Recommender systems

Neural recommenders

Lecture 6
2024

Anna Volodkevich

Contents

Matrix factorization as a neural network

MF formulation: Recap

MF associates each user and item with a real-valued vector of latent features. Let $\mathbf{p}_u$ and $\mathbf{q}_i$ denote the latent vector for user u and item i, respectively; MF estimates an interaction y_{ui} as the inner product of $\mathbf{p}_u$ and $\mathbf{q}_i$:

$$\hat{y}_{ui} = f(u, i|\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u^T \mathbf{q}_i = \sum_{k=1}^{K} p_{uk} q_{ik}, \qquad (2)$$

Regression formulation

To learn model parameters, existing pointwise methods [14, 39] largely perform a regression with squared loss:

$$L_{sqr} = \sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} w_{ui}(y_{ui} - \hat{y}_{ui})^2, \qquad (5)$$

where $\mathcal{Y}$ denotes the set of observed interactions in $\mathbf{Y}$, and $\mathcal{Y}^-$ denotes the set of negative instances, which can be all (or sampled from) unobserved interactions; and w_{ui} is a hyper-parameter denoting the weight of training instance (u, i).

Classification formulation

$$p(\mathcal{Y}, \mathcal{Y}^- | \mathbf{P}, \mathbf{Q}, \Theta_f) = \prod_{(u,i)\in\mathcal{Y}} \hat{y}_{ui} \prod_{(u,j)\in\mathcal{Y}^-} (1 - \hat{y}_{uj}). \qquad (6)$$

Taking the negative logarithm of the likelihood, we reach

$$\begin{aligned} L &= -\sum_{(u,i)\in\mathcal{Y}} \log \hat{y}_{ui} - \sum_{(u,j)\in\mathcal{Y}^-} \log(1 - \hat{y}_{uj}) \\ &= -\sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log(1 - \hat{y}_{ui}). \end{aligned} \qquad (7)$$

Negative sampling

- **Uniform negative sampling**. For each training example (consisting of a user and a positive item), we sample *m* negative (unobserved) items, uniformly at random. May be slow to converge ("easy" negatives).
- **Popularity-based negative sampling**. For each training example (consisting of a user and a positive item), we sample m negative (unobserved) items, proportionally to their popularity in dataset.
- **"Hard" negative sampling**. Sample items using previous model, giving the maximal gradient update by being similar to the user's positives or giving the highest score.
- **Rule-based negatives**. E.g. negatives from the same category.
- **In-batch negatives**. The other users positives are the user negatives.

More on negative sampling

- *Chen, Chong, et al. "Revisiting negative sampling vs. non-sampling in implicit recommendation." ACM Transactions on Information Systems 41.1 (2023): 1-25.*
- *Pellegrini, Roberto, Wenjie Zhao, and Iain Murray. "Don't recommend the obvious: estimate probability ratios." Proceedings of the 16th ACM Conference on Recommender Systems. 2022.*
- *Wang, HaiYing, Aonan Zhang, and Chong Wang. "Nonuniform negative sampling and log odds correction with rare events data." Advances in Neural Information Processing Systems 34 (2021): 19847-19859.*

