Machine Learning with Graphs (MLG)

Deep RecSys (2)

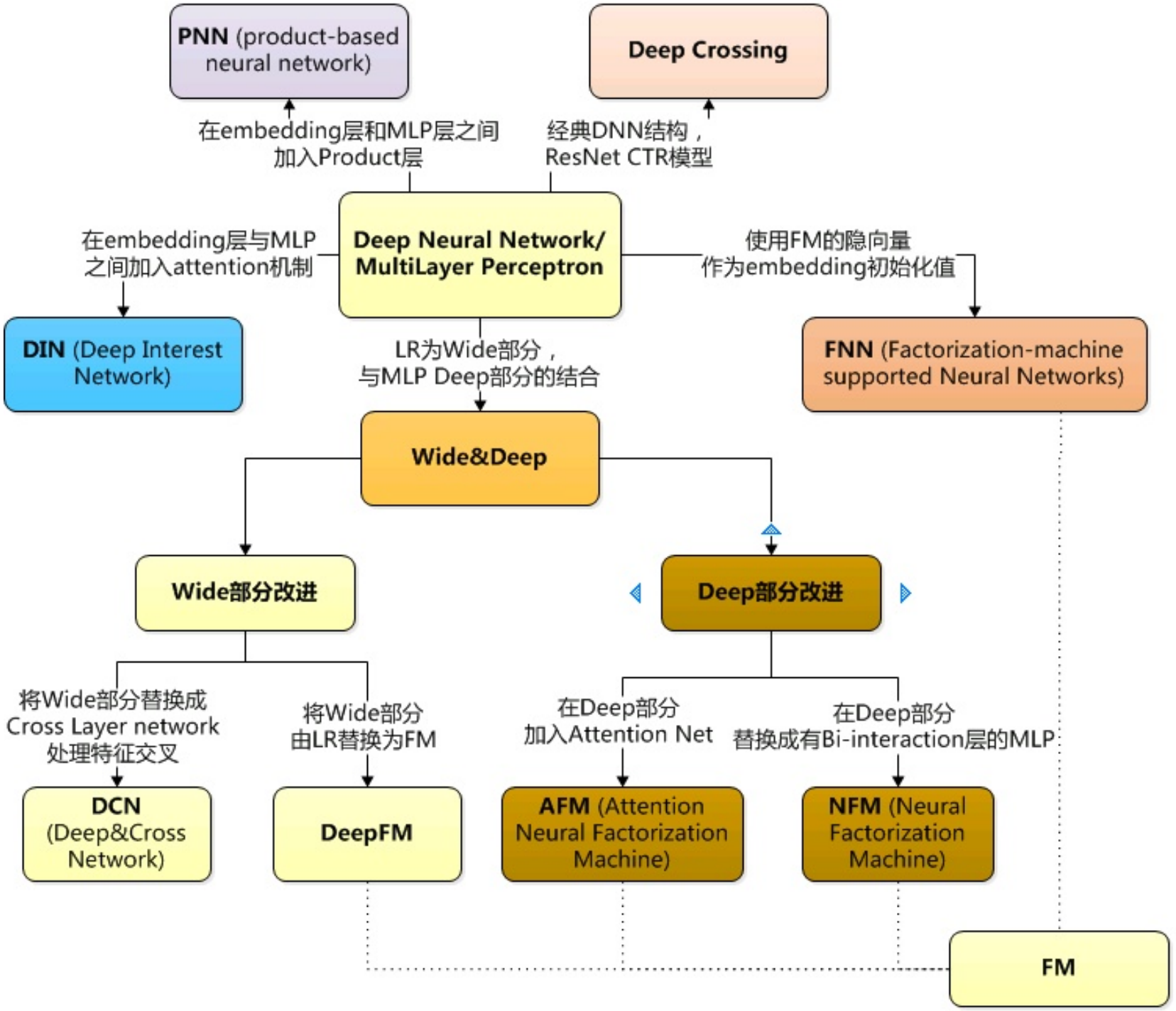
Neural network-based viewpoint of RecSys

Cheng-Te Li (李政德)

Institute of Data Science

National Cheng Kung University

chengte@mail.ncku.edu.tw


PNN (product-based neural network)
Deep Crossing
在embedding层和MLP层之间加入Product层
经典DNN结构，ResNet CTR模型
Deep Neural Network/ MultiLayer Perceptron
在embedding层与MLP之间加入attention机制
使用FM的隐向量作为embedding初始化值
DIN (Deep Interest Network)
LR为Wide部分，与MLP Deep部分的结合
FNN (Factorization-machine supported Neural Networks)
Wide&Deep
Wide部分改进
Deep部分改进
将Wide部分替换成Cross Layer network处理特征交叉
将Wide部分由LR替换为FM
在Deep部分加入Attention Net
在Deep部分替换成有Bi-interaction层的MLP
DCN (Deep&Cross Network)
DeepFM
AFM (Attention Neural Factorization Machine)
NFM (Neural Factorization Machine)
FM

Limitation of Matrix Factorization

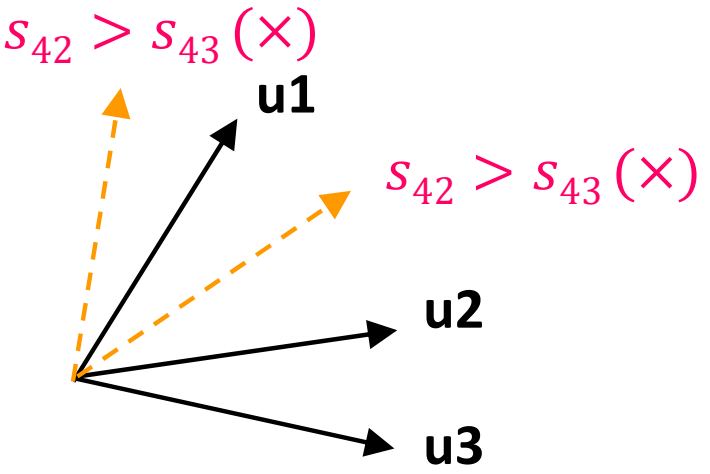
$$\hat{Y} = \mathbf{P}^T\mathbf{Q} \backsimeq \left[\cos\left(\mathbf{p}_i, \mathbf{q}_j\right)\right]$$

- The simple choice of inner product function can limit the expressiveness of a MF model
- Example

	i_1	i_2	i_3	i_4	i_5
u_1	1	1	1	0	1
u_2	0	1	1	0	0
u_3	0	1	1	1	0
u_4	1	0	1	1	1

sim(u2, u3) = 0.66
sim(u1, u2) = 0.5
sim(u1, u3) = 0.4
$s_{23} > s_{12} > s_{13}$

sim(u4, u1) = 0.6 *****
sim(u4, u2) = 0.2 *
sim(u4, u3) = 0.4 ***
$s_{41} > s_{43} > s_{42}$



If a MF model places u4 closest to u1, it will result in u4 closer to u2 than u3

Ground-truth by Jaccard similarity $s_{ij} = \frac{|\mathcal{R}_i| \cap |\mathcal{R}_j|}{|\mathcal{R}_i| \cup |\mathcal{R}_j|}$

