

# Neural Collaborative Filtering and Side Information

STAT3009 Recommender Systems

by **Ben Dai** (CUHK-STAT)

on **November 19, 2024**

## » Recall: Neural Networks

### Using TF2.0 to Implement Your Own Model

M Define your **Model** mathematically

- \* Motivation, EDA, input/output, parameters/hyperparameters, etc.

T **Translate** your model into a **neural network**

F **Loss function**, **regularization**

OPT **Optimizer**, cross-validation, early stopping, etc.

### Implementation

Build Define the model using **Keras**

- \* Layers and path from **input** to **output**

Compile Compile your model with **keras.losses**, **keras.optimizers**, and **keras.metrics**

Fit Train your model with **data** and **hyperparameters**

Pred Make predictions using **model.predict**

Can we **reformulate MF** as a neural network?

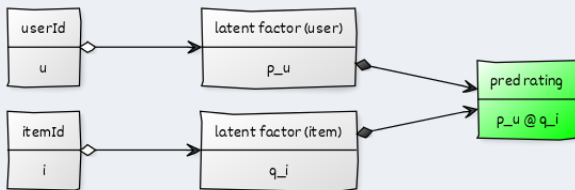
» Steps: MF to NN

M MF model:

$$\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i$$

\* Input user-item pair:  $(u, i) \rightarrow$  Output inner product:  $\mathbf{p}_u^T \mathbf{q}_i$

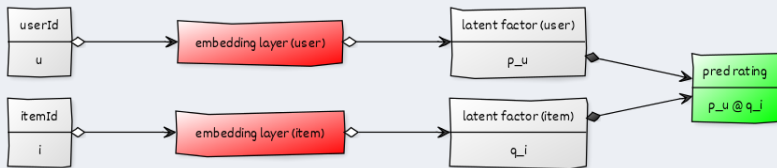
T Translate your model to a neural network



CREATED WITH YUML

The key is to map **userId/itemId** to **latent factors**. Do we have a layer to perform this mapping? YES!

## » What: Embedding Layer



CREATED WITH YUML

We can incorporate **MF** by introducing **embedding layers** into the network.

An embedding layer:

**Input** A categorical feature  $j \in \{1, \dots, J\}$

**Output** A latent factor for the input  $\mathbf{w}_j \in \mathbb{R}^r$

**Example** **userId**  $u \rightarrow$  **latent factor**  $\mathbf{p}_u$

## » What: Embedding Layer

```
tf.keras.layers.Embedding(  
    input_dim,  
    output_dim,  
    embeddings_regularizer=None,  
    activity_regularizer=None,  
    embeddings_constraint=None,  
    **kwargs)
```

1  
2  
3  
4  
5  
6  
7

### Key Arguments:

`input_dim`  $J$  - Integer. Size of the vocabulary, i.e., maximum integer index + 1.

`output_dim`  $r$  - Integer. Dimension of the dense embedding.

## » How: Embedding Layer

Steps for the embedding layer:

One-hot Encode `cate_feat` as **one-hot encoding** (a binary dummy vector)

$$j \rightarrow \mathbf{e}_{j,j} = (0, \dots, \underbrace{1}_{j\text{th}}, \dots, 0)^T \in \{0, 1\}^J,$$

Mapping Map **one-hot** encoding to a **latent factor** using the **embedding matrix**

$$j \rightarrow \mathbf{w}_j = \mathbf{e}_{j,j}^T \mathbf{W},$$

Essentially, **embedding** is the same as **latent factors**; they are just different names.

## » NCF: Embedding Layer

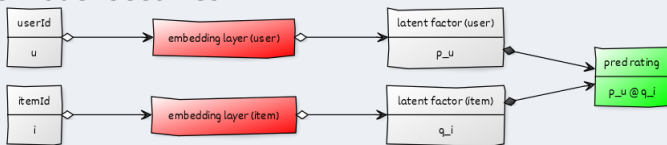
Let's summarize the **Embedding Layer**...