Matrix factorization as a neural network

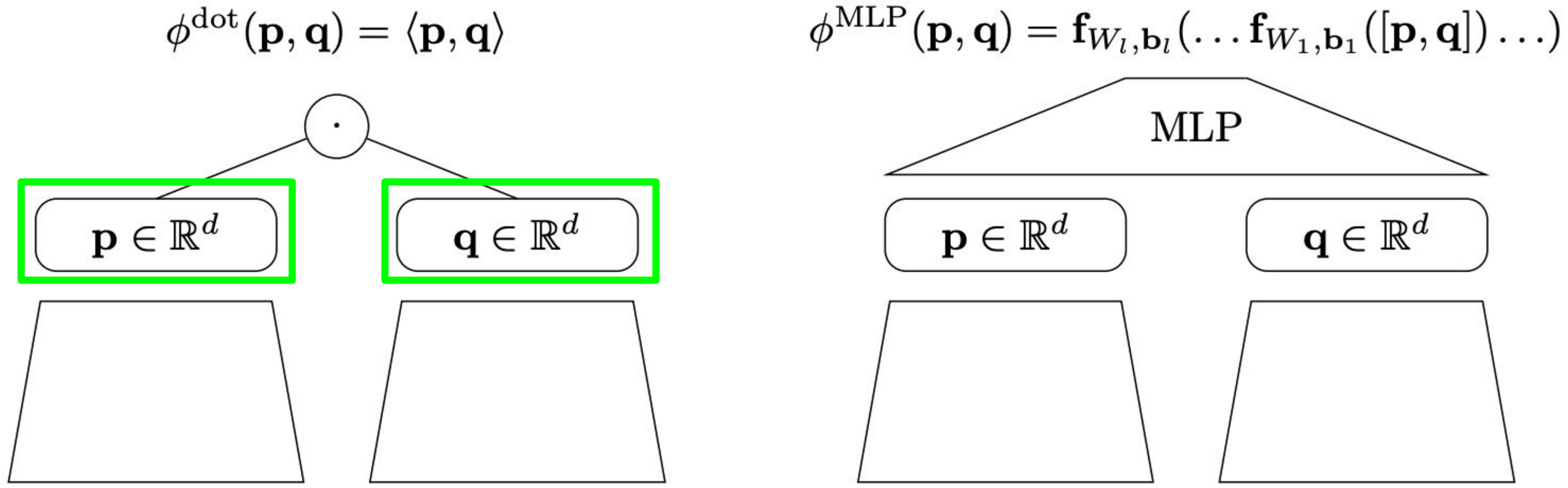

$\phi^{\text{dot}}(\mathbf{p},\mathbf{q}) = \langle \mathbf{p},\mathbf{q} \rangle$
$\cdot$
$\mathbf{p} \in \mathbb{R}^d$
$\mathbf{q} \in \mathbb{R}^d$
$\phi^{\text{MLP}}(\mathbf{p},\mathbf{q}) = \mathbf{f}_{W_l,\mathbf{b}_l}(\ldots \mathbf{f}_{W_1,\mathbf{b}_1}([\mathbf{p},\mathbf{q}])\ldots)$
MLP
$\mathbf{p} \in \mathbb{R}^d$
$\mathbf{q} \in \mathbb{R}^d$

Figure 1: A model with dot product similarity (left) and MLP-based learned similarity (right).

[Rendle, Steffen, et al. "Neural collaborative filtering vs. matrix factorization revisited." RecSys'20]

Matrix factorization as a neural network: NeuMF (MLP+GMF)