- → Inner product can incur a large **ranking** loss of MF

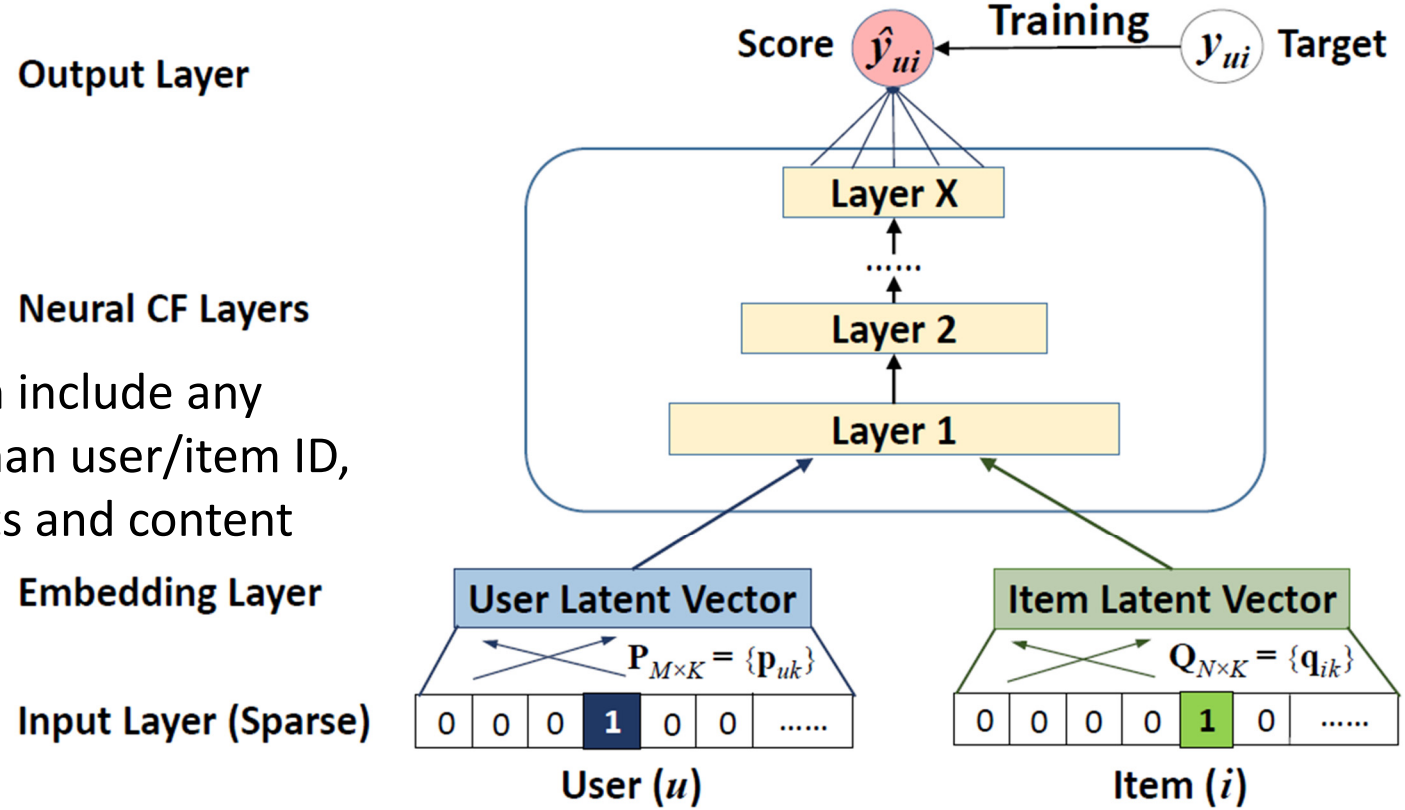
Neural Collaborative Filtering (NCF)

- NCF: a framework to learn user-item interaction function via a DNN

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

  - A NCF instance: Generalized MF (GMF)
  - A NCF instance: MLP to model non-linearaities
  - A NCF instance: fuse GMF and MLP

Input feature vectors can include any categorical variables other than user/item ID, such as attributes, contexts and content



Generalized MF in NCF

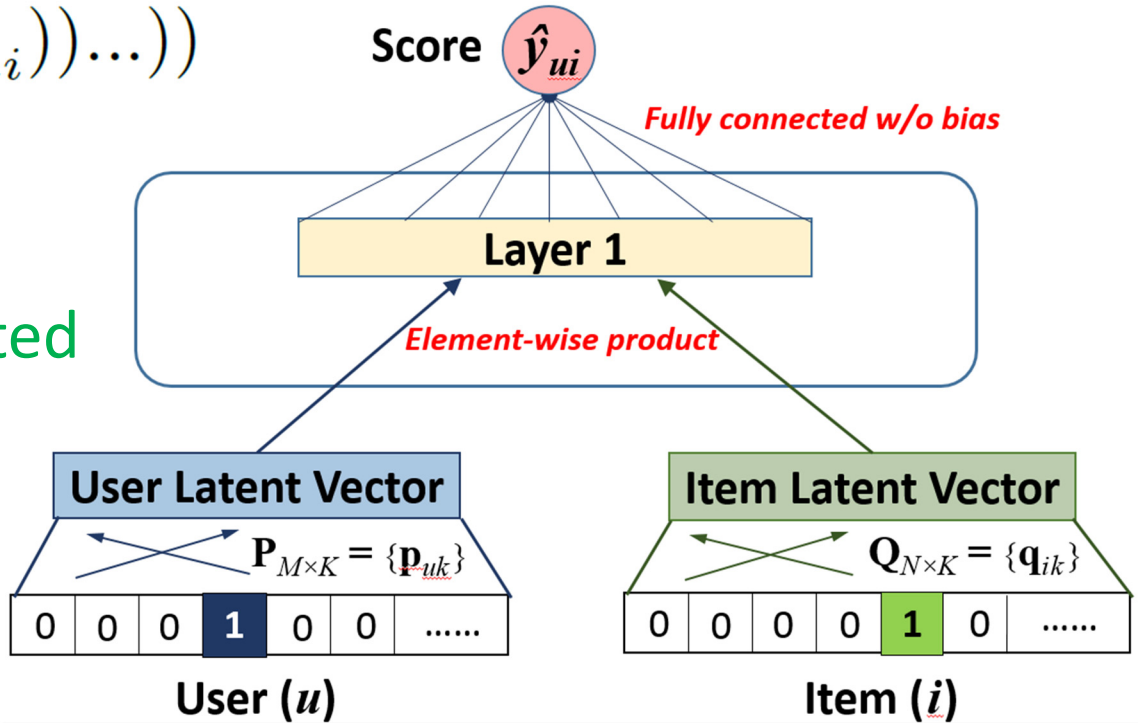
- NCF can express and generalize MF

$$\hat{y}_{ui} = f(\mathbf{p}_u, \mathbf{q}_i)$$
$$= \phi_{out}(\phi_X(...\phi_2(\phi_1(\mathbf{p}_u, \mathbf{q}_i))...))$$

Let we define ***Layer* 1** as an element-wise product, and ***Output Layer*** as a fully connected layer without bias, we have:

$$\hat{y}_{ui} = a_{out}(\mathbf{h}^T(\mathbf{p}_u \odot \mathbf{q}_i))$$

If a_{out} is an identity function, and h is a uniform vector of "**1**", → we have the MF model



Multi-Layer Perceptron (MLP) in NCF

- NCF can endow more nonlinearities to learn the interaction function

Activation function: ReLU > tanh > sigmoid

Layer 1: concatenation

$$\mathbf{z}_1 = \phi_1(\mathbf{p}_u, \mathbf{q}_i) = \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix}$$

