Machine Learning with Graphs (MLG)

Deep RecSys (1)

From the perspective of CTR prediction

Cheng-Te Li (李政德)

Institute of Data Science

National Cheng Kung University

chengte@mail.ncku.edu.tw

Multi-Field Categorical Data

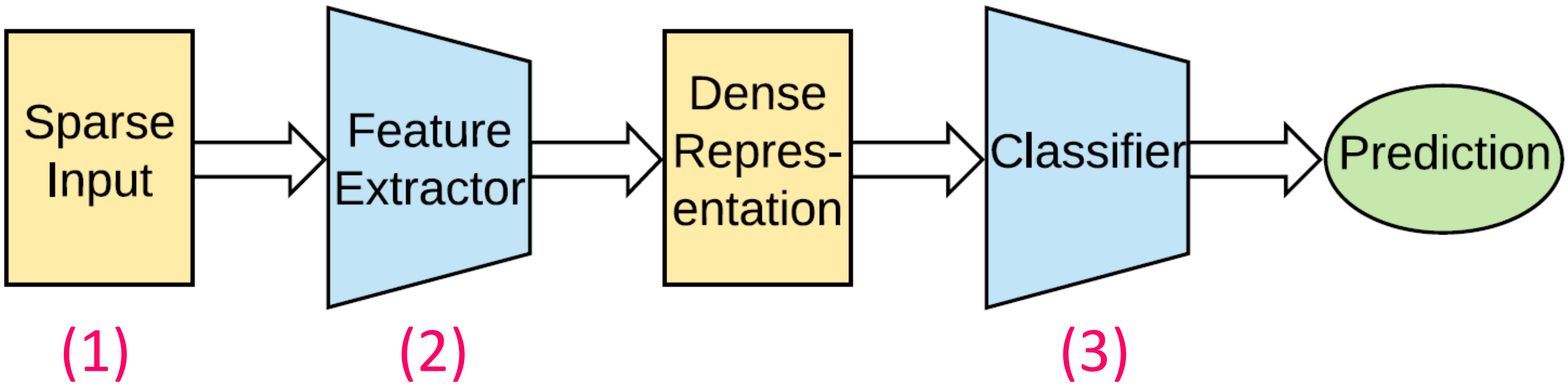
TARGET	WEEKDAY	GENDER	CITY
1	Tuesday	Male	London
0	Monday	Female	New York
1	Tuesday	Female	Hong Kong
0	Tuesday	Male	Tokyo
Number	7	2	1,000

- Use ID to represent each field
  - User field: one-hot of user ID
  - Item field: one-hot of item ID
  - Other: category ID, time, query ID, device ID, cat. attributes

$$\underbrace{[0,1,0,0,0,0,0]}_{\texttt{Weekday=Tuesday}} \quad \underbrace{[0,1]}_{\texttt{Gender=Male}} \quad \underbrace{[0,0,1,0,\ldots,0,0]}_{\texttt{City=London}}$$

- Learning feature interactions (e.g., FM) is effective
  - However, infeasible for high-dimensional sparse vectors
- DNN to learn feature interactions?
  - Assume input vector is 1000000-dim, 1st layer is 500 dim
    - #users and #items are million-scale
  - $\rightarrow$ One-layer NN needs to train 500000000 parameters
  - $\rightarrow$ Require more than 20B or 50B data instances