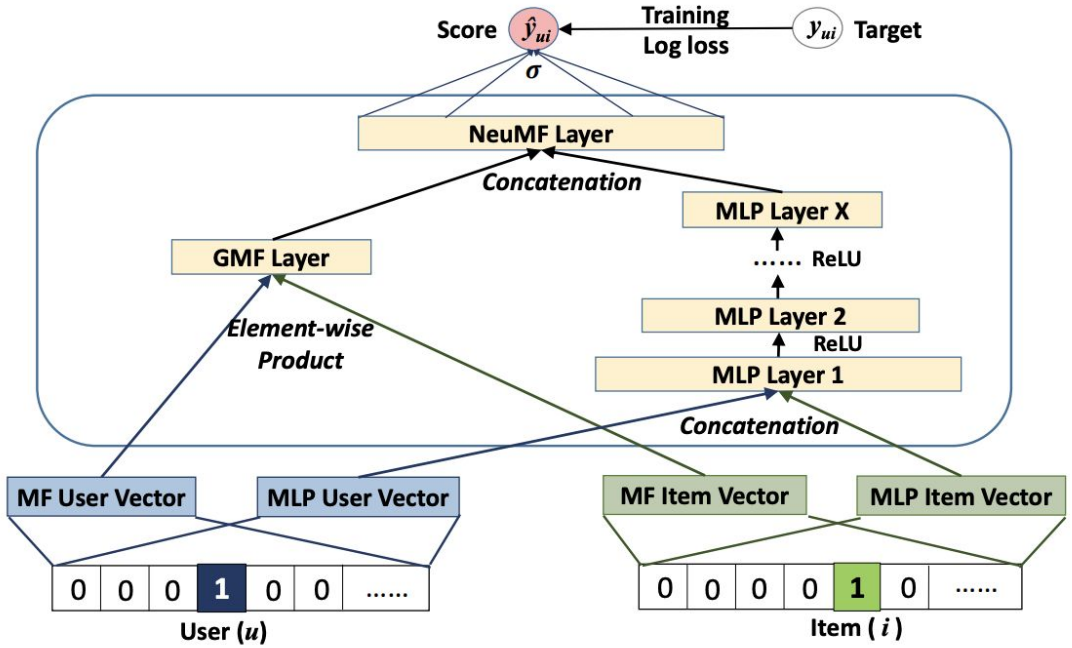


Figure 3: Neural matrix factorization model

Matrix factorization as a neural network

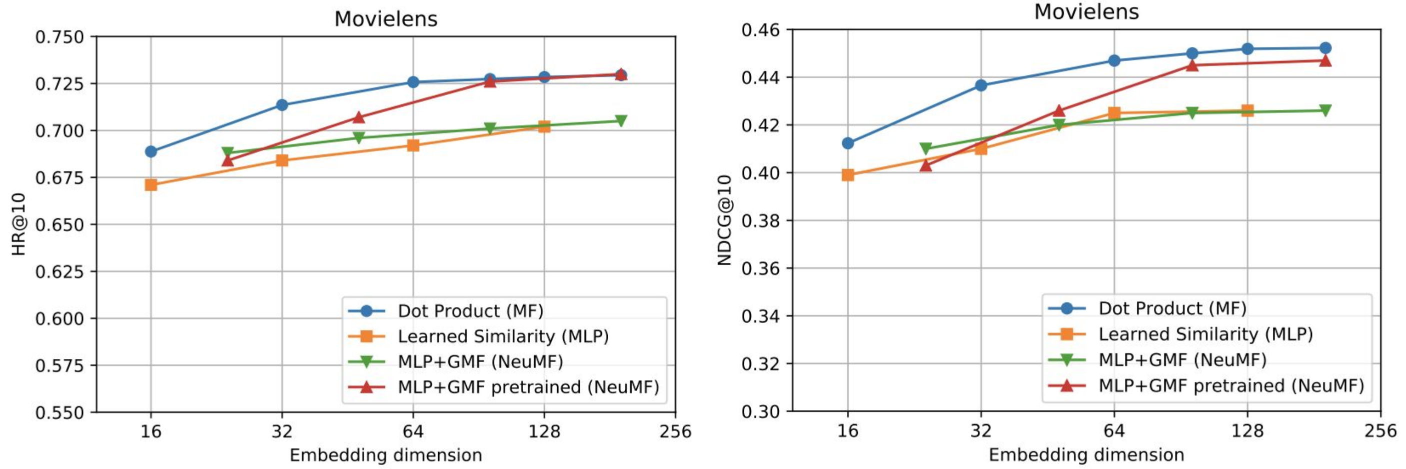
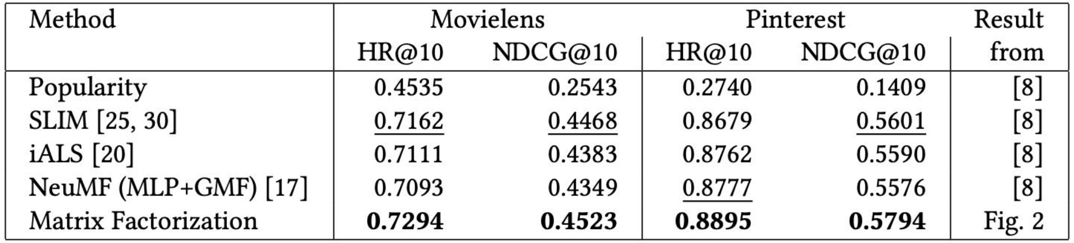


Figure 2: Comparison of learned similarities (MLP, NeuMF) to a dot product: The results for MLP and NeuMF are from [17]. The dot product substantially outperforms the learned similarity measures. Only the pretrained NeuMF is competitive, on one dataset, and for large embedding dimension.



Method	Movielens		Pinterest		Result
	HR@10	NDCG@10	HR@10	NDCG@10	from
Popularity	0.4535	0.2543	0.2740	0.1409	[8]
SLIM [25, 30]	0.7162	0.4468	0.8679	0.5601	[8]
iALS [20]	0.7111	0.4383	0.8762	0.5590	[8]
NeuMF (MLP+GMF) [17]	0.7093	0.4349	0.8777	0.5576	[8]
Matrix Factorization	**0.7294**	**0.4523**	**0.8895**	**0.5794**	Fig. 2

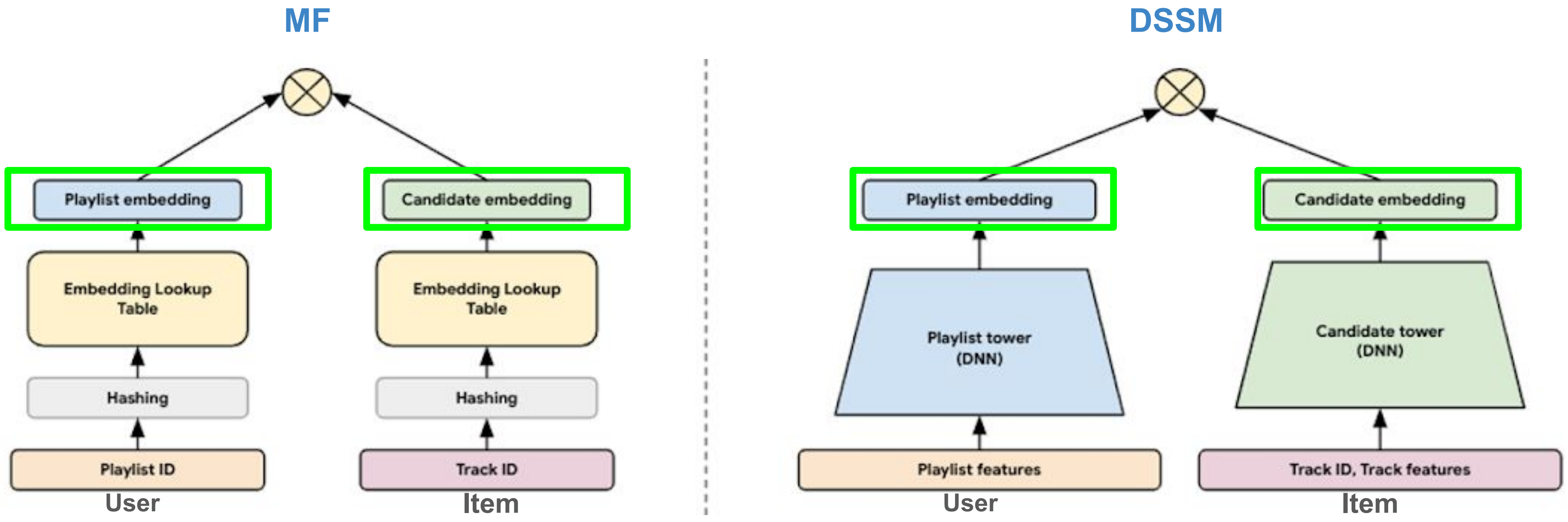
- a simple dot product substantially outperforms the proposed learned similarities
- a MLP can in theory approximate any function, we show that it is non-trivial to learn a dot product with an MLP
- feature interactions should be directy modelled

