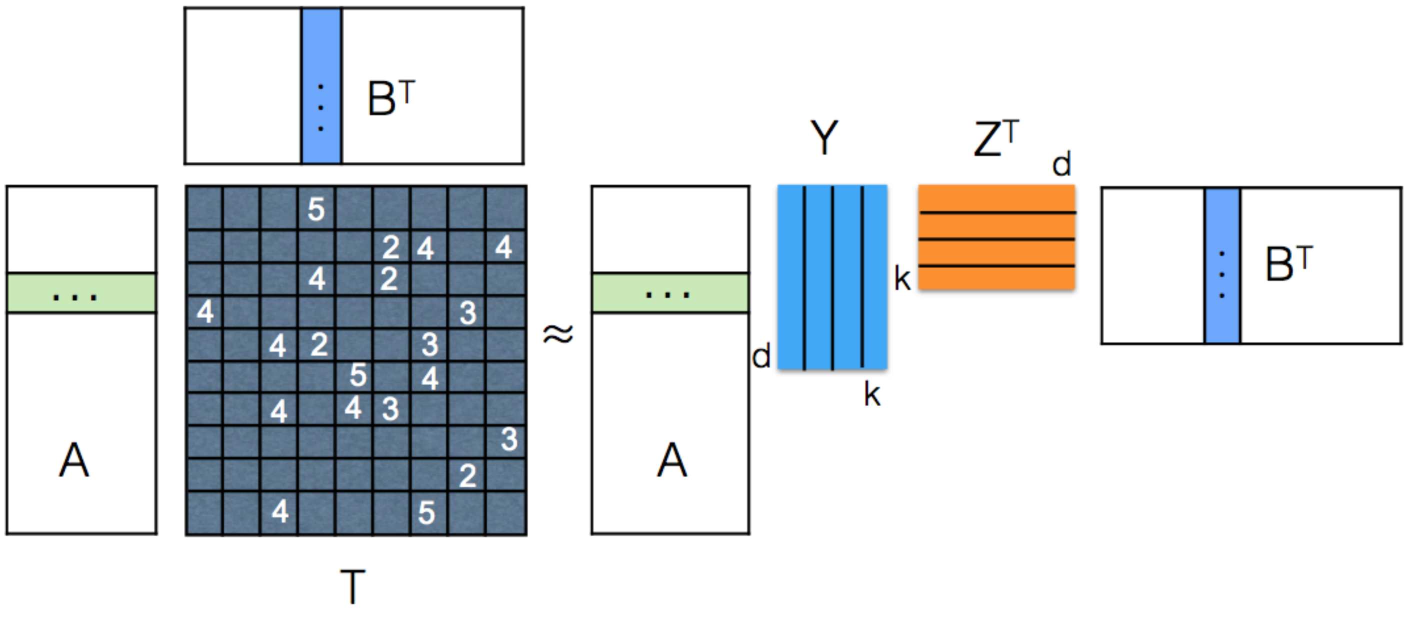
- $\mu + b_u + b_i$ is the base-line prediction;
- $\langle P_u, Q_i \rangle$ is the SVD of rating matrix;
- $\left\langle |N(u)|^{-0.5} \sum_{i \in N(u)} y_i, Q_i \right\rangle$ is the implicit feedback where $N(u)$ is user u's item set, y_j is the implicit feedback of item j.

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

- https://orange3-recommendation.readthedocs.io/en/latest/scripting/rating.html

- https://orange3-recommendation.readthedocs.io/en/latest/widgets/svdplusplus.html
- https://cloud.tencent.com/developer/article/1107364
- https://www.bbsmax.com/A/KE5Q0M9ZJL/

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



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 vector.
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 $\ell(a,b)=\log(1+\exp(-ab))$.

- http://bigdata.ices.utexas.edu/software/inductive-matrix-completion/
- http://www.cs.utexas.edu/users/inderjit/public_papers/imc_bioinformatics14.pdf

Probabilistic Matrix Factorization

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.