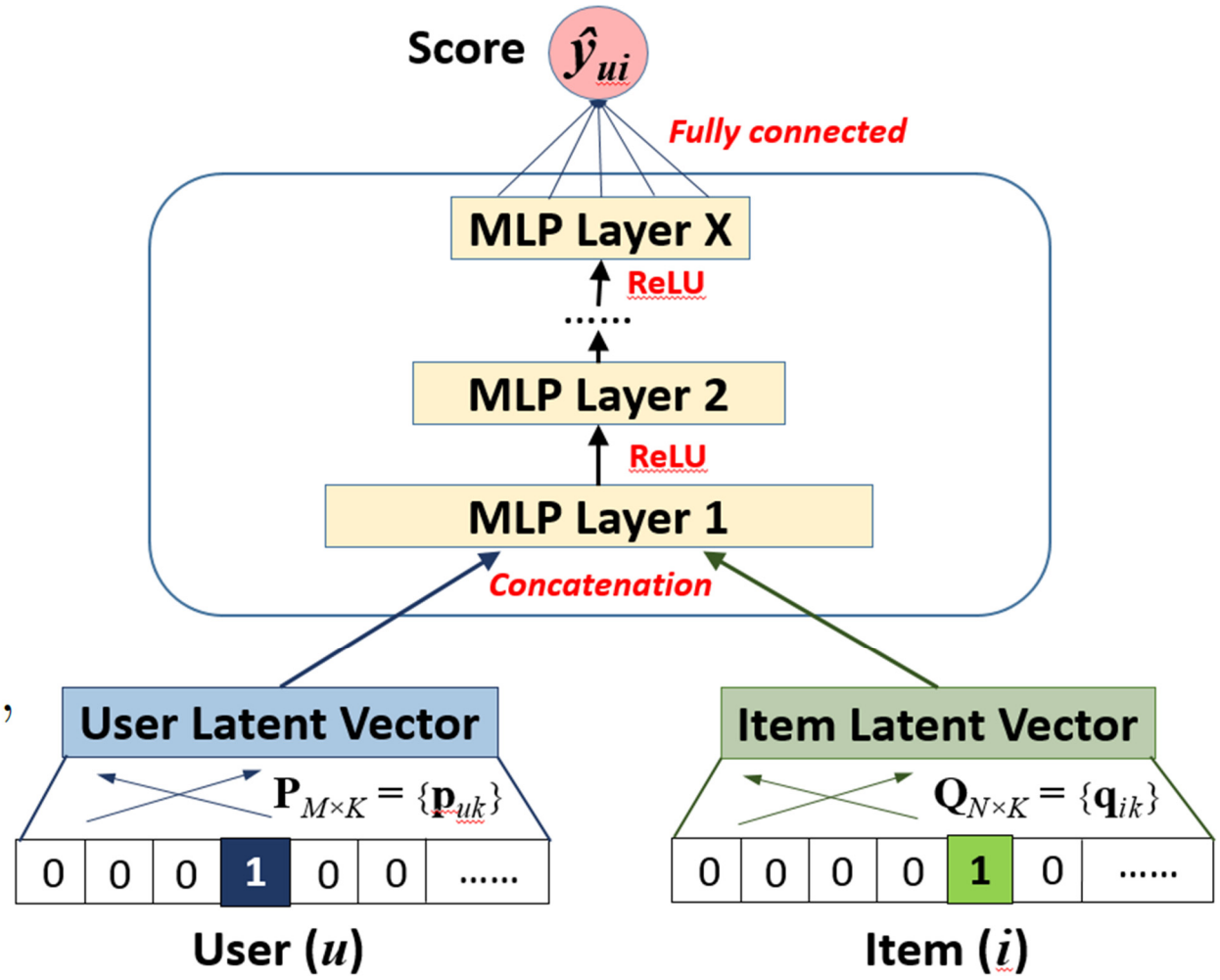
Remaining Layers:

$$\phi_2(\mathbf{z}_1) = a_2(\mathbf{W}_2^T \mathbf{z}_1 + \mathbf{b}_2),$$

$$\ldots\ldots$$

$$\phi_L(\mathbf{z}_{L-1}) = a_L(\mathbf{W}_L^T \mathbf{z}_{L-1} + \mathbf{b}_L),$$

$$\hat{y}_{ui} = \sigma(\mathbf{h}^T \phi_L(\mathbf{z}_{L-1})),$$

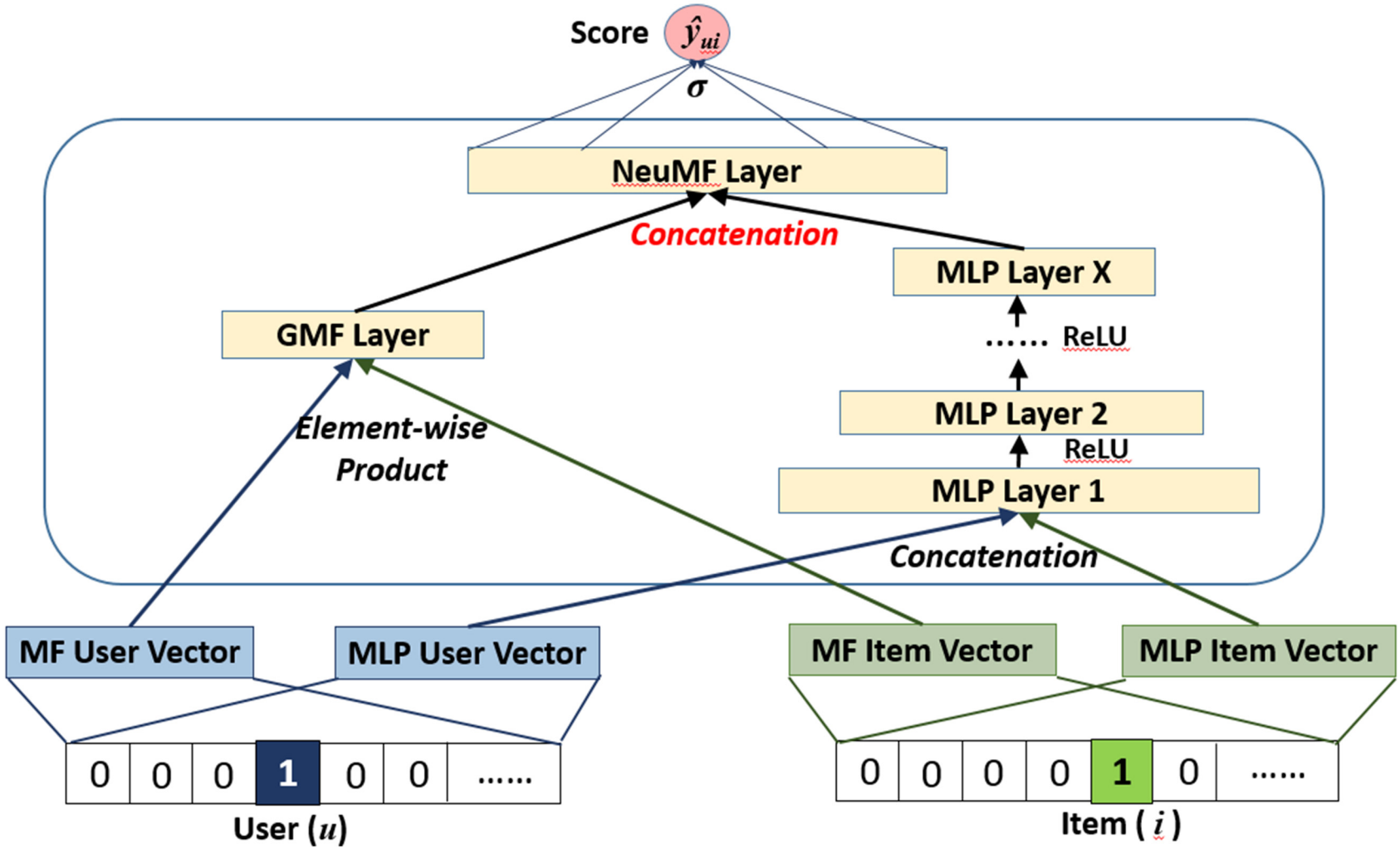


MF vs. MLP

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

- MF uses inner product as interaction function (IF)
  - Latent factors are independent with each other
  - Empirically has good generalization ability
- MLP uses nonlinearity to learn IF
  - Latent factors are not independent with each other
  - Learn IF from data → better representation ability
  - However, its generalization ability is unknown
    - It is seldom explored in recommender literature/challenge
- Can we fuse MF and MLP to get a powerful model?

