16. Recommender Systems

Junggwon Shin, shinjunggwon@gmail.com

Summary for Dive Into Deep Learning, https://d2l.ai/chapter_preface/index.html

16.5. Personalized Ranking for Recommender Systems

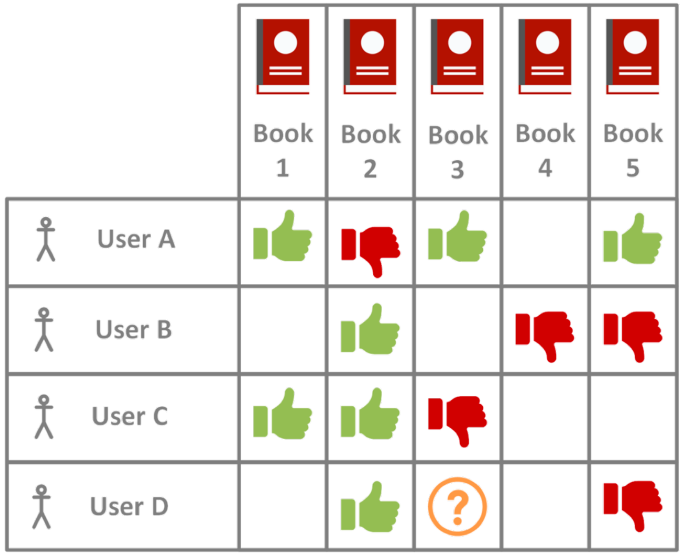
16.6. Neural Collaborative Filtering for Personalized Ranking (NeuMF)

16.7 Sequence-Aware Recommender Systems

16.5. Personalized Ranking for Recommender Systems

Limitation of feedback-based Recommendation algorithms

- Incomplete and biased labels
    - Only explicit user feedback was considered
    - Most feedback in real-world scenarios are is implicit (i.e. click, add to cart)
    - Non-observed user-item pairs (missing values)
- Traditional approach
    - Ignoring the non-observed pairs for model training

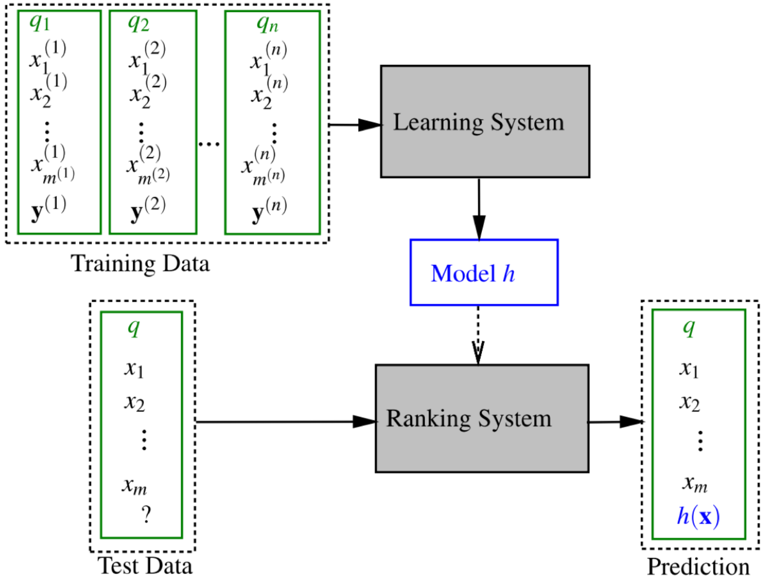


Generative models for recommendations

- Generating ranked item lists from implicit feedback

Approaches for ranking model

- Pointwise: calculate ranking score or relevance class (0 or 1) for a item and a user
    - Matrix Factorization and AutoRec
- Pairwise: predicting more relevant one between two items for a given user
- Listwise: ordering entire list of items for a given user
    - evaluation mettric: nDCG



Bayesian Personalized Ranking Loss and its Implementation

- A **pairwise** personalized ranking the maximum posterior estimator
- Training data: list of tuples of a user, positive item and negative item
  - positive item: item with user feedback
  - negative item: item **without** user feedback
- Label: positive item > negative item