- http://www.ideal.ece.utexas.edu/seminar/LatentFactorModels.pdf
- https://web.njit.edu/~zhiwei/CS732/papers/Regression-basedLatentFactorModels_KDD2009.pdf

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

---

Implict Information

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

- http://yifanhu.net/PUB/cf.pdf
- https://github.com/benfred/implicit

- https://www.ethanrosenthal.com/2016/10/19/implicit-mf-part-1/
- https://www.ethanrosenthal.com/2016/11/07/implicit-mf-part-2/
- http://datamusing.info/blog/2015/01/07/implicit-feedback-and-collaborative-filtering/
- https://www.benfrederickson.com/fast-implicit-matrix-factorization/
- https://www.benfrederickson.com/implicit-matrix-factorization-on-the-gpu/

- Matrix Completion/Sensing
- http://statweb.stanford.edu/~candes/papers/MatrixCompletion.pdf
- http://surpriselib.com/
- https://www.cnblogs.com/Xnice/p/4522671.html
- https://blog.csdn.net/turing365/article/details/80544594
- https://en.wikipedia.org/wiki/Collaborative_filtering
- \url{https://www.wikiwand.com/en/Matrix_factorization_(recommender_systems)}
- https://bugra.github.io/work/notes/2014-04-19/alternating-least-squares-method-for-collaborative-filtering/
- https://www.ibm.com/developerworks/cn/web/1103_zhaoct_recommstudy2/index.html
- http://www.ipam.ucla.edu/abstract/?tid=14552&pcode=DLT2018
- http://www.cnblogs.com/DjangoBlog/archive/2014/06/05/3770374.html
- http://perception.csl.illinois.edu/matrix-rank/home.html
- https://www.acemap.info/author/page?AuthorID=7E61F31B

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

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

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
- https://getstream.io/blog/factorization-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://huangzhanpeng.github.io/2018/01/04/Field-aware-Factorization-Machines-for-CTR-Prediction/
- 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.