NN-based RecSys

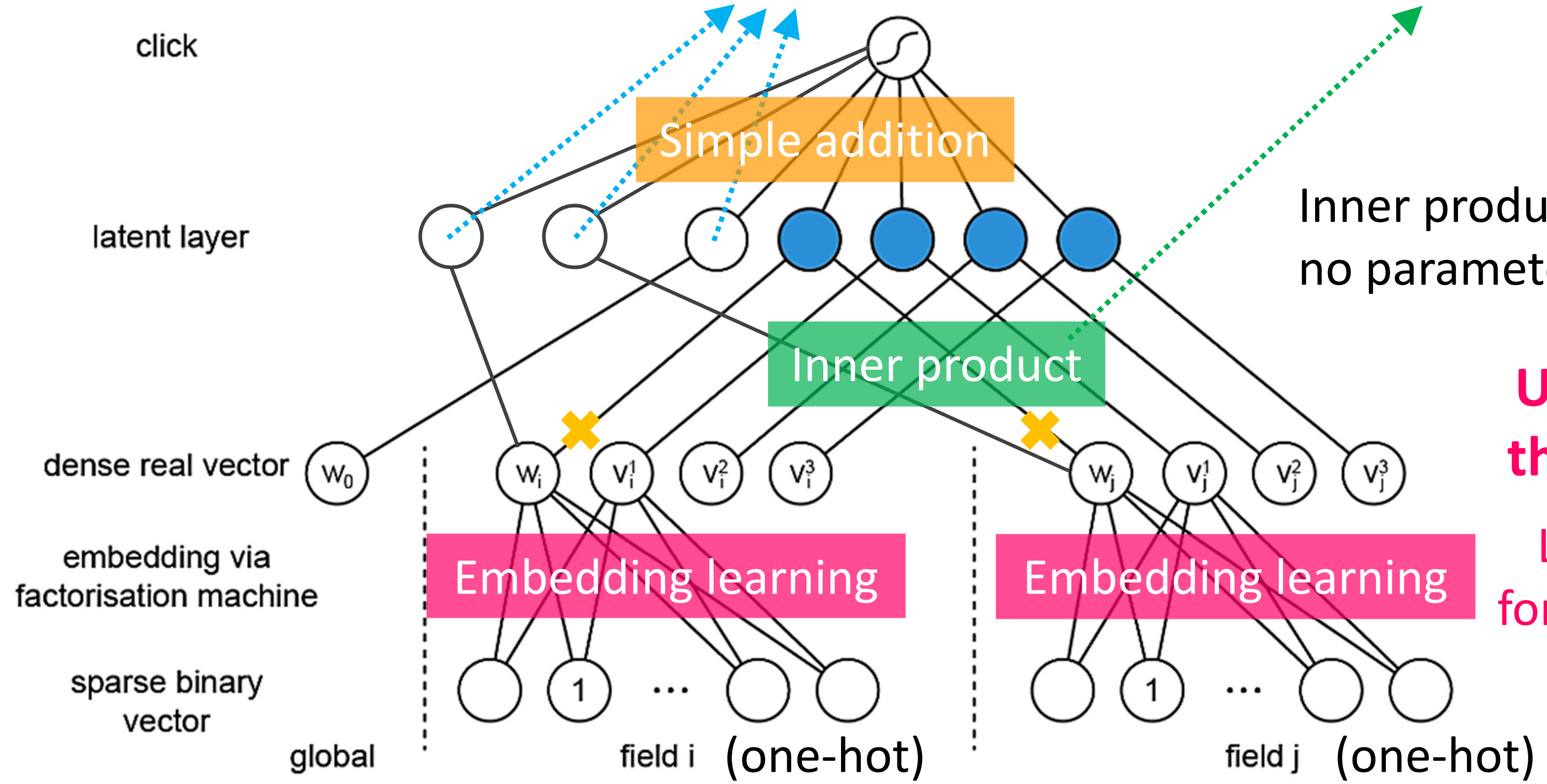


1) Multi-field categorical data
2) Convert the sparse binary input into dense representations (embeddings)
   - Weights, latent vectors
3) Search for a separation hyperplane

FM as NN

- Map high-dim features to low-dim embeddings
  - FM is a good example

$$y_{\mathrm{FM}}(\boldsymbol{x}) := \mathrm{sigmoid}\left(\underbrace{w_0 + \sum_{i=1}^{N} w_i x_i}_{\text{Logistic Regression}} + \underbrace{\sum_{i=1}^{N}\sum_{j=i+1}^{N} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle x_i x_j}_{\text{Feature Interactions}}\right)$$