NeuMF: Neural Matrix Factorization



For explicit feedback (e.g., ratings 1-5):

Regression loss

$$L_r = \sum_{(u,i)\in\mathcal{Y}} (y_{ui} - \hat{y}_{ui})^2 + \lambda_\Theta ||\Theta||^2$$

For implicit feedback (e.g., watches, 0/1):

Classification loss

$$L_c = \sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log(1 - \hat{y}_{ui}) + \lambda_\Theta ||\Theta||^2$$

Two Main Issues of RecSys

- **Memorization** (記憶)
  - Learn the frequent co-occurrence of items or features and exploit the correlation available in the historical data
  - → Obtain better accuracy
  - → **Wide** models
- **Generalization** (泛化)
  - Based on transitivity of correlation and explores new feature combinations that have never or rarely occurred in the past
  - → Bring higher diversity/novelty
  - → **Deep** models

Wide Model vs. Deep Model

Wide Model

Generalized Linear Model

$$y = \sigma\left(\mathbf{w}^{\mathrm{T}}[\mathbf{x}, \phi(\mathbf{x})] + b\right)$$

$\phi(\mathbf{x})$: cross features
(e.g., gender=F & lang=EN)

Wide = long feature vector
& feature engineering

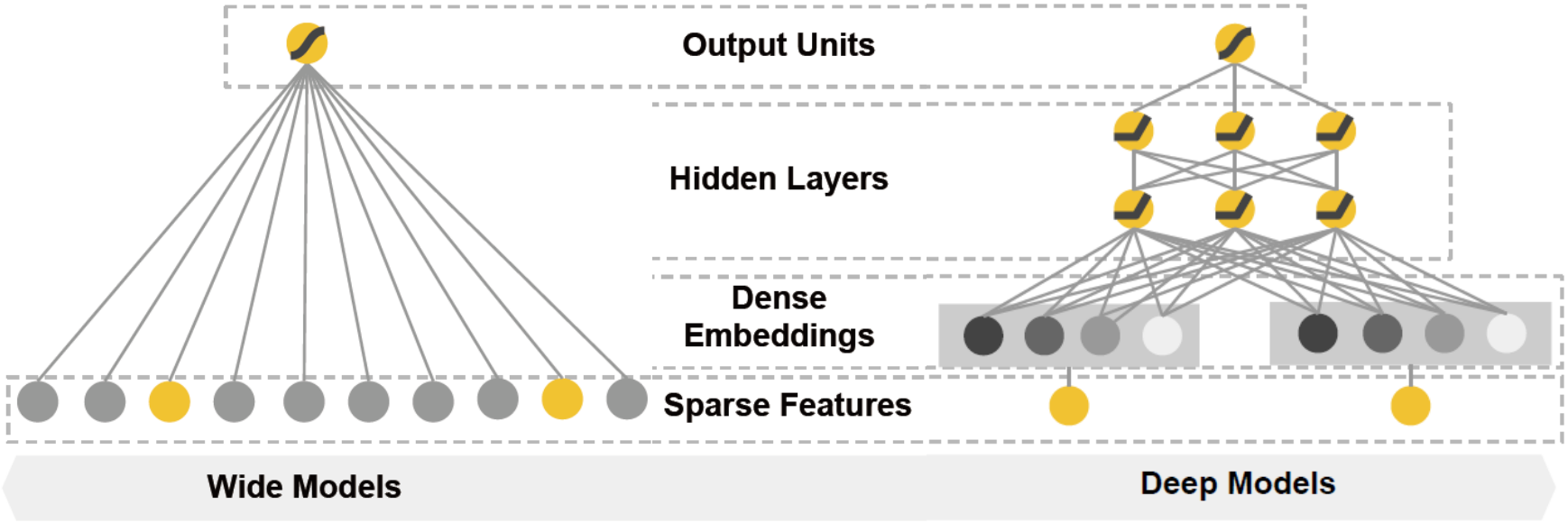
Deep Model

Deep Neural Networks

$$a^{(l+1)} = f\left(\mathbf{w}^{(l)} a^{(l)} + b^{(l)}\right)$$

Mapping high-dim vector
to low-dim embedding layers

Deep = learn high-order feature interaction
no feature engineering

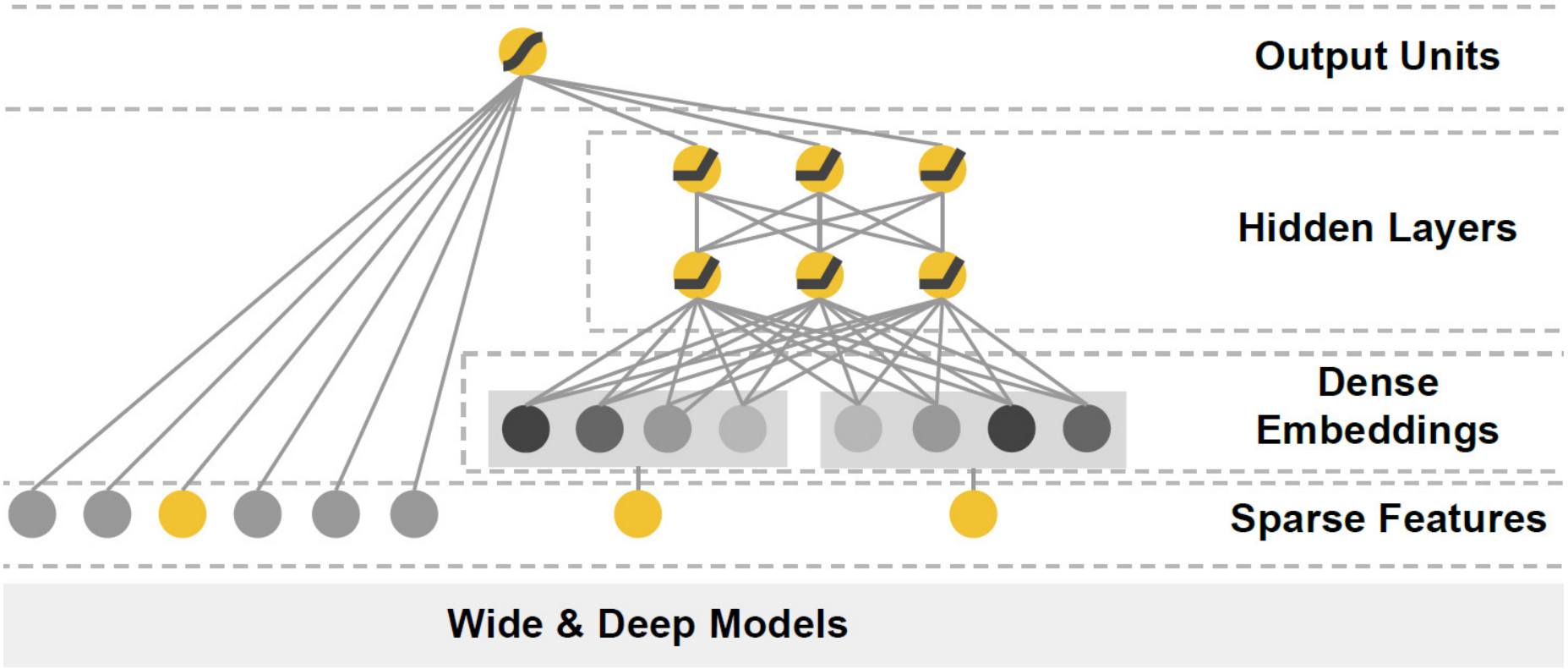


Wide & Deep Model

$$y = \sigma\left(\mathbf{w}_{wide}^{\mathrm{T}}[\mathbf{x}, \phi(\mathbf{x})] + \mathbf{w}_{deep}^{\mathrm{T}}{}^{(l)} a^{(l)} + b\right)$$