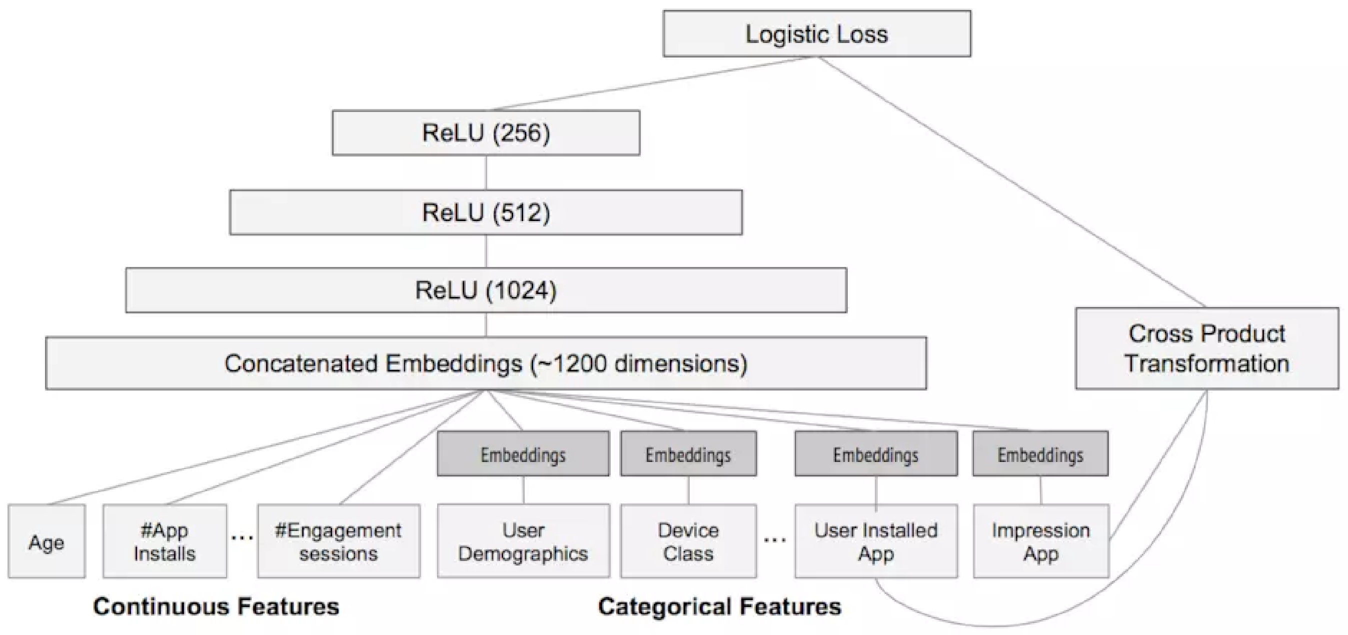
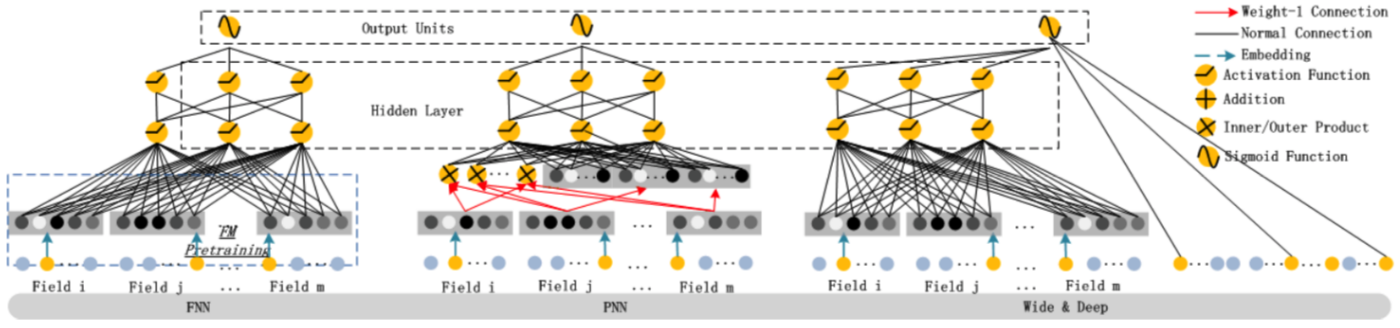
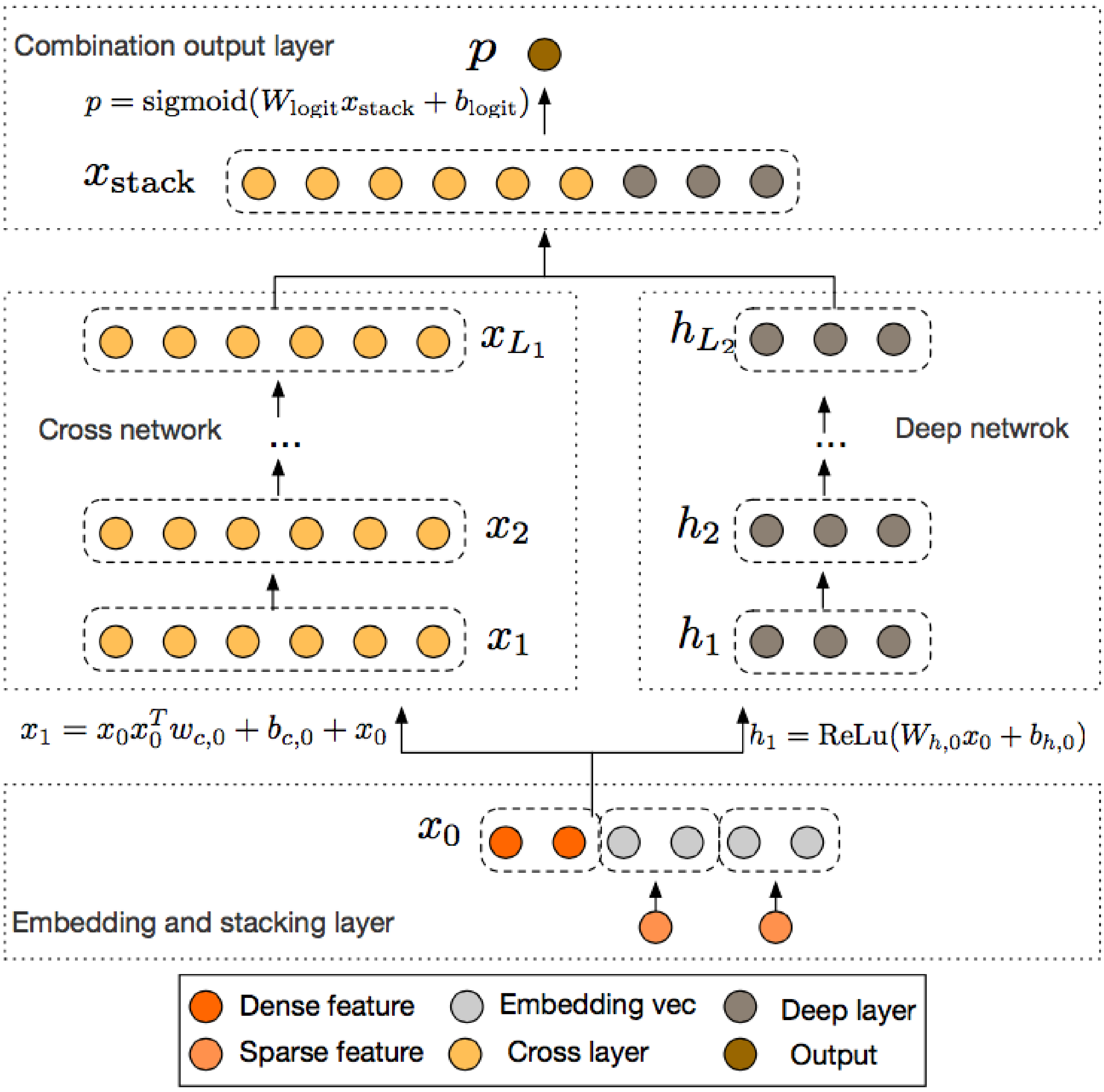


Figure 4: Wide & Deep model structure for apps recommendation.