Rendle, Steffen, et al. "Neural collaborative filtering vs. matrix factorization revisited." RecSys'20

Matrix factorization as a neural network

- Advantages
  - commonly-used frameworks and optimization algorithms
  - easy incremental updates
  - gives user and item vectors for fast inference
- Disadvantages
  - no user/item features
  - no context features
  - no user sequence / latest interests representation

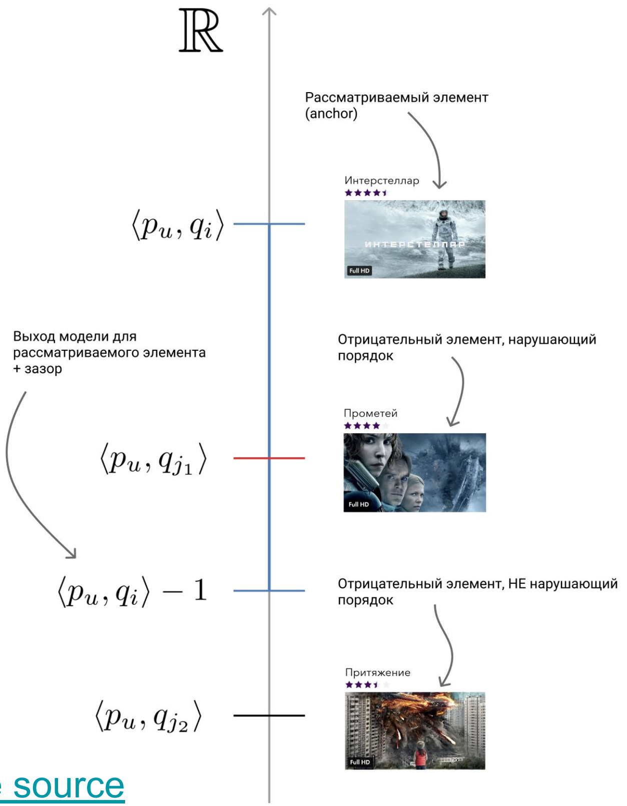
Incorporating features: Two-tower/DSSM



Huang, Po-Sen, et al. "Learning deep structured semantic models for web search using clickthrough data." Proceedings of the 22nd ACM international conference on Information & Knowledge Management. 2013.

image source: Scaling deep retrieval with TensorFlow Recommenders and Vertex AI Matching Engine

Recap: ranking loss functions



$\mathbb{R}$
Рассматриваемый элемент (anchor)
Интерстеллар
$\langle p_u, q_i \rangle$
Выход модели для рассматриваемого элемента + зазор
Отрицательный элемент, нарушающий порядок
Прометей
$\langle p_u, q_{j_1} \rangle$
$\langle p_u, q_i \rangle - 1$
Отрицательный элемент, НЕ нарушающий порядок
Притяжение
$\langle p_u, q_{j_2} \rangle$

BPR loss

score for positive item *score for negative item*

$$\hat{x}_{uij} := \hat{x}_{ui} - \hat{x}_{uj} \qquad \sigma(x) := \frac{1}{1+e^{-x}}$$

$$\boxed{\text{BPR-OPT} := \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2 \quad \to \textit{max}}$$

model params

Margin loss / WARP loss

score for positive item *score for negative item*

$$\boxed{L\left(\left\lfloor \frac{Y-1}{N} \right\rfloor\right) |1 - f_y(x_i) + f_{\bar{y}}(x_i)|_+ \quad \to \textit{min}}$$

$$L(k) = log(k)$$

N - number of negatives sampled
Y - number of items

image source

BPR paper

WARP paper

Two-tower/DSSM: feature generation

- Categorical features
  - embeddings
- Text
  - pre-trained embeddings combination
  - sentence embeddings
- Numerical
  - squashing (sigmid over scaled feature values with learned scale and bias)
  - discretization (binning)
  - piecewise linear encoding
- Sequences
  - Long/short term user history as an additional input (user as a combination of item embeddings