Optimizer:
Follow-the-regularized-leader (FTRL) algorithm

Optimizer:
AdaGrad



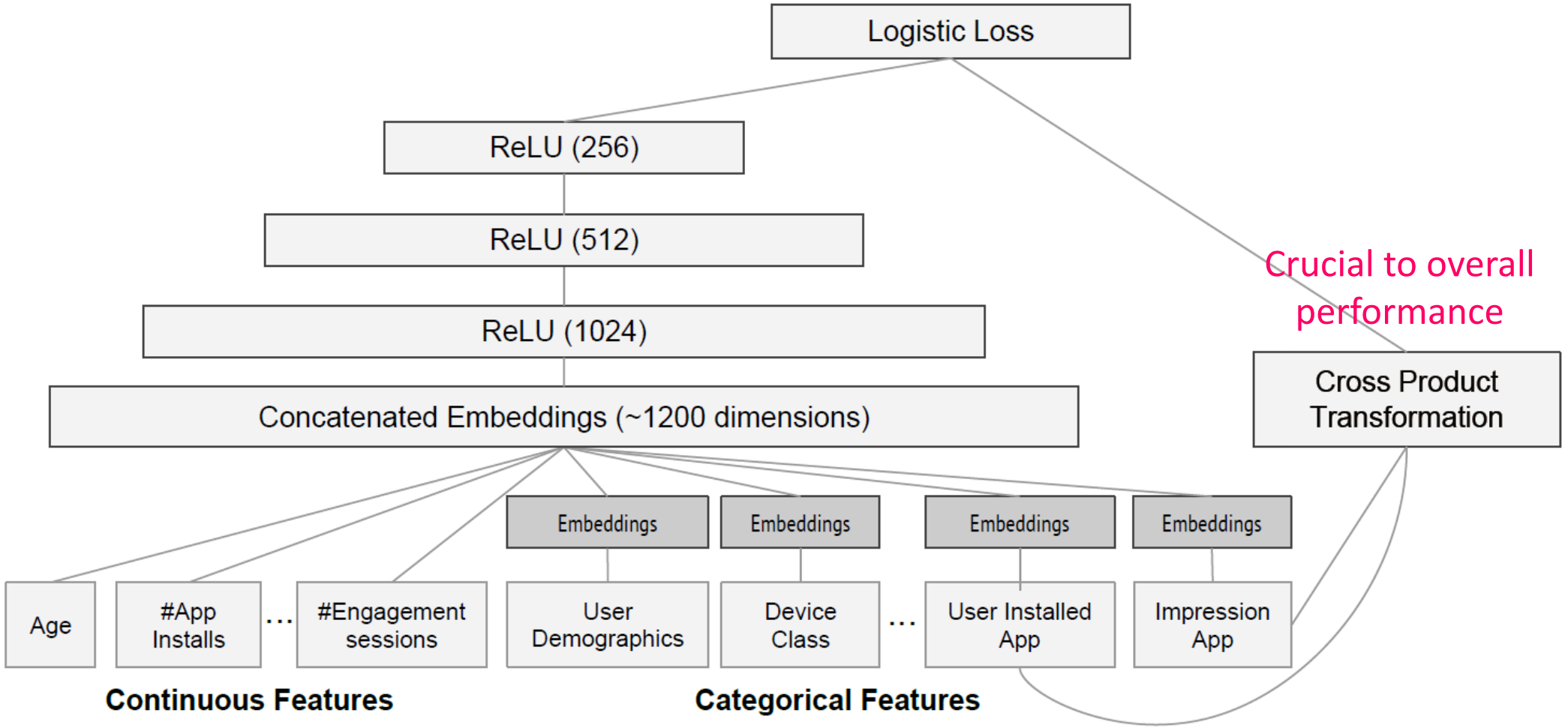
Wide&Deep for Google App RecSys

Deep:

(1) Continuous features to FC layer

(2) Categorical feature to Embedding layer to FC layer

Wide: cross features of User-installed APP and Impression APP



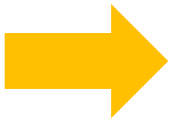
DNN to Deal with Diverse Features

- Handling diverse feature types
  - One-hot: e.g., user ID, item ID, item category
  - Multi-hot: e.g., query terms, item description

  } Sparse feat.

  - Numerical: e.g., #clicks, item price, user age → Dense feat.
- NO feature engineering to learn cross features
  - User features as raw as possible
  - Let DNN take care of the rest

Complicated data is impossible to learn cross features



Feature name	Type	Dimension
Query	Text	49,292
Keyword	Text	49,292
Title	Text	49,292
MatchType	Category	4
CampaignID	ID	10,001

DeepCross

Benefits of residual nets:
(1) Deal with overfitting
(2) Speedup convergence via short-cut gradient BP
(3) ReLU alleviates gradient vanishing

Scoring Layer:
Logic, softmax, or linear prediction

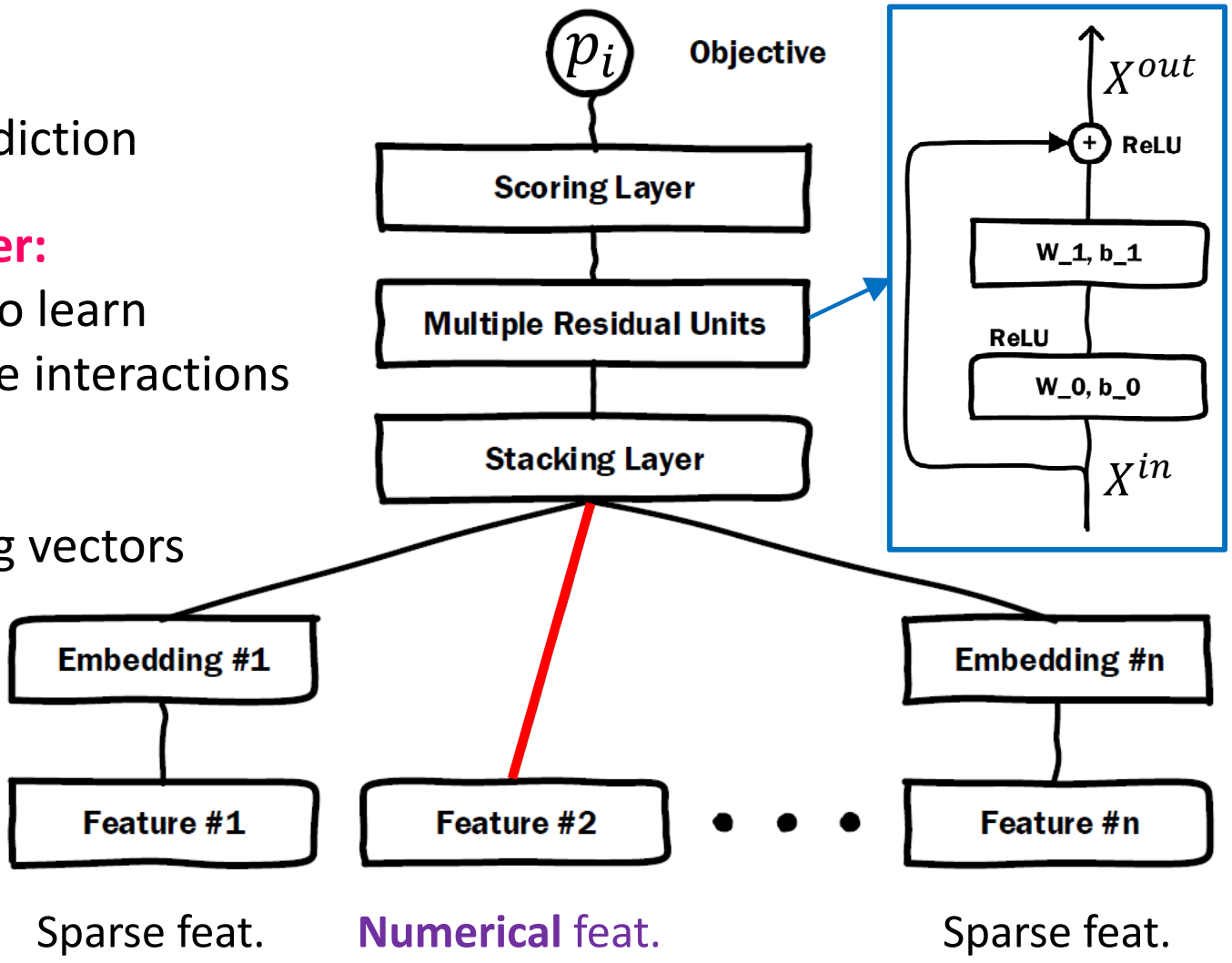
Multiple Residual Units Layer:
MLP with residual addition to learn
non-linear high-order feature interactions