- https://arxiv.org/pdf/1606.07792.pdf
- https://ai.googleblog.com/2016/06/wide-deep-learning-better-together-with.html
- https://www.jianshu.com/p/dbaf2d9d8c94
- https://www.sohu.com/a/190148302_115128



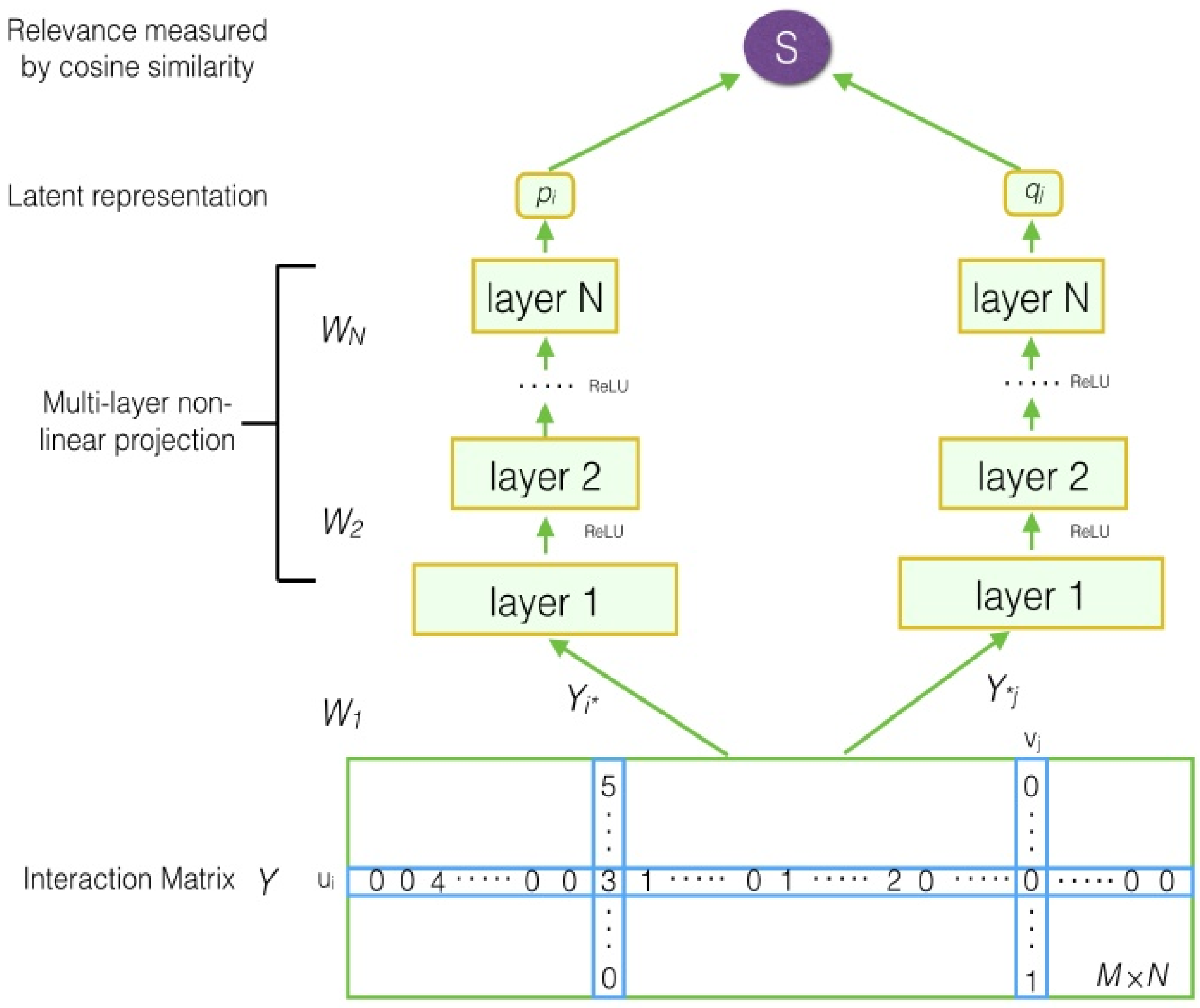
Combination output layer
p
$p = \text{sigmoid}(W_{\text{logit}} x_{\text{stack}} + b_{\text{logit}})$
x_{stack}
x_{L_1}
Cross network
x_2
x_1
h_{L_2}
Deep netwrok
h_2
h_1
$x_1 = x_0 x_0^T w_{c,0} + b_{c,0} + x_0$
$h_1 = \text{ReLu}(W_{h,0} x_0 + b_{h,0})$
x_0
Embedding and stacking layer
Dense feature
Sparse feature
Embedding vec
Cross layer
Deep layer
Output

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 $e^x = 1 + x + \frac{x^2}{2!} + \dots, + \frac{x^n}{n!} + \dots$, which include all the order of interactions.