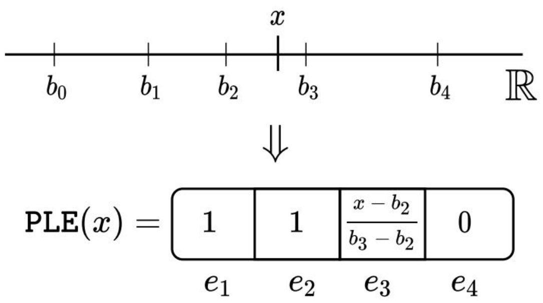


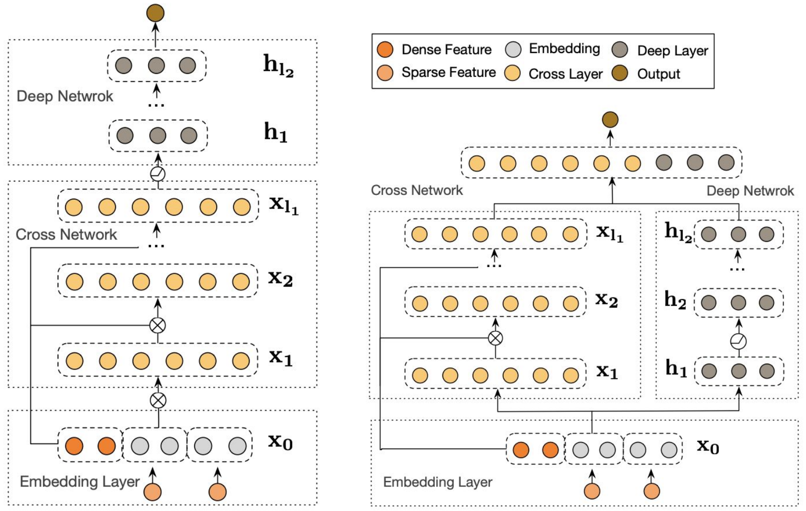
Figure 1: The piecewise linear encoding (PLE) in action for $T = 4$ (see Equation 1).

Gorishniy, Yury, Ivan Rubachev, and Artem Babenko. "On embeddings for numerical features in tabular deep learning." Advances in Neural Information Processing Systems 35 (2022): 24991-25004.

Two-tower / DSSM

- Advantages
  - commonly-used frameworks and optimization algorithms
  - easy incremental updates
  - gives user and item vectors for fast inference
  - utilize user/item features
- Disadvantages
  - no user sequence / latest interests representation in original architecture
  - no feature cross (like in Factorization Machines)
- Useful links
  - MTS Your second recsys: video, code

Adding feature cross (DCN-V2)



(a) Stacked (b) Parallel

Figure 1: Visualization of DCN-V2. ⊗ represents the cross operation in Eq. (1), *i.e.*, $x_{l+1} = x_0 \odot (W_l x_l + b_l) + x_l$.

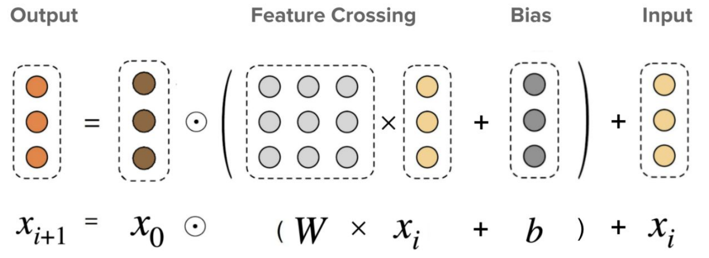


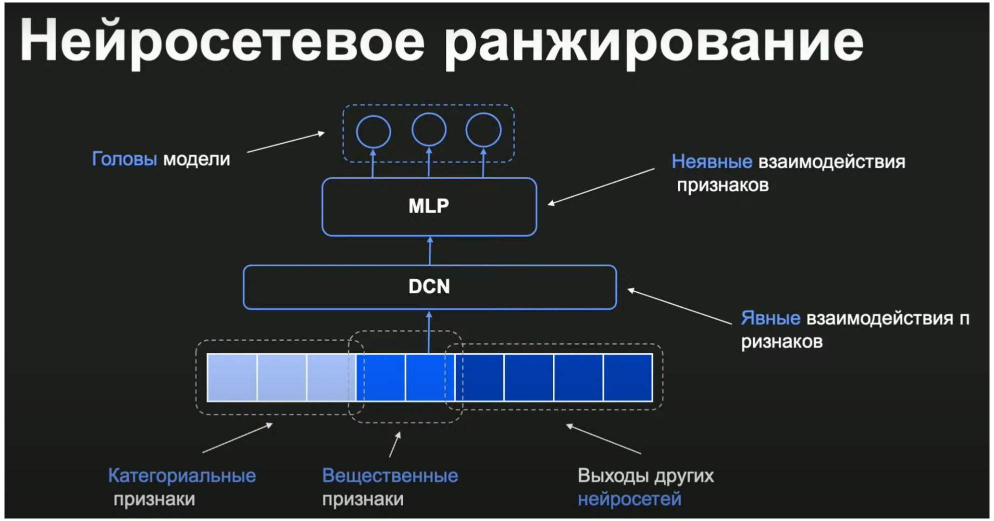
Figure 2: Visualization of a cross layer.