Mapping  $u \rightarrow \mathbf{p}_u$ , or **cate\_feat**  $\rightarrow$  **dense** representation

Params Embedding matrix:  $\# \text{Users (or \#Items)} \times \# \text{LatentFactors}$

hp  $\# \text{LatentFactors}$  or **embedding size**

M The model becomes



CREATED WITH YUML

F The **formulation** becomes:

$$\hat{f}_{\theta} = \underset{f_{\theta}}{\operatorname{argmin}} \frac{1}{|\Omega|} \sum_{(u,i) \in \Omega} (r_{ui} - f_{\theta}(u,i))^2 + \lambda \operatorname{Reg}(\theta)$$

code Demo in **Colab**

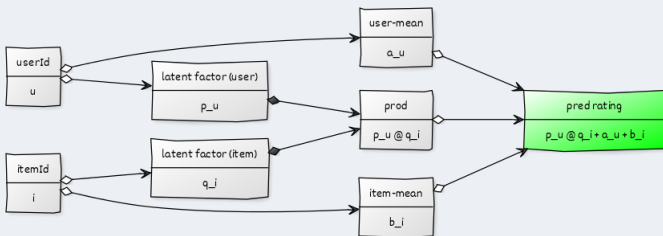
## » In-Class Practice

### In-class practice:

#### M MF-mean Model:

$$\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i + a_u + b_i$$

NN-view From a neural network perspective:



CREATED WITH YUMI



## » NCF: Nonlinear Interaction

Incorporating **neural networks** to model **high-order** interactions between users and items.

M Model:

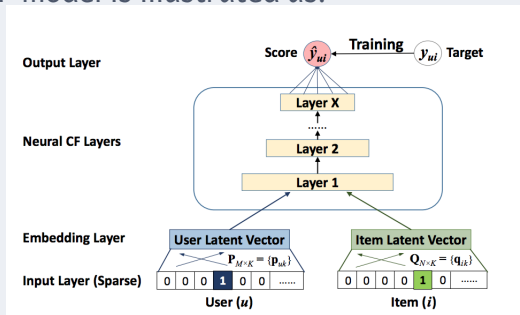
Input User-item pair  $(u, i) \in \{1, \dots, n\} \times \{1, \dots, m\}$   
Output Predicted rating  
Math

$$\hat{r}_{ui} = \phi(\mathbf{p}_u, \mathbf{q}_i),$$

where  $\phi(\cdot)$  is a nonlinear function

## » NCF: RS Formulation

Model The NCF model is illustrated as:



1

Params All parameters in the network include: (i) two embedding matrices in the embedding layers and (ii) weights in the other layers

hp **Embedding size and network architecture**

<sup>1</sup>He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T. S. (2017, April). Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web (pp. 173-182).

## » NCF: Implementation

Data  $(u, i) \rightarrow r_{ui}$

Loss Mean Squared Error (MSE)

Metrics RMSE, MAE, etc.

Opt SGD/Adam

**InClass demo: Neural Collaborative Filtering #1** in Colab

## » ANCF: Additional Formulation

Motivation (from additive semi-parametric model)

The main effect is captured by **first-order MF** interaction; we then introduce **additional** latent factors to model **higher-order** interactions.

Model We aim to retain the **MF** terms in the model:

$$\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i + \phi([\mathbf{s}_u, \mathbf{t}_i])$$

## » ANCF: Additional Formulation

Model We aim to retain the **MF** terms in the model:

$$\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i + f([\mathbf{s}_u, \mathbf{t}_i])$$

Parameters All parameters in the network include: (i) **two** embedding matrices in the embedding layers and (ii) **weights** in the other layers.