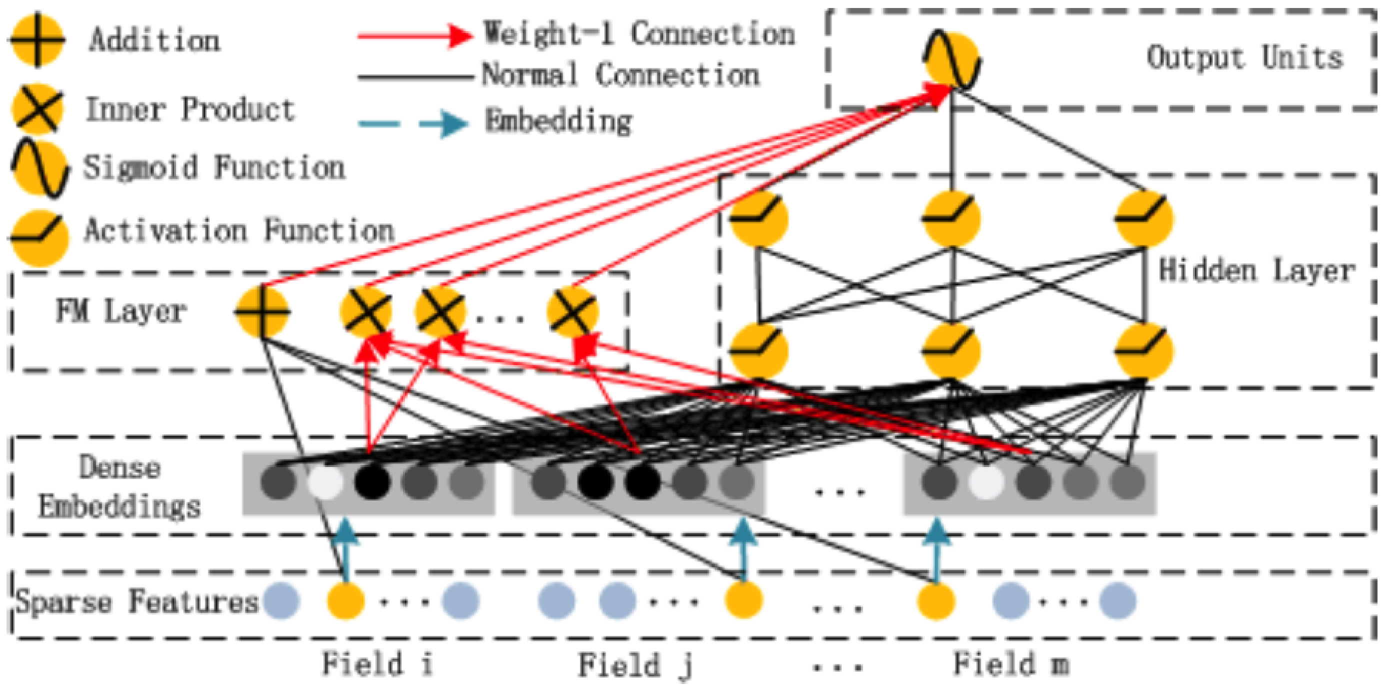
We would like to point out the two interesting features of this network structure:

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.

