

Neural Matrix Factorization

STAT3009 Recommender Systems

by **Ben Dai** (CUHK)

on **Department of Statistics and Data Science**

» 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?

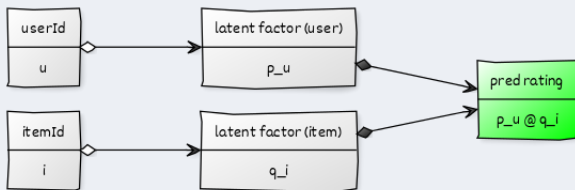
» Steps: MF to NN

M MF model:

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

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

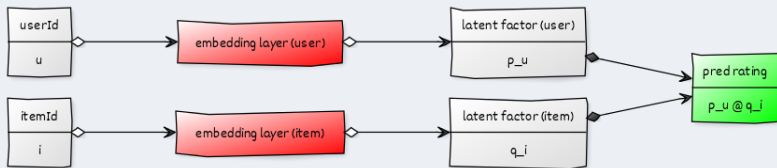
T Translate your model to a neural network



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

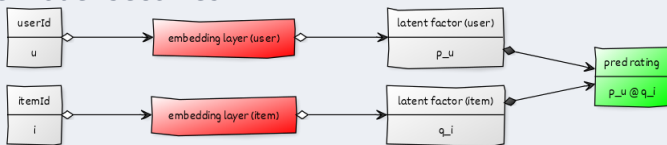
Let's summarize the **Embedding Layer**...

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**

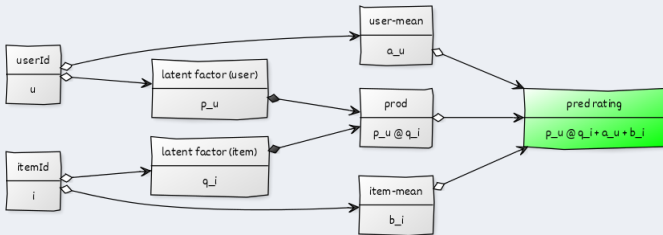
» In-Class Practice

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:

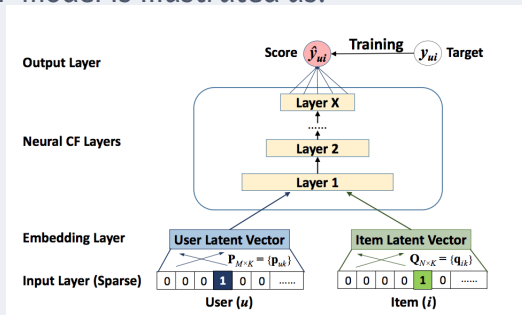
Input User-item pair $(u, i) \in \{1, \dots, n\} \times \{1, \dots, m\}$
Output Predicted rating
Math

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

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

» NCF: RS Formulation

Model The NCF model is illustrated as:



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**

¹He, 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

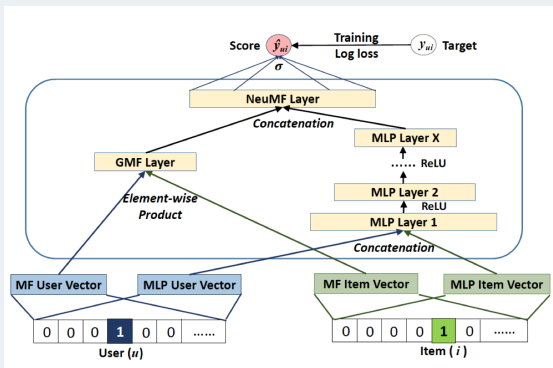
$$\hat{r}_{ui} = \phi([\mathbf{p}_u \circ \mathbf{q}_i, \psi([\mathbf{s}_u, \mathbf{t}_i])]),$$

where ϕ and ψ 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

²He, 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

» Appendix: Embedding layers in NLP (Optional)

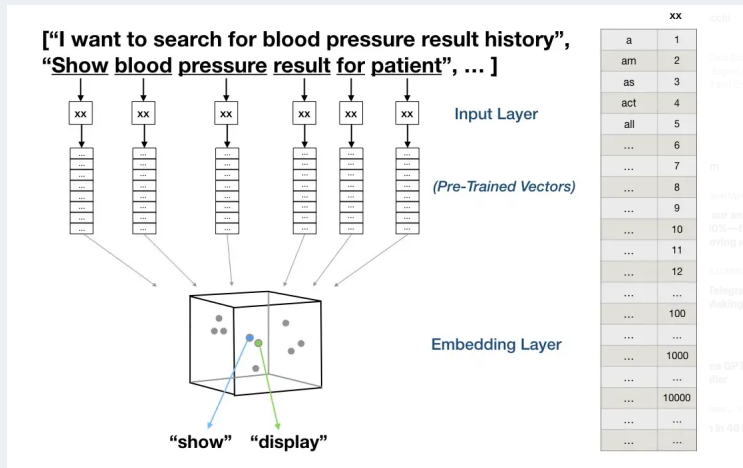
- * **Embedding layers** are a **fundamental tool** for NLP tasks

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**?

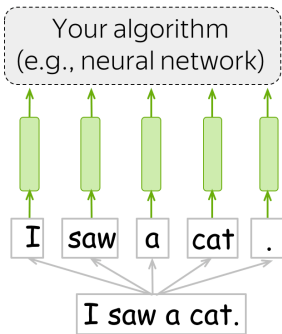
» Appendix: Embedding layers in NLP (Optional)



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