Parameters **Embedding size** and **network architecture**

**InClass demo: Neural Collaborative Filtering #2**

## » Neural-NCF: Neural Formulation (Optional)

More general operators based on MF terms:

- \* Without assuming a **additive** structure
- \* Interaction between **first-order** and **higher-order** interactions

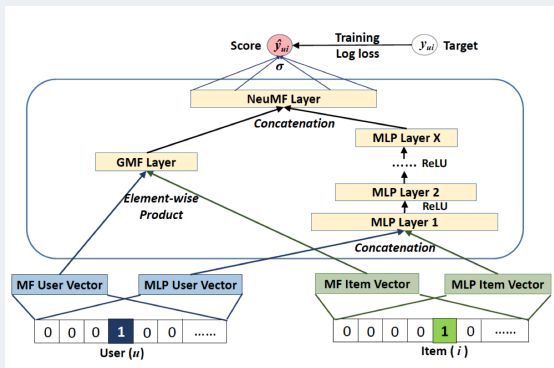
$$\hat{r}_{ui} = \phi([\mathbf{p}_u \circ \mathbf{q}_i, \psi([\mathbf{s}_u, \mathbf{t}_i])]),$$

where  $\phi$  and  $\psi$  are custom nonlinear functions.

## » Neural-NCF: Neural Formulation (Optional)

- \* Transform the model into a neural network:

$$\hat{r}_{ui} = \phi([\mathbf{p}_u \circ \mathbf{q}_i, \psi(\mathbf{s}_u, \mathbf{t}_i)])$$



2

<sup>2</sup>He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T. S. (2017, April). Neural Collaborative Filtering. In Proceedings of the 26th International Conference on World Wide Web (pp. 173-182).

## » NCF: RS formulation (Optional)

Model The model becomes

$$\hat{r}_{ui} = \phi([\mathbf{p}_u \circ \mathbf{q}_i, \psi(\mathbf{s}_u, \mathbf{t}_i)])$$

Params All params in the network: (i) **four** embedding matrices in embedding layers and (ii) **weights** in the other layers

hp **embedding size** and **network architecture**

Data  $(u, i) \rightarrow r_{ui}$

Loss MSE

Metrics RMSE, MAE, ...

Opt SGD/B-SGD

**InClass demo: Neural collaborative filtering #3**



## » MF: Cold-Start Issue

### Cold-Start Issue:

Problem Completely **new** users/items in recommender systems

Prediction Existing methods:

User/Item-Average Unable to compute a meaningful average

Correlation Cannot compute distance

MF/SVD For example, if user  $u$  has **no** ratings in the system  
( $I_u = \emptyset$ ):

Conclusion For **cold-start users/items**, most existing methods **fail**  
to provide a solution.

**We NEED side information!**

## » MovieLens: Side Information

**YES**, we have **side information**.

For example, in the **MovieLens** dataset:

User USER\_ID, AGE, GENDER, OCCUPATION, ZIP\_CODE,  
RATING\_MEAN, RATING\_COUNT, RATING\_QUANTILE,  
etc.

Item ITEM\_ID, DATE, GENRE,  
RATING\_MEAN, RATING\_COUNT, RATING\_QUANTILE,  
etc.

Other DATE

### **Key Message:**

- \* **Exploratory Data Analysis (EDA)** in **MovieLens** indicates that **side information** is critical.
- \* **Side information** is promising for solving **cold-start** issues.
- \* How to incorporate **side information** to build a new recommender system.

## » Conclusion from EDA (MovieLens)

### Insights from EDA in MovieLens:

- \* **Side information** or user/item features are critical for predicting ratings:  $f(\mathbf{x}_u, \mathbf{z}_i) \rightarrow r_{ui}$
- \* **Personalization/itemization** remains important even for users/items with similar features—**MF** terms are necessary:  $f(\mathbf{x}_u, \mathbf{z}_i) + \mathbf{p}_u^T \mathbf{q}_i \rightarrow r_{ui}$
- **Joint effect** of features should be considered in recommender system modeling:

$$f(\mathbf{p}(u, \mathbf{x}_u), \mathbf{q}(i, \mathbf{z}_i)) \rightarrow r_{ui}$$

## » MovieLens: Side Information

Update our setting:

Rating [userID, itemID, rating]:  $(u, i, r_{ui})$

Side Info User [continuous\_features, categorical\_features]

Item [continuous\_features, categorical\_features]

Examples: Continuous [AGE, RATING\_MEAN, RATING\_COUNT, RATING\_QUANTILE]

Categorical USER\_ID + [GENDER, OCCUPATION, ZIPCODE]

Train [userID, itemID, userFeatures, itemFeatures, rating]

Test [userID, itemID, userFeatures, itemFeatures, ?]