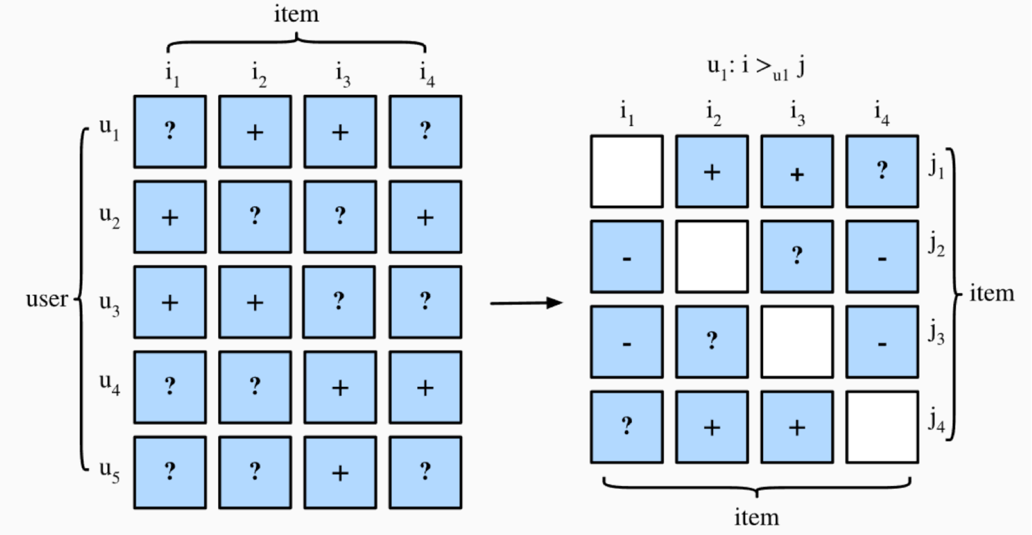
Training data tuple (u, i, j)

- u: user
- i: positive item
- j: negative item

Model

$$\begin{aligned}
&\mathrm{p}(\theta | i > j) \\
&\quad \propto \mathrm{p}(i > j|\theta)\mathrm{p}(\theta) \\
&\quad \propto \ln \mathrm{p}(i > j|\theta)\mathrm{p}(\theta) \\
&\quad = \ln \prod_{(u,i,j)\in D} \sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right)\mathrm{p}(\theta) \\
&\quad = \sum_{(u,i,j)\in D} \ln \sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right) + \ln \mathrm{p}\,(\theta) \\
&\quad = \sum_{(u,i,j)\in D} \ln \sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right) + \lambda \|\theta\|^2
\end{aligned}$$

- θ: model parameter
- $\hat{y}_{ui}$: predicted score for positive item i for given user u
- $\hat{y}_{uj}$: predicted score for negative item j for given user u
- $\|\theta\|^2$: L2 Regularization
  - Gaussian prior with zero-mean and covariance matrix of λI

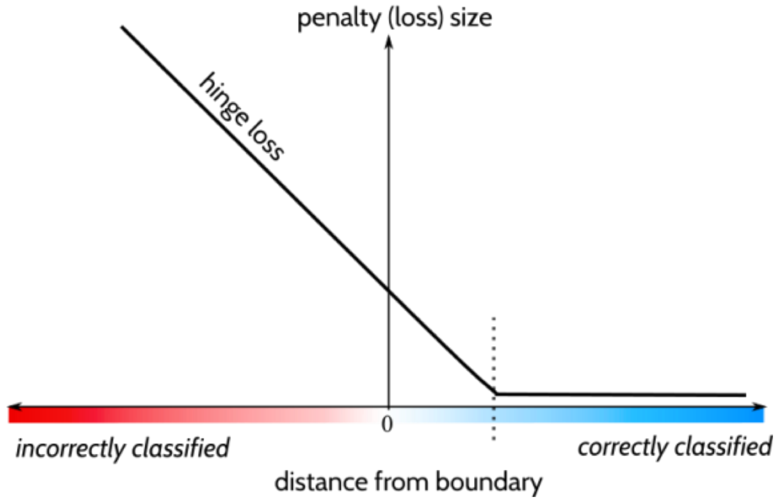


Hinge Loss

- A **pairwise** objective function for a given pair in classification tasks
- push negative items away from positive items

$$\sum_{(u,i,j\in D)} \max(m - \hat{y}_{ui} + \hat{y}_{uj}, 0)$$

- m: margin size
- $\hat{y}_{ui}$: predicted score for p item i for given user u
- $\hat{y}_{uj}$: predicted score for item j for given user u