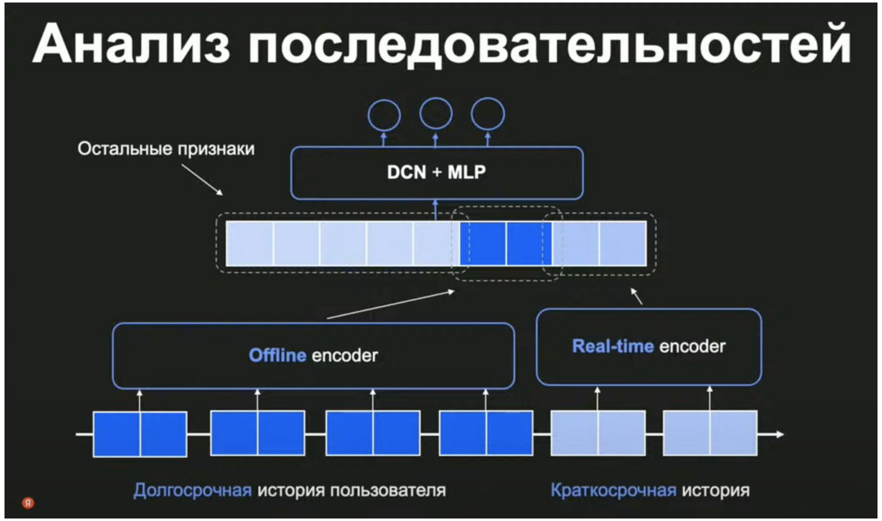
$$x_{l+1} = x_0 \odot (W_l x_l + b_l) + x_l \qquad (1)$$

where $x_0 \in \mathbb{R}^d$ is the base layer that contains the original features of order 1, and is normally set as the embedding (input) layer. $x_l, x_{l+1} \in \mathbb{R}^d$, respectively, represents the input and output of the $(l+1)$-th cross layer. $W_l \in \mathbb{R}^{d \times d}$ and $b_l \in \mathbb{R}^d$ are the learned weight matrix and bias vector. Figure 2 shows how an individual cross layer functions.

Wang, Ruoxi, et al. "Deep & cross network for ad click predictions." Proceedings of the ADKDD'17. 2017. 1-7.
Wang, Ruoxi, et al. "Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems." Proceedings of the web conference 2021. 2021.
TensorFlow Recommenders tutorial

Neural ranking by Yandex

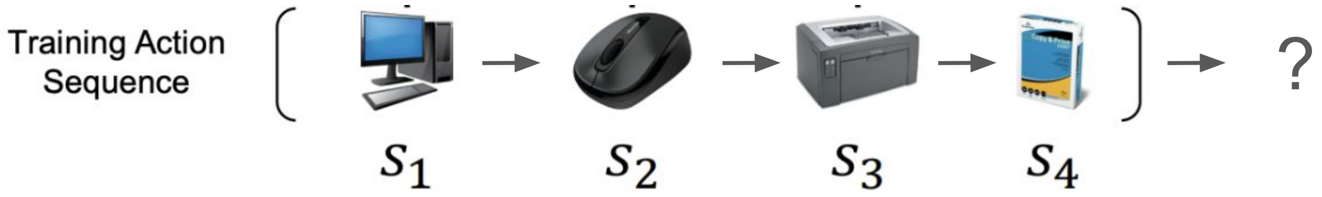

Нейросетевое ранжирование
Головы модели
Неявные взаимодействия признаков
MLP
DCN
Явные взаимодействия п ризнаков
Категориальные признаки
Вещественные признаки
Выходы других нейросетей



- Neural ranking works for industrial tasks with large data volumes
- Allows to use more data and feature sources and gives bigger `model capacity`
- Suits for multiple feedback types out of the box

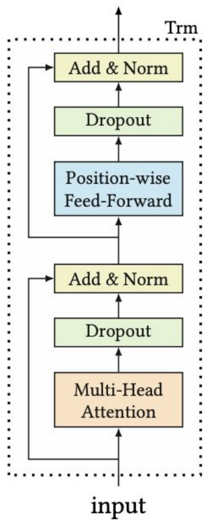
Slide source and highly recommended talk by Kirill Khrylchenko: Yandex ML Party

Sequential recommendations: task

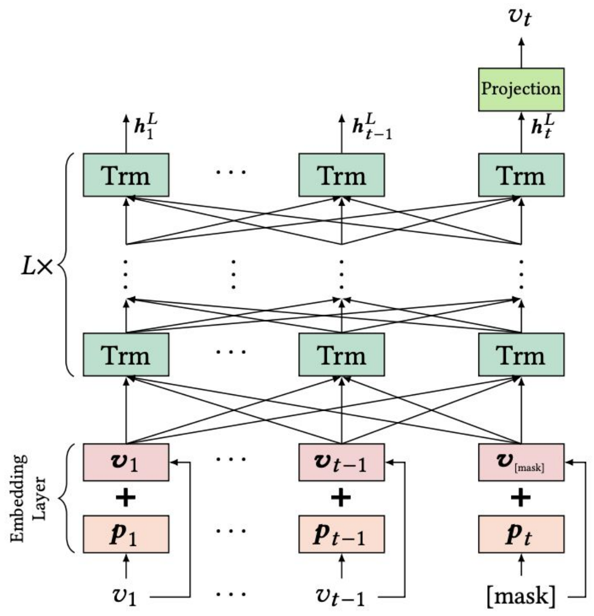


- items are treated as tokens
- users are treated as item sequences
- item embeddings are learned from user sequences with masked language modeling / next item prediction / other task
- can capture temporal dynamics and user preference drift
- suits for online recommendations
- rapidly developing research area
- commonly-used models
  - Caser (Convolutional)
  - GRU4Rec
  - BERT4Rec
  - SASRec