Stacking Layer:
Concatenation of embedding vectors

Embedding Layer:
Turn sparse features
to dense vectors

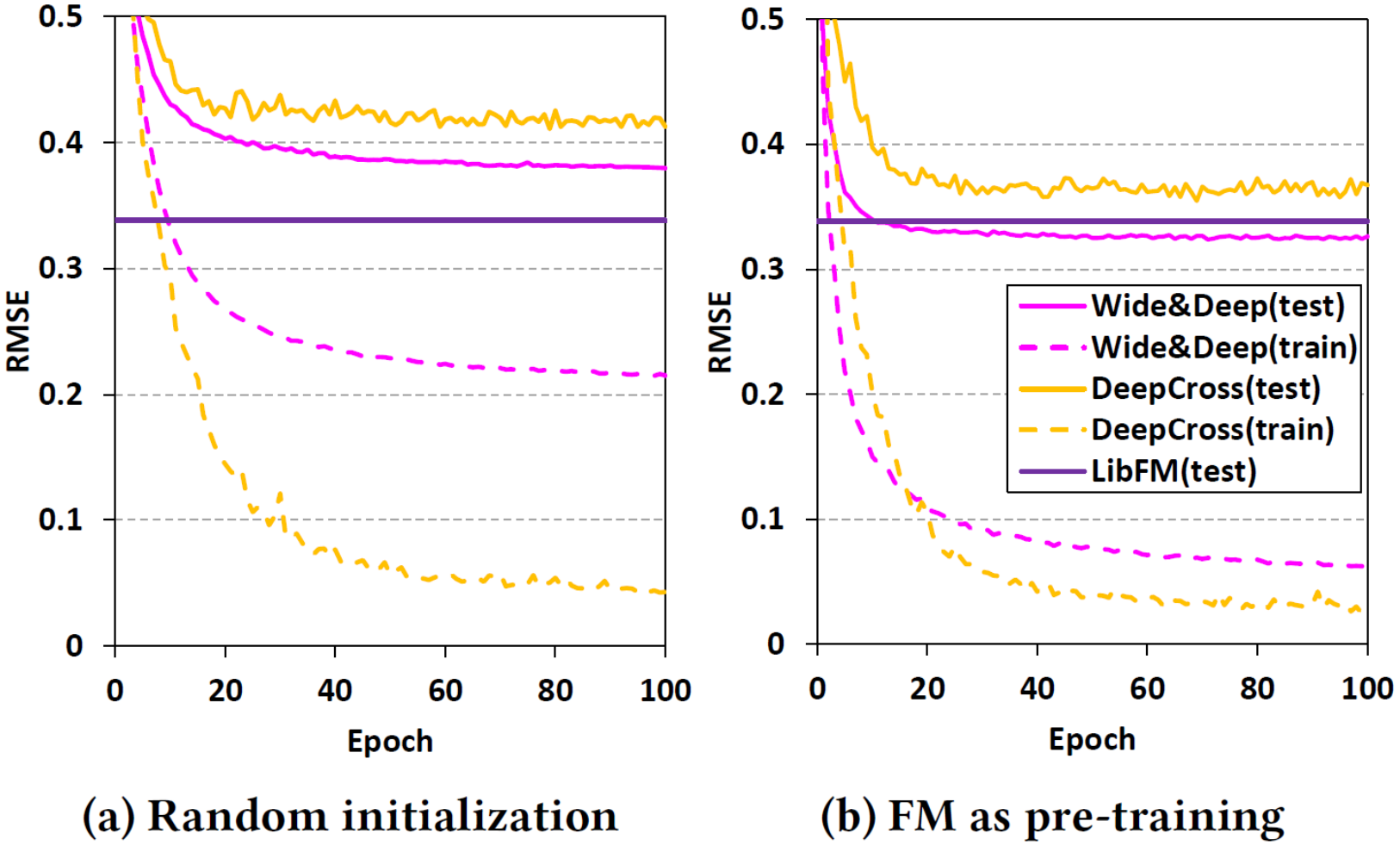
$$\mathbf{X}_j^O = ReLU\left(\mathbf{W}_j\mathbf{X}_j^I + \mathbf{b}_j\right)$$

j: index of individual feature



How do Wide&Deep and DeepCross perform?

- Both do not work well for learning feature interactions
- FM as pre-training of feature embeddings show the effectiveness learning feature interactions!

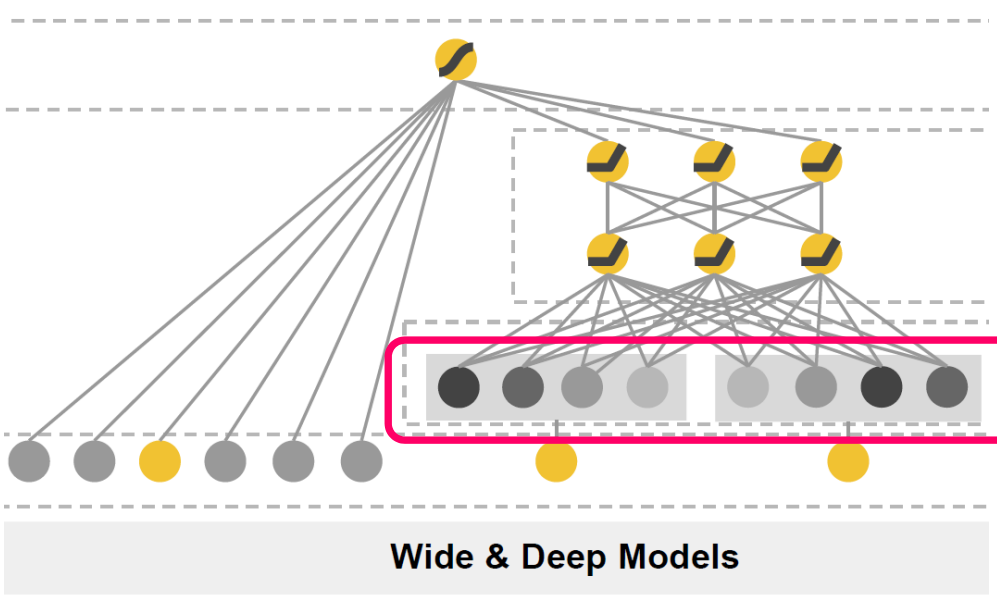


(a) Random initialization (b) FM as pre-training

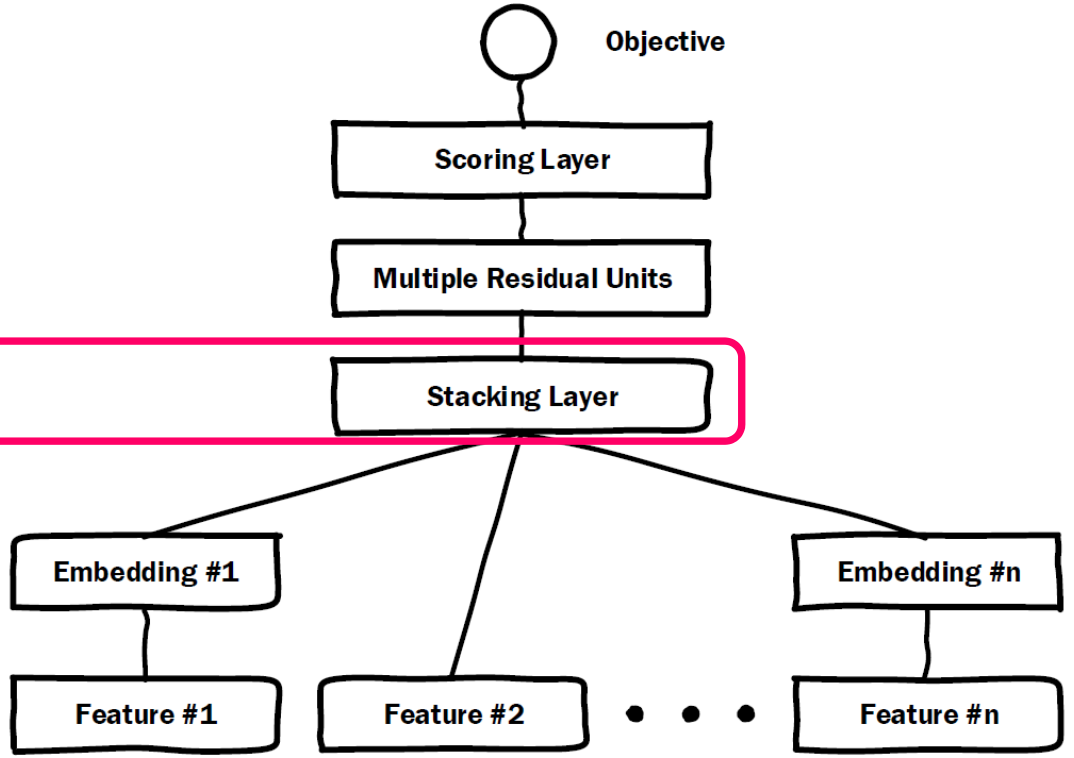
Limitation of Existing DL RecSys

- Both DL methods expect that deep layers can learn feature interactions of arbitrary orders
- Simply concatenating feature embeddings carries too little info about feature interactions in the low level
  - No guidance on feature interaction