## » Pre-processing: Side Information

### Cont Continuous Features

Type → **float**

Process Standardize features

Python **SKLEARN.PREPROCESSING.STANDARDSCALER**

### Cate Categorical Features

Type → **int**

Process Label encoding

Python **SKLEARN.PREPROCESSING.LABELENCODER**

### Cont/Cate Can be continuous or categorical

Example "year" in item\_feature

EDA Use **EDA** to decide: **continuous effect** or group effect

Both Or include in both

## » Missing Data and Imputation (Optional)

	item_id	count	mean	gene	year	month
266	267	3	2.666667	unknown	NaN	NaN

Python **Sklearn: Imputation of Missing Values**<sup>3</sup>

Methods **Approaches to impute missing values**

Univariate Mean (**continuous**); Most frequent (**categorical**), etc.

Multivariate IterativeImputer, etc.

---

<sup>3</sup><https://scikit-learn.org/stable/modules/impute.html>

## » Continuous/Categorical Features

Cont Continuous features can be directly used as **inputs**.

Cate Utilizing **categorical features**;

Recall the concept of **Matrix Factorization (MF)**:

id USER\_ID  $\rightarrow \mathbf{p}_u$       ITEM\_ID  $\rightarrow \mathbf{q}_i$

We can apply the **MF** approach to other **categorical features**, illustrated here with **gender**.

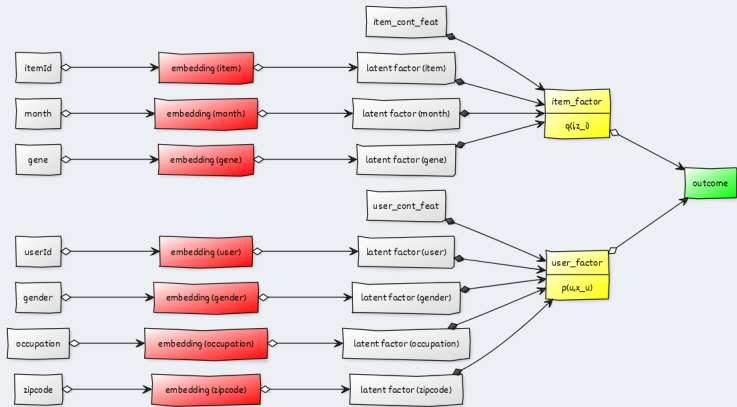
Encoding **M**  $\rightarrow 0$ , **F**  $\rightarrow 1$

(**SKLEARN.PREPROCESSING.LABELENCODER**)

Embedding **M**  $\rightarrow 0 \rightarrow \mathbf{w}_0$ ;    **F**  $\rightarrow 1 \rightarrow \mathbf{w}_1$

## » LinearRS (Optional)

### Plot Network:



CREATED WITH YUML

Code Implementation in Colab



## » Two-tower neural nets: overview (Optional)

### Motivation:

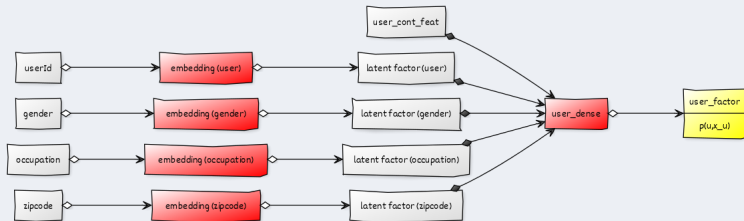
nl **nonlinear** modeling for features  
within&btw user/item feats

effect **with-in** effects of user/item features:

E.g. (GENDER + OCCUPATION) @ ITEMID  $\rightarrow$  rating

## » Two-tower neural nets: user-tower (Optional)

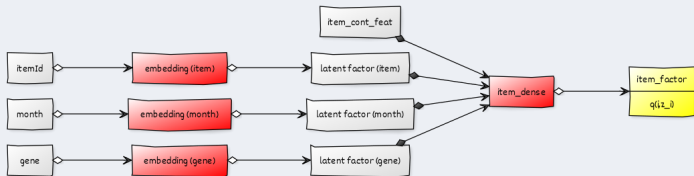
### Plot user-tower



CREATED WITH YUML

## » Two-tower neural nets: item-tower (Optional)

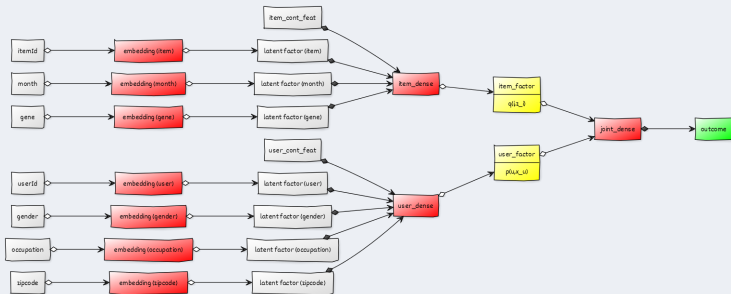
### Plot item-tower



CREATED WITH YUML

## » Two-tower neural nets (Optional)

Plot Dense layer to model the **joint** effect:



CREDITED WITH YUHL

M The final formulation is given as:

$$\rho(\mathbf{p}(u, \mathbf{x}_u), \mathbf{q}(i, \mathbf{z}_i)) \rightarrow r_{ui}$$

## » Side information in industry (Optional)

Text Searching query; reviews; description; comments;

Image profile for users; images for items;

Network social network for users; item networks

Dynamic behavior sequence; historical series

The general idea is **mapping** side information as **numerical vectors**, and feed into a two-tower based model.

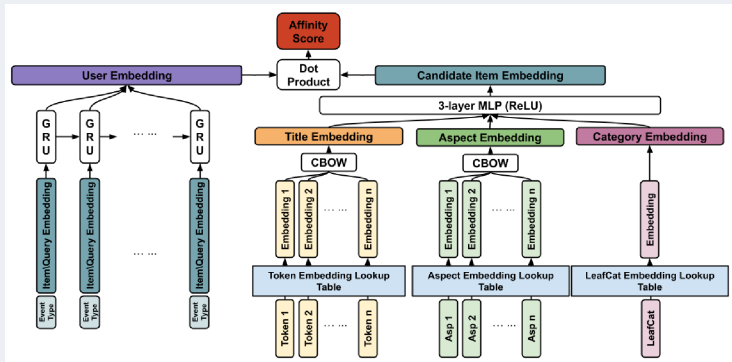
Text → embedding; Word2Vec; recurrent layers

Image → convolutional layers

Network → embedding; Node2Vec; graph convolutional layers

## » Two-tower models in industry (Optional)

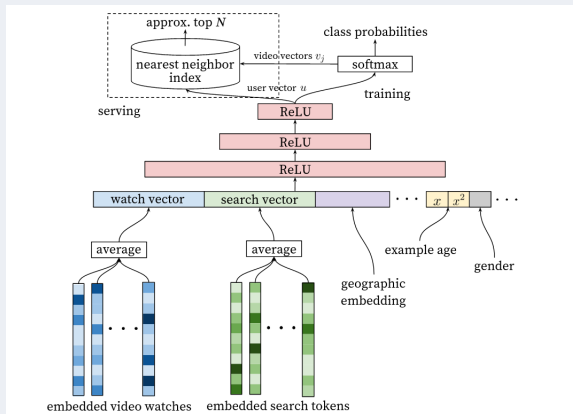
### eBay Model



Ref Wang, T., Brovman, Y. M., & Madhvanath, S. (2021). *Personalized embedding-based e-commerce recommendations at ebay.*