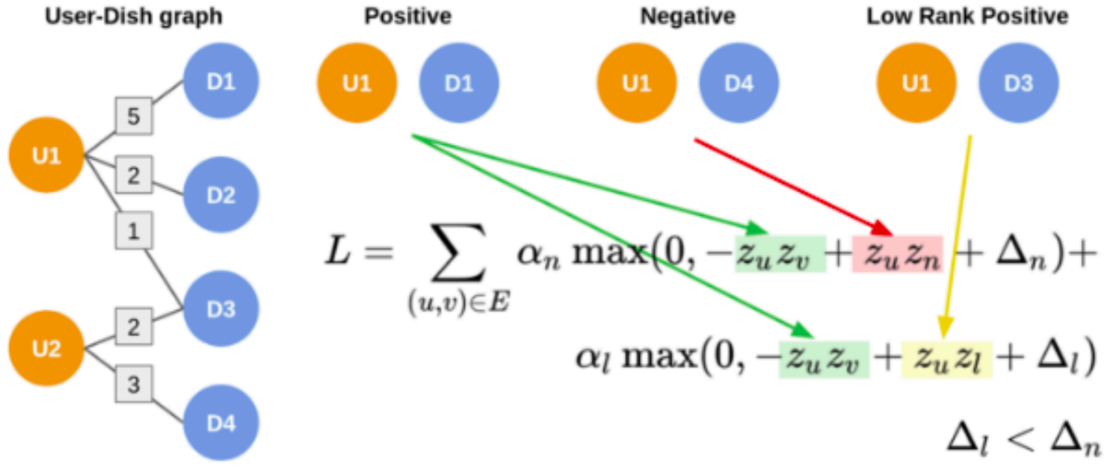


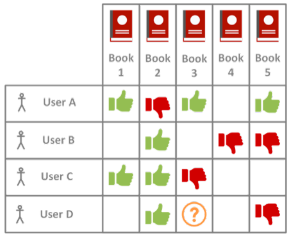
Figure 4: Our Uber Eats recommendation system leverages max-margin loss augmented with low rank positives.

https://eng.uber.com/uber-eats-graph-learning/

Neural matrix factorization

- Binary output (relevant or not)
- Leverage the flexibility and non-linearity of neural networks to replace dot products of matrix factorization

Approach 1: Generalized matrix factorization



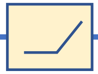
matrix factorization

factorized item vector

$\odot$

factorized user vector

$\mathbf{x} = \mathbf{p}_u \odot \mathbf{q}_i$



$\hat{y}_{ui} = \alpha(\mathbf{h}^\top \mathbf{x}),$

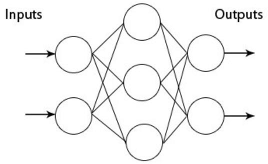
- p: factorized user vector
- q: factorized item vector
- α : activation function
- h : weight of the output layer.
- $\hat{y}_{ui}$: prediction score of user u for item i

Approach 2: MLP (multi-layer perceptron)