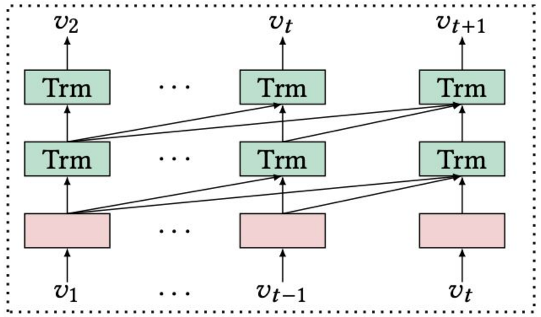
Sequential recommendations: examples



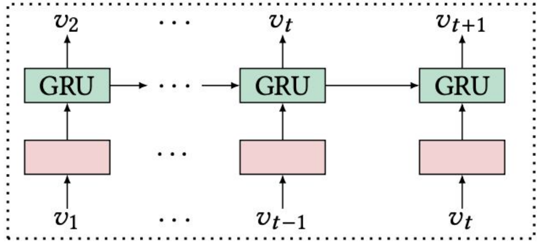
(a) Transformer Layer.



(b) BERT4Rec model architecture.



(c) SASRec model architecture.



(d) RNN based sequential recommendation methods.

image source: Sun, Fei, et al. "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer." *Proceedings of the 28th ACM international conference on information and knowledge management*. 2019.

Sequential modelling: SASRec

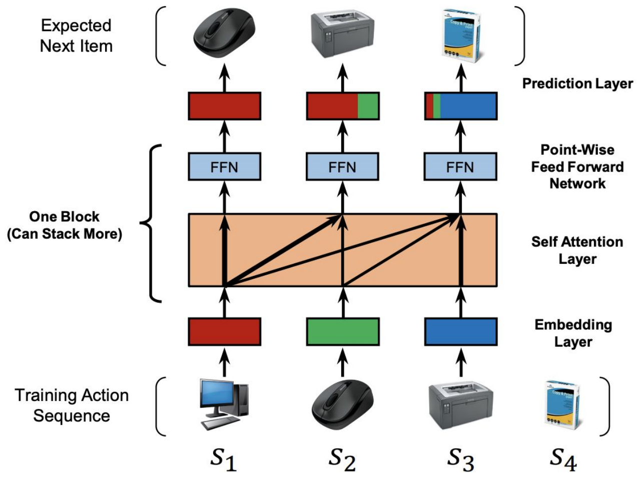


Figure 1: A simplified diagram showing the training process of SASRec. At each time step, the model considers all previous items, and uses attention to 'focus on' items relevant to the next action.

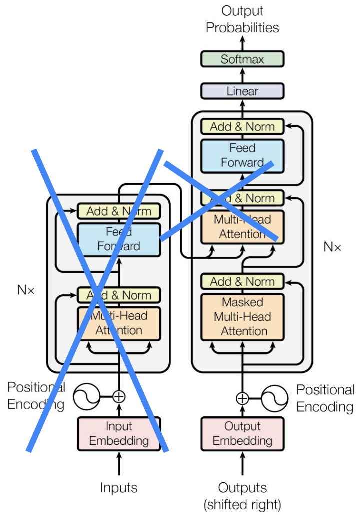
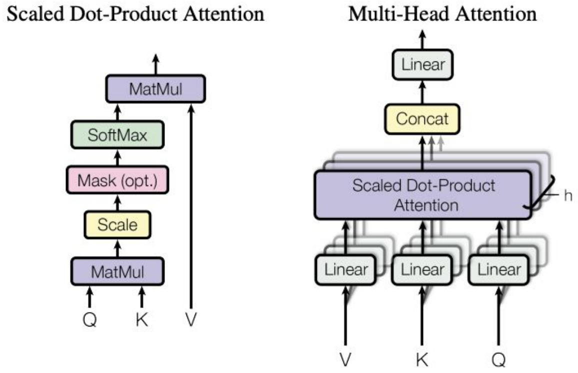


Figure 1: The Transformer - model architecture.



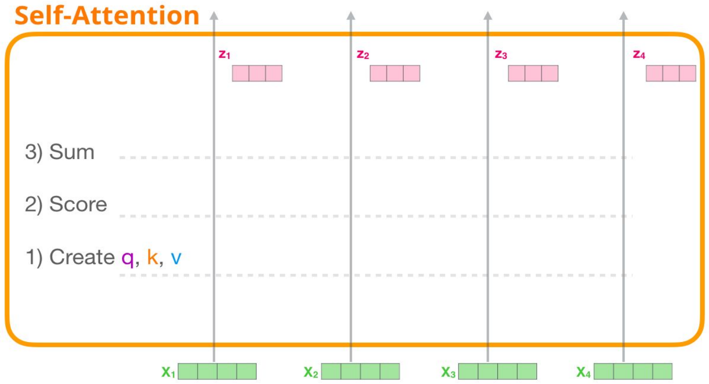
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