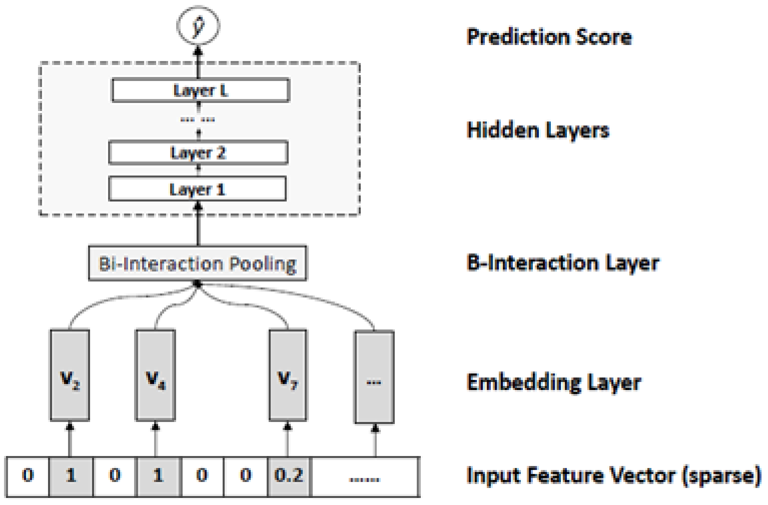


- https://zhuanlan.zhihu.com/p/27999355
- https://zhuanlan.zhihu.com/p/25343518
- https://zhuanlan.zhihu.com/p/32127194
- https://arxiv.org/pdf/1703.04247.pdf
- https://blog.csdn.net/John_xyz/article/details/78933253#deep-fm

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`.

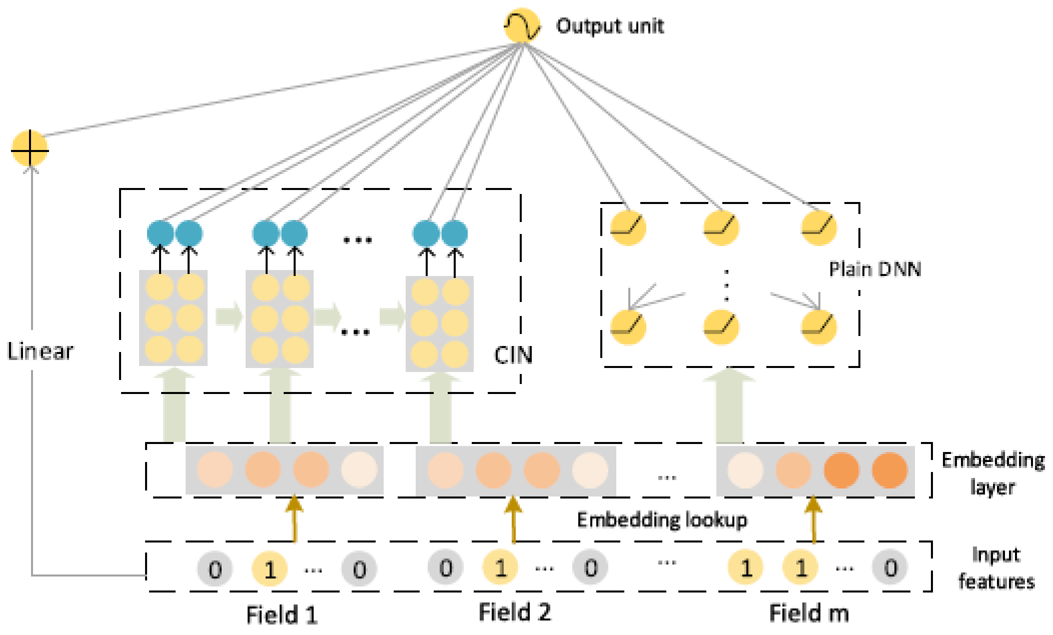


- https://www.comp.nus.edu.sg/~xiangnan/papers/sigir17-nfm.pdf

Attentional Factorization Machines

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

xDeepFM



- [✓] https://www.msra.cn/zh-cn/news/features/kdd-2018-xdeepfm
- [ ] http://kubicode.me/2018/09/17/Deep Learning/eXtreme-Deep-Factorization-Machine/
- [ ] http://d0evi1.com/xdeepfm/
- [ ] https://www.jianshu.com/p/b4128bc79df0

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:

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

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

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:

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

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

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

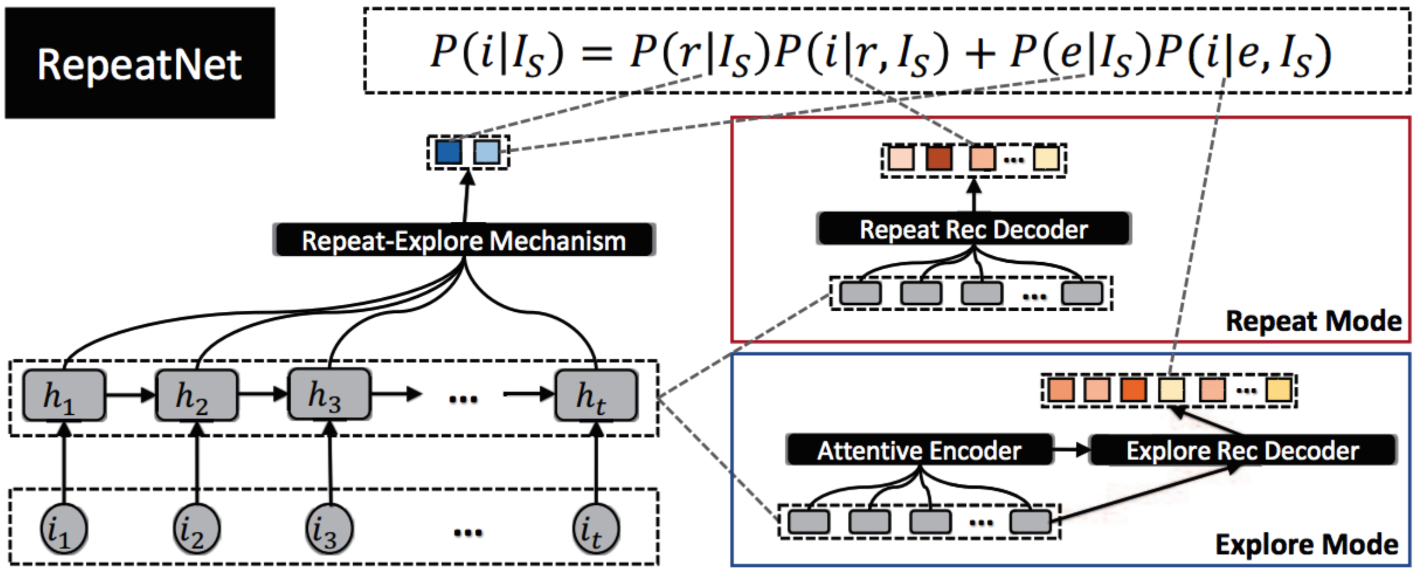
- https://www.cnblogs.com/pinard/p/6530523.html
- https://www.cnblogs.com/kemaswill/p/3269138.html
- https://www.cs.toronto.edu/~rsalakhu/papers/rbmcf.pdf
- http://www.cs.toronto.edu/~fritz/absps/cdmiguel.pdf
- http://deeplearning.net/tutorial/rbm.html

AutoRec

AutoRec is a novel `autoencoder` framework for collaborative filtering (CF).

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

---



- https://github.com/hwwang55/DKN
- https://www.csie.ntu.edu.tw/~b97053/paper/Rendle2010FM.pdf
- http://lipixun.me/2018/02/01/youtube
- https://www.cnblogs.com/pinard/p/6370127.html
- https://www.jianshu.com/p/6f1c2643d31b
- https://blog.csdn.net/John_xyz/article/details/78933253
- http://kubicode.me/2018/02/23/Deep Learning/Deep-in-out-Factorization-Machines-Series/
- https://zhuanlan.zhihu.com/p/38613747
- https://www.infosec-wiki.com/?p=394011
- http://danyangliu.me/深度学习相关推荐模型/
- https://amundtveit.com/2016/11/20/recommender-systems-with-deep-learning/
- http://kubicode.me/2018/10/25/Deep Learning/More-Session-Based-Recommendation/

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.

- http://mirlab.org/conference_papers/International_Conference/ICASSP 2018/pdfs/0006852.pdf
- 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 Matching Models for Recommendation

The recommender system is essential to find the item which matches the user's demand.

- http://sonyis.me/dnn.html
- https://akmenon.github.io/
- https://sigir.org/sigir2018/program/tutorials/
- https://www.comp.nus.edu.sg/~xiangnan/papers/www18-tutorial-deep-matching.pdf
- http://www.hangli-hl.com/uploads/3/4/4/6/34465961/wsdm_2019_workshop.pdf

---

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

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 reach┐ability 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.

Evolution of the Recommender Problem

Evolution of the Recommender Problem
Rating
Ranking
Page Optimization
Context-aware Recommendations

- [ ] 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
- [ ] Deep Learning based Recommender System: A Survey and New Perspectives
- [ ] http://dlrs-workshop.org/
- [ ] 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
- [ ] https://nycdatascience.com/blog/student-works/deep-learning-meets-recommendation-systems/
- [ ] https://www.ethanrosenthal.com/2016/12/05/recasketch-keras/
- [ ] https://tech.meituan.com/2018/06/07/searchads-dnn.html
- [ ] http://benanne.github.io/2014/08/05/spotify-cnns.html
- [ ] https://github.com/grahamjenson/list_of_recommender_systems

Implementation

- [ ] https://github.com/maciejkula/spotlight
- [ ] http://surpriselib.com/
- [ ] 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
- [ ] http://www.mymedialite.net/index.html
- [ ] http://www.mymediaproject.org/