Remember FM: $\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \mathbf{v}_i^{\mathrm{T}} \mathbf{v}_j x_i x_j$

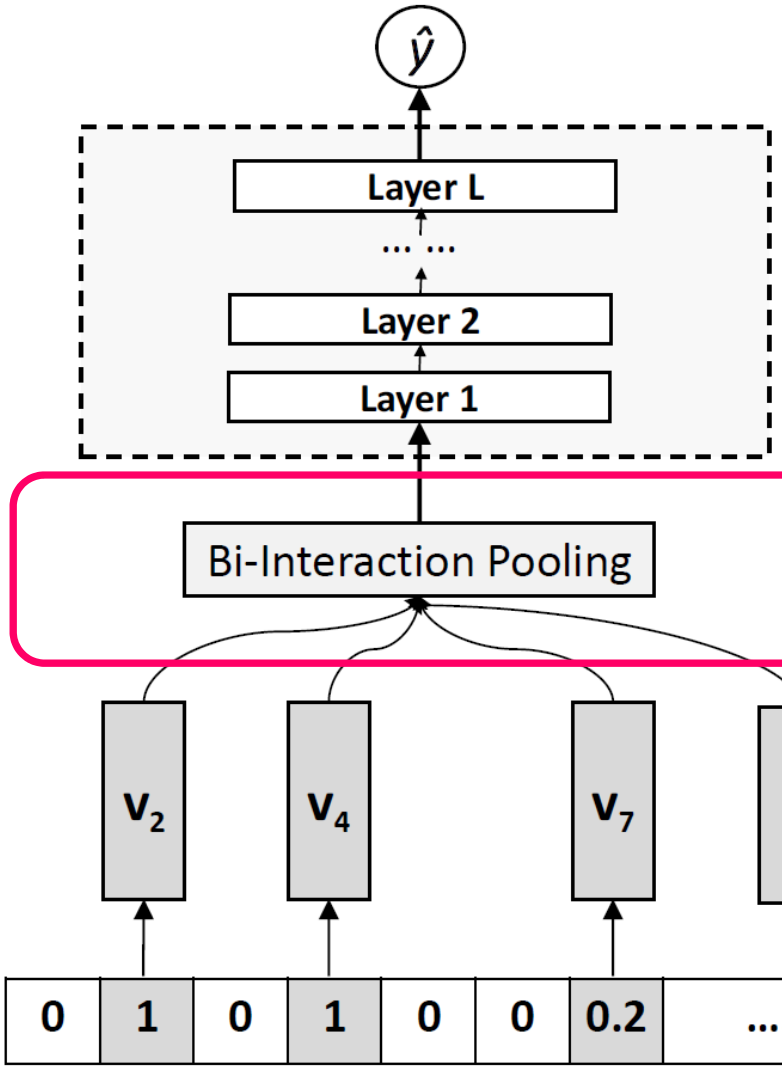


Wide & Deep Models



Neural Factorization Machines (NFM)

- Bilinear interaction (BI) layer: learn the second-order feature interactions in the low level (e.g., *female likes pink*)



Prediction

Hidden Layers

Deep layers learn high-order feature interactions only

BI Layer

Embedding Vectors

Input Feature Vector (Sparse)

Benefits of BI Layer

(1) BI Layer is pooling that converts a set of vectors to a single vector

(2) More informative than max/mean pooling and concatenation

(3) Provide a novel NN view for FM (without deep layers, it is FM)

$$f_{BI}(V_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i \mathbf{v}_i \odot x_j \mathbf{v}_j$$

$$= \frac{1}{2}\left[\left(\sum_{i=1}^{n} x_i \mathbf{v}_i\right)^2 - \sum_{i=1}^{n} (x_i \mathbf{v}_i)^2\right]$$

$$\hat{y}_{NFM}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \mathbf{h}^T \sigma_L(\mathbf{W}_L(\ldots\sigma_1(\mathbf{W}_1 f_{BI}(\mathcal{V}_x) + \mathbf{b}_1)\ldots) + \mathbf{b}_L)$$

NFM Performance

- Modeling feature interactions with embeddings is promising
- Pre-training by FM for Wide&Deep and DeepCross is useful
- NFM is better than FM with fewest additional parameters

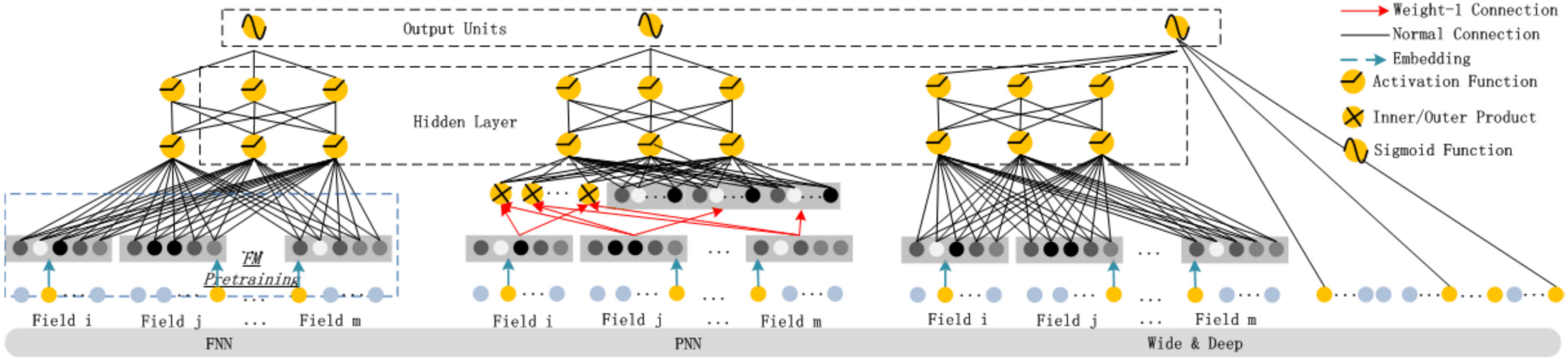
	Frappe		MovieLens	
	Factors=128		Factors=128	
Method	**Param#**	**RMSE**	**Param#**	**RMSE**
LibFM [28]	0.69M	0.3437	11.67M	0.4793
HOFM	1.38M	0.3405	23.24M	0.4752
Wide&Deep [9]	2.66M	0.3621	12.72M	0.5323
Wide&Deep (pre-train)	2.66M	0.3311	12.72M	0.4595
DeepCross [31]	4.47M	0.4025	12.71M	0.5885
DeepCross (pre-train)	4.47M	0.3388	12.71M	0.5084
NFM (1 Layer)	**0.71M**	**0.3127****	**11.68M**	**0.4557***

Pre-train means using FM embeddings as pre-training embeddings

DNN-based RecSys

- Key: learning feature interactions
  - Linear → Pair-wise feature → High-order features
  - Low- and high-order are equally important
    - 2nd order: “lunch time” AND “restaurant”
    - 3rd order: “teenager” AND “male” AND “shooting game”
  - Comparison of existing DNN-based RecSys

	No Pre-training	High-order Features	Low-order Features	No Feature Engineering
FNN	×	√	×	√
PNN	√	√	×	√
Wide & Deep	√	√	√	×
DeepFM	√	√	√	√