concat

item vector

user vector

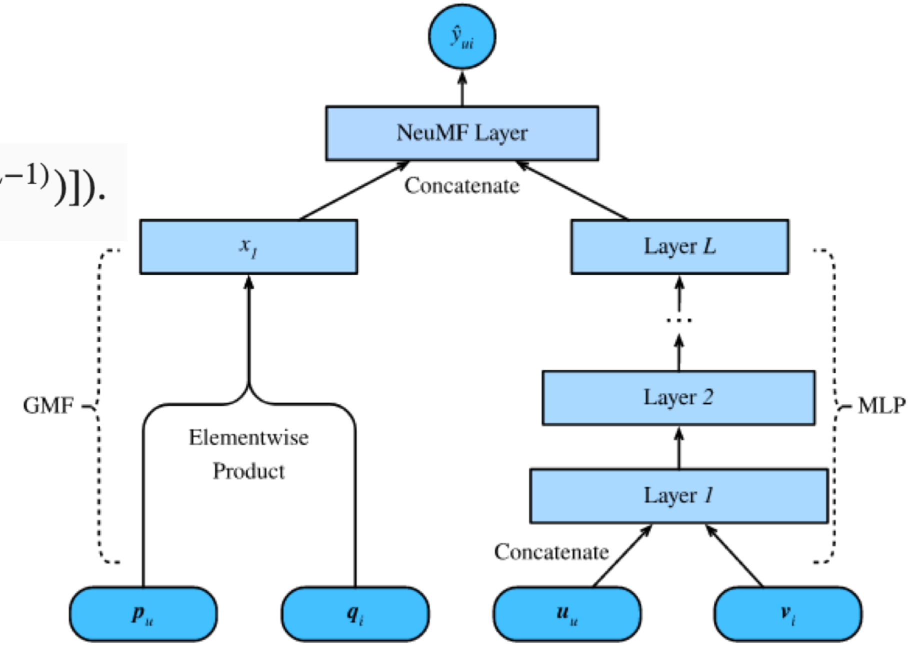


concat

$\hat{y}_{ui} = \sigma(\mathbf{h}^\top[\mathbf{x}, \phi^L(z^{(L-1)})]).$

$$
\begin{aligned}
z^{(1)} &= \phi_1(\mathbf{U}_u, \mathbf{V}_i) = [\mathbf{U}_u, \mathbf{V}_i] \\
\phi^{(2)}(z^{(1)}) &= \alpha^1(\mathbf{W}^{(2)} z^{(1)} + b^{(2)}) \\
&\ldots \\
\phi^{(L)}(z^{(L-1)}) &= \alpha^L(\mathbf{W}^{(L)} z^{(L-1)} + b^{(L)})) \\
\hat{y}_{ui} &= \alpha(\mathbf{h}^\top \phi^L(z^{(L-1)}))
\end{aligned}
$$

- u: user vector
- v: item vector



16.6. Neural Collaborative Filtering for Personalized Ranking (NeuMF)

Negative sampling

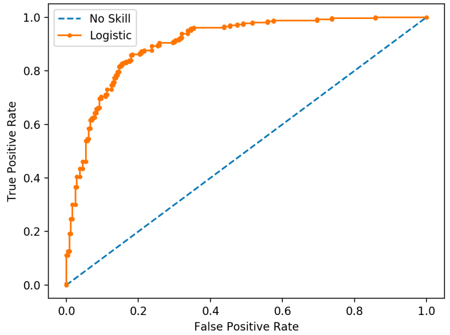
- sampling items that a user has not interacted and giving its label as zero (not-relevant)

Performance Evaluation

- Hit rate at given cutting off ℓ

$$\text{Hit}@\ell = \frac{1}{m} \sum_{u \in \mathcal{U}} \mathbf{1}(rank_{u,g_u} <= \ell),$$

- AUC (Area under ROC curve)
  - 0.5 (worst) ~ 1 (best)
- Precision, Recall, nDCG..




```
class NeuMF(nn.Block):
    def __init__(self, num_factors, num_users, num_items, nums_hiddens,
                 **kwargs):
        super(NeuMF, self).__init__(**kwargs)
        self.P = nn.Embedding(num_users, num_factors)
        self.Q = nn.Embedding(num_items, num_factors)
        self.U = nn.Embedding(num_users, num_factors)
        self.V = nn.Embedding(num_items, num_factors)
        self.mlp = nn.Sequential()
        for num_hiddens in nums_hiddens:
            self.mlp.add(nn.Dense(num_hiddens, activation='relu',
                                  use_bias=True))
        self.prediction_layer = nn.Dense(1, activation='sigmoid', use_bias=False)

    def forward(self, user_id, item_id):
        p_mf = self.P(user_id)
        q_mf = self.Q(item_id)
        gmf = p_mf * q_mf
        p_mlp = self.U(user_id)
        q_mlp = self.V(item_id)
        mlp = self.mlp(np.concatenate([p_mlp, q_mlp], axis=1))
        con_res = np.concatenate([gmf, mlp], axis=1)
        return self.prediction_layer(con_res)
```