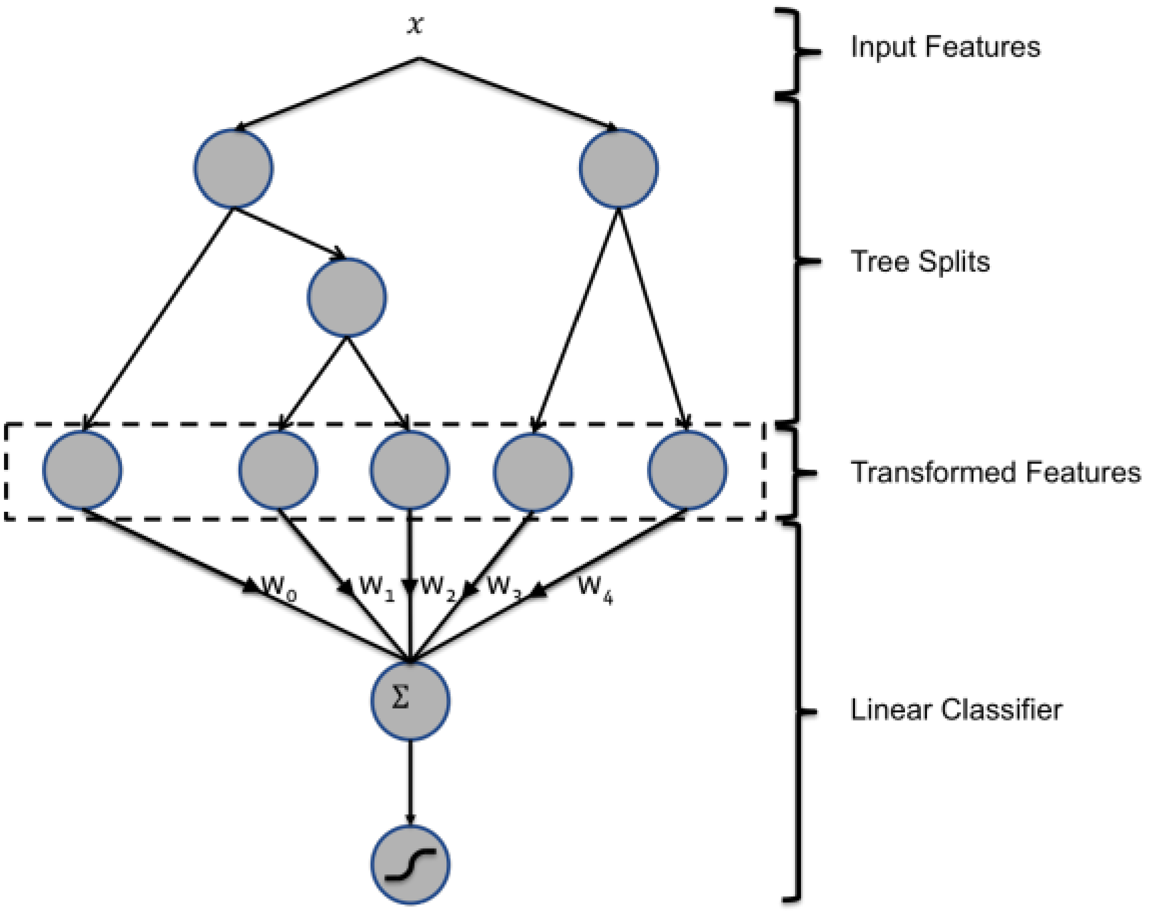
Computational Advertising

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.

- http://kubicode.me/2018/03/19/Deep Learning/Talk-About-CTR-With-Deep-Learning/
- https://github.com/shenweichen/DeepCTR
- https://github.com/wzhe06/Ad-papers

- https://github.com/wnzhang/rtb-papers
- https://github.com/wzhe06/CTRmodel
- https://github.com/cnkuangshi/LightCTR
- http://www.cse.fau.edu/~xqzhu/courses/cap6807.html
- https://www.soe.ucsc.edu/departments/technology-management/research/computational-advertising
- http://alex.smola.org/teaching/ucsc2009/ucsc_1.pdf
- https://deepctr.readthedocs.io/en/latest/models/DeepModels.html
- https://blog.csdn.net/john_xyz/article/details/78933253
- https://people.eecs.berkeley.edu/~jfc/DataMining/SP12/lecs/lec12.pdf
- http://quinonero.net/Publications/predicting-clicks-facebook.pdf
- https://tech.meituan.com/2019/01/17/dianping-search-deeplearning.html
- http://yelp.github.io/MOE/