## » Two-tower models in industry (Optional)

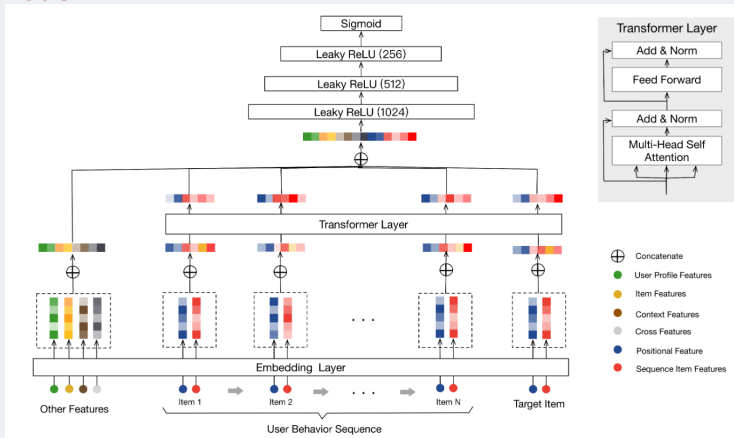
### Youtube **Model**



Ref Covington, P., Adams, J., & Sargin, E. (2016). *Deep neural networks for youtube recommendations.*

## » Two-tower models in industry (Optional)

### Alibaba Model



Ref Chen, Q., Zhao, H., Li, W., Huang, P., & Ou, W. (2019). *Behavior sequence transformer for e-commerce recommendation in alibaba.*



## » Appendix: Embedding layers in NLP (Optional)

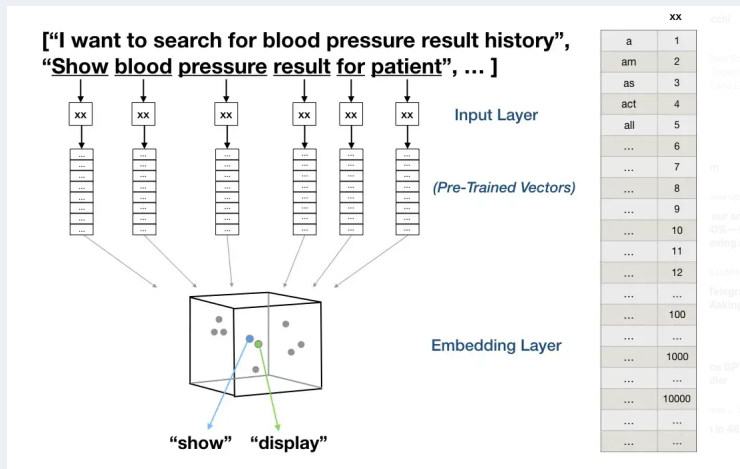
- \* **Embedding layers** is a **fundamental tool** for NLP tasks

Data Restaurant Review dataset

Training Examples	Labels
Simply loved it	Positive
Most disgusting food I have ever had	Negative
Stay away, very disgusting food!	Negative
Menu is absolutely perfect, loved it!	Positive
A really good value for money	Positive
This is a very good restaurant	Positive
Terrible experience!	Negative
This place has best food	Positive
This place has most pathetic serving food!	Negative

Goal Given **textual reviews**, can you provide a label to the **review**?

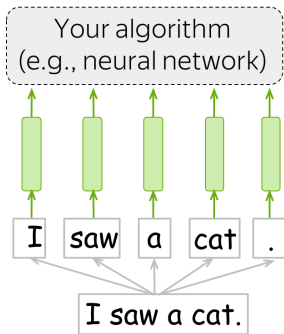
## » Appendix: Embedding layers in NLP (Optional)



source: <https://medium.com/@JMangia/coreml-with-glove-word-embedding-and-recursive-neural-network-part-2-ab238ca90970>

`tf.keras.Embedding(input_dim, output_dim, input_length)`

- \* *input\_dim* - #vocabulary: total number of words in dictionary
- \* *output\_dim* - Embedding size: dimension of latent factors
- \* *input\_length* - length of sentence/document
- \* All words share the same embedding layer



Any algorithm for solving a task

Word representation - vector  
(input for your model/algorithm)

Sequence of tokens

Text (your input)

- Embedding Using the **embedding layer** to convert **Words** to a **Matrix**; then mapping **Matrix** to a target **Outcome**
- \* **Padding words** with the same length
  - \* Train and fit the model as a general network

---

source: [https://lena-voita.github.io/nlp\\_course/word\\_embeddings.html](https://lena-voita.github.io/nlp_course/word_embeddings.html)