SASRec is a decoder-only model pretty close to language models like GPT-2

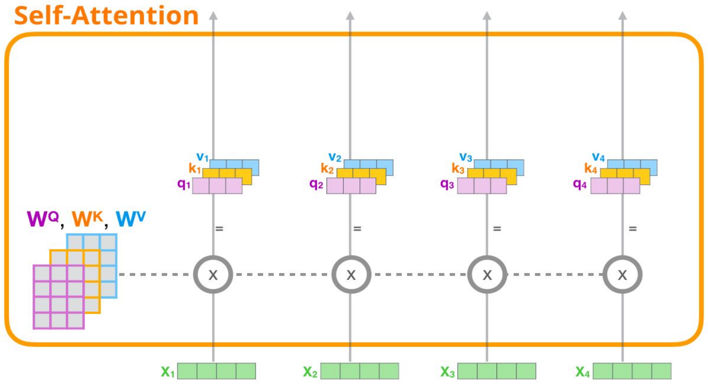
- Learned positional embeddings
- Layer normalization
- Small in comparison with LMs (two blocks, two heads, latent dimensions up to 50)

Kang, Wang-Cheng, and Julian McAuley. "Self-attentive sequential recommendation." IEEE, 2018.
Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

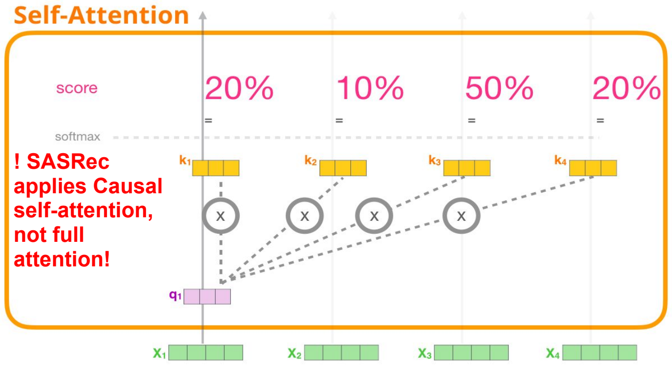
Selt-Attention: Recap



1) For each input token, create a query vector, a key vector, and a value vector by multiplying by weight Matrices W^Q, W^K, W^V



2) Multiply (dot product) the current query vector, by all the key vectors, to get a score of how well they match



! SASRec applies Causal self-attention, not full attention!

3) Multiply the value vectors by the scores, then sum up

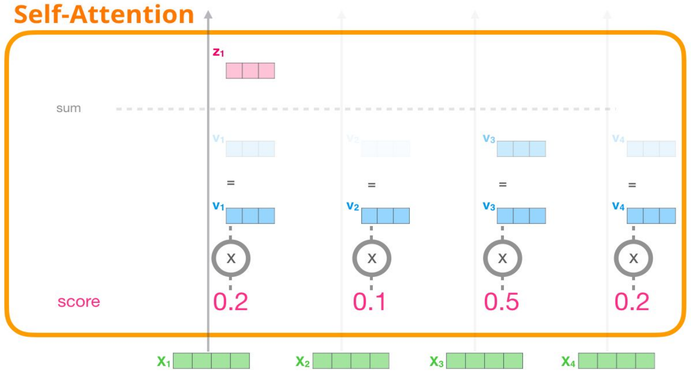


image source: The Illustrated GPT-2 (Visualizing Transformer Language Models)

Sequential modelling: Loss functions

Original SASRec loss: binary cross entropy with one negative sample for each positive

$$\mathcal{L}_{BCE} = -\sum_{u \in U} \sum_{t=1}^{n_u} \log(\sigma(r_{t,i_t}^{(u)})) + \log(1 - \sigma(r_{t,-}^{(u)})),$$

BERT4Rec loss: full cross entropy

$$\mathcal{L}_{CE} = -\sum_{u \in U} \sum_{t \in T_u} \log \frac{\exp(r_{t,i_t}^{(u)})}{\sum_{i \in I} \exp(r_{t,i}^{(u)})}$$

Sampled cross-entropy from “Turning Dross Into Gold Loss: is BERT4Rec really better than SASRec?”

$$\mathcal{L}_{CE-sampled_N} = -\sum_{u \in U} \sum_{t=1}^{n_u} \log \frac{\exp(r_{t,i_t}^{(u)})}{\exp(r_{t,i_t}^{(u)}) + \sum_{i \in I_N^{-(u)}} \exp(r_{t,i}^{(u)})},$$

Klenitskiy, Anton, and Alexey Vasilev. "Turning Dross Into Gold Loss: is BERT4Rec really better than SASRec?." Proceedings of the 17th ACM Conference on Recommender Systems. 2023.

Sequential modelling: Open questions and research directions

- Application for different recommendation tasks, e.g. for top-k prediction task
- Quality improvement: positional encoding, negative sampling, negative feedback/different feedback types incorporation, hyperbolic embeddings
- Computational efficiency: embeddings quantization, efficient negative sampling

Petrov, Aleksandr V., and Craig Macdonald. "RecJPQ: Training Large-Catalogue Sequential Recommenders." Proceedings of the 17th ACM International Conference on Web Search and Data Mining. 2024.

Sequential models

- Advantages
  - commonly-used frameworks and optimization algorithms
  - easy incremental updates
  - sequential patterns learning
- Disadvantages
  - no features out-of-the box
  - no feature cross
  - no user model
  - computational limitations
- Useful links
  - The Illustrated GPT-2 (Visualizing Transformer Language Models)
  - [RU] Интенсив GPT Week Школы анализа данных 2023