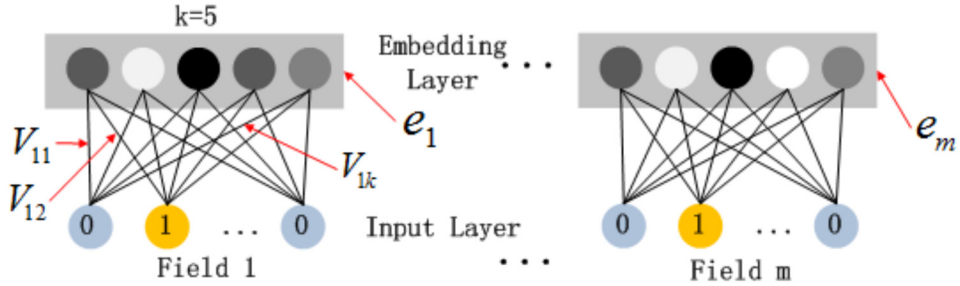


DeepFM

FM models (1,2)-order feature interactions
DNN learns higher-order feature interactions

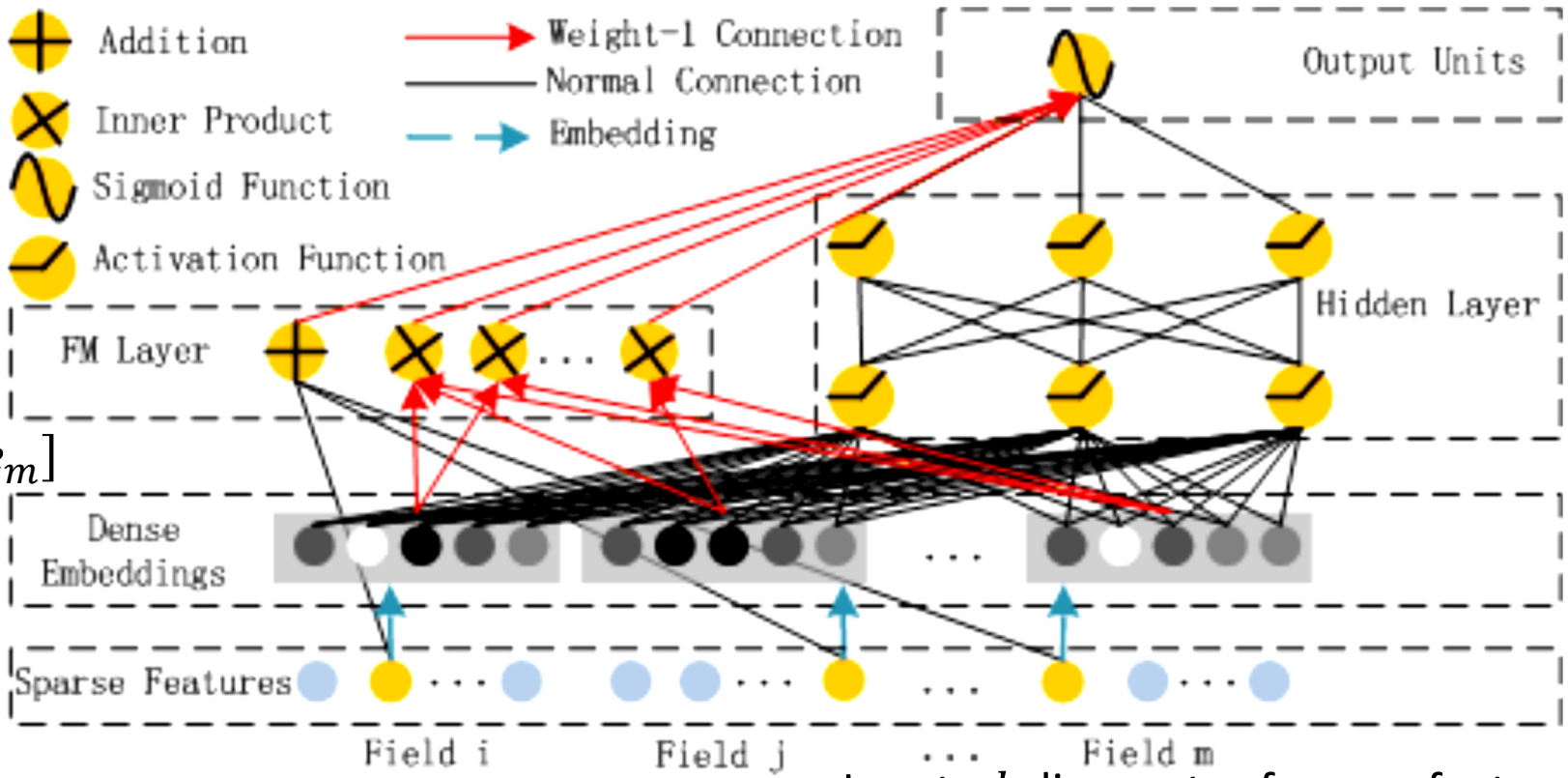
$$y_{FM} = \mathbf{w}^T\mathbf{x} + \sum_{i=1}^{d}\sum_{j=i+1}^{d} \mathbf{v}_i^{\mathrm{T}}\mathbf{v}_j x_i x_j$$

$$y_{DNN} = a^{(l+1)} = \sigma\left(\mathbf{W}^{(l)}a^{(l)} + b^{(l)}\right)$$



Convert variant-length field vectors to same-size embeddings $\boldsymbol{V}$ under FM

$$\hat{y} = \sigma(y_{FM} + y_{DNN})$$



$$a^{(0)} = [e_1, e_2, \ldots, e_m]$$

FM and DNN share dense embeddings

Input: d-dim vector from m features

DeepFM Performance on CTR Prediction

- Performance is improved by:
  - Learning feature interactions (FI) is useful
  - Learning low- and high-order FI simultaneously
  - Sharing the same feature embeddings for FI learning

	Company*		Criteo	
	AUC	LogLoss	AUC	LogLoss
LR	0.8641	0.02648	0.7804	0.46782
FM	0.8679	0.02632	0.7894	0.46059
FNN	0.8684	0.02628	0.7959	0.46350
IPNN	0.8662	0.02639	0.7971	0.45347
OPNN	0.8657	0.02640	0.7981	0.45293
PNN*	0.8663	0.02638	0.7983	0.45330
LR & DNN	0.8671	0.02635	0.7858	0.46596
FM & DNN	0.8658	0.02639	0.7980	0.45343
DeepFM	**0.8715**	**0.02619**	**0.8016**	**0.44985**

References / Code

Models	Papers	#Cites
NeuMF (NCF)	Neural Collaborative Filtering (WWW 2017) https://github.com/hexiangnan/neural_collaborative_filtering https://github.com/guoyang9/NCF	1203
Wide&Deep	Wide & Deep Learning for Recommender Systems (KDD DLRS 2016) https://github.com/jrzaurin/pytorch-widedeep	835
DeepCross	Deep Crossing: Web-Scale Modeling without Manually Crafted Combinatorial Features (KDD 2016) https://github.com/Nirvanada/Deep-and-Cross-Keras	129
NFM	Neural Factorization Machines for Sparse Predictive Analytics (SIGIR 2017) https://github.com/hexiangnan/neural_factorization_machine https://github.com/guoyang9/NFM-pyorch	318
DeepFM	DeepFM: A Factorization-Machine based Neural Network for CTR Prediction (IJCAI 2017) https://github.com/ChenglongChen/tensorflow-DeepFM https://github.com/chenxijun1029/DeepFM_with_PyTorch	326

Recommended Packages & Papers

- DeepCTR
  - https://deepctr-doc.readthedocs.io/en/latest/index.html
  - https://github.com/shenweichen/DeepCTR
- NeuRec https://github.com/wubinzzu/NeuRec
- RecQ https://github.com/Coder-Yu/RecQ
- "Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches" ACM RecSys 2019 (**Best Paper Award**)
  - https://github.com/MaurizioFD/RecSys2019_DeepLearning_Evaluation
- "Deep Learning Based Recommender System: A Survey and New Perspectives" ACM Computing Survey 2019

462 cites