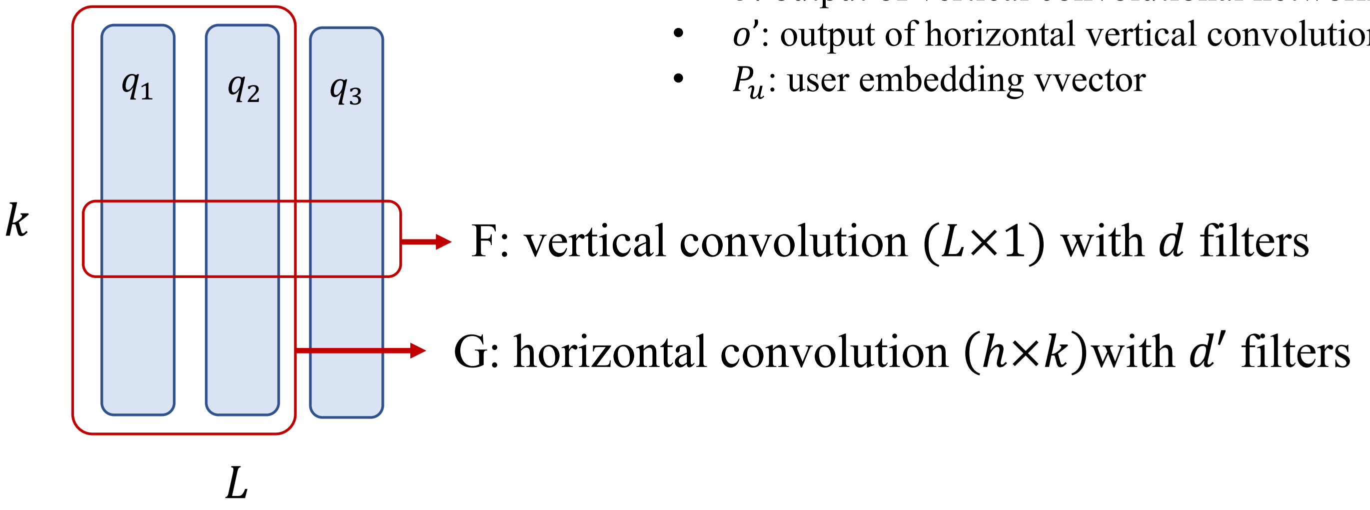
Ref: He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T.-S. (2017). Neural collaborative filtering. *Proceedings of the 26th international conference on world wide web*

Model architecture

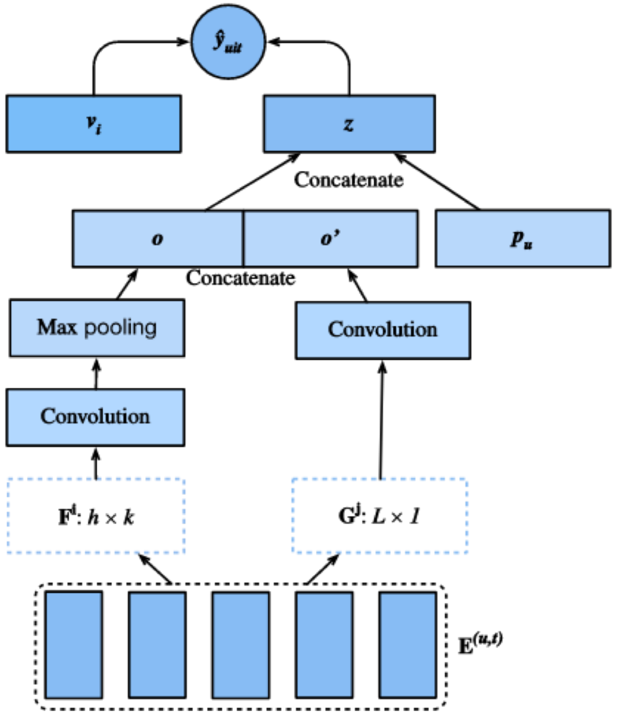
- Modeling users' temporal behavioral patterns and discovering their interest drift

$$\mathbf{E}^{(u,t)} = [\mathbf{q}_{S^u_{t-L}}, \ldots, \mathbf{q}_{S^u_{t-2}}, \mathbf{q}_{S^u_{t-1}}]^\top$$

- E: user u's sequence of item interest
- q: item vector
- o: output of vertical convolutional network
- o': output of horizontal vertical convolutional network
- P_u: user embedding vvector



$$\mathbf{o} = \mathrm{HConv}(\mathbf{E}^{(u,t)}, \mathbf{F})$$
$$\mathbf{o}' = \mathrm{VConv}(\mathbf{E}^{(u,t)}, \mathbf{G}),$$
$$\mathbf{z} = \phi(\mathbf{W}[\mathbf{o}, \mathbf{o}']^\top + \mathbf{b}),$$
$$\hat{y}_{uit} = \mathbf{v}_i \cdot [\mathbf{z}, \mathbf{p}_u]^\top + \mathbf{b}'_i,$$



- Horizontal convolution: to discover union-level sequence patterns
- Vertical convolution: aiming to uncover point-level sequence patterns

Data sampling

- Modeling users' temporal behavioral patterns and discovering their interest drift

Tang, J., & Wang, K. (2018). Personalized top-n sequential recommendation via convolutional sequence embedding. Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (pp. 565–573).

Data sampling strategy

- Organize items in chronological order for each user's feedback
- To generate fixed sequence of items with length L,
  - The last item for the entire sequence is left out as the positive test item and sampling
  - Dividing the items with fixed window size (L+1) and setting the last item in the window becomes target
  - Negative sampling