Inner product layer has no parameters to learn

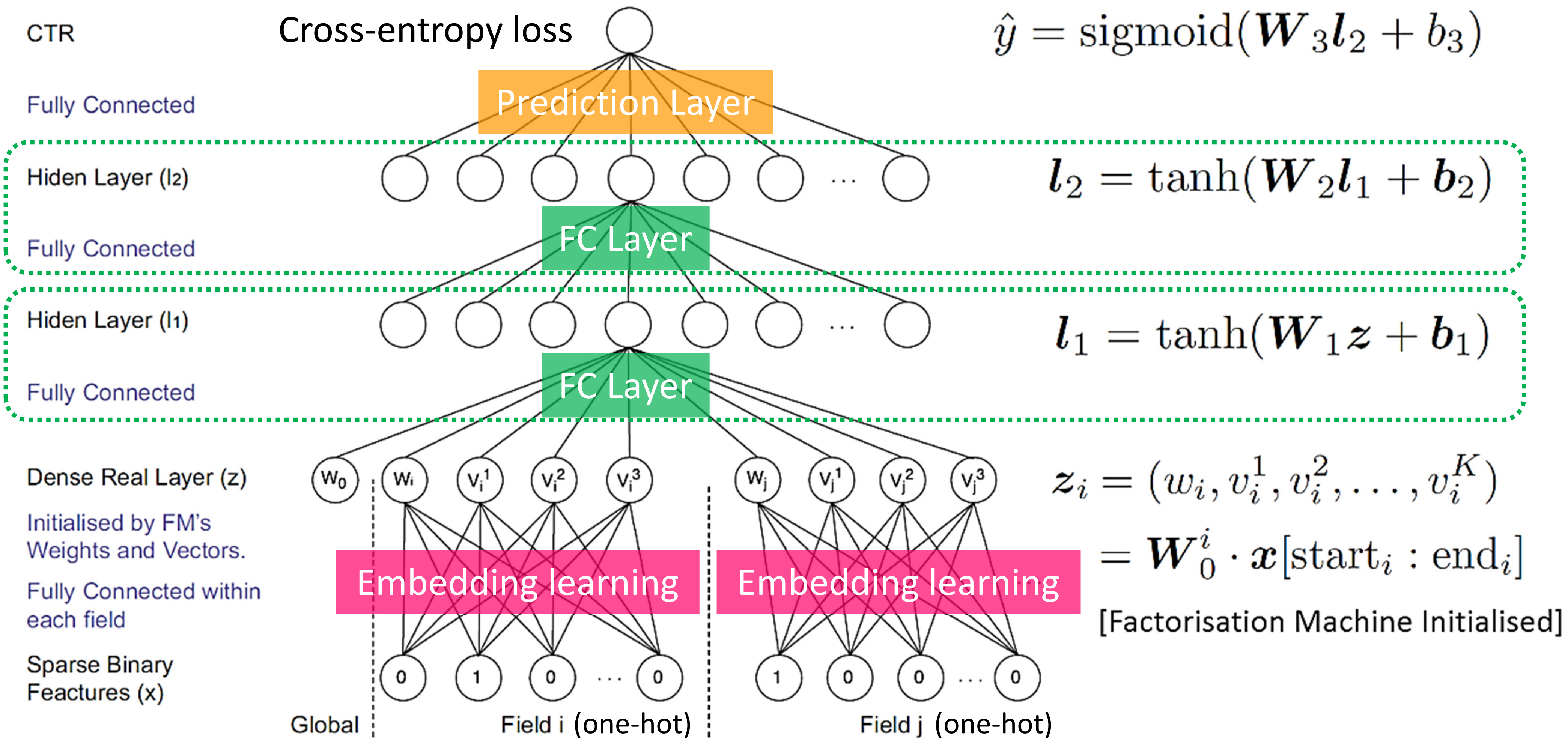
Use FM to pre-train the embedding layer

Learning embedding for each field **separately**

FM-supported Neural Networks (FNN)

- Use fully-connected layers to replace inner product layer
- Learn parameters for FC layers and prediction layer

→ Better feature representation learning (if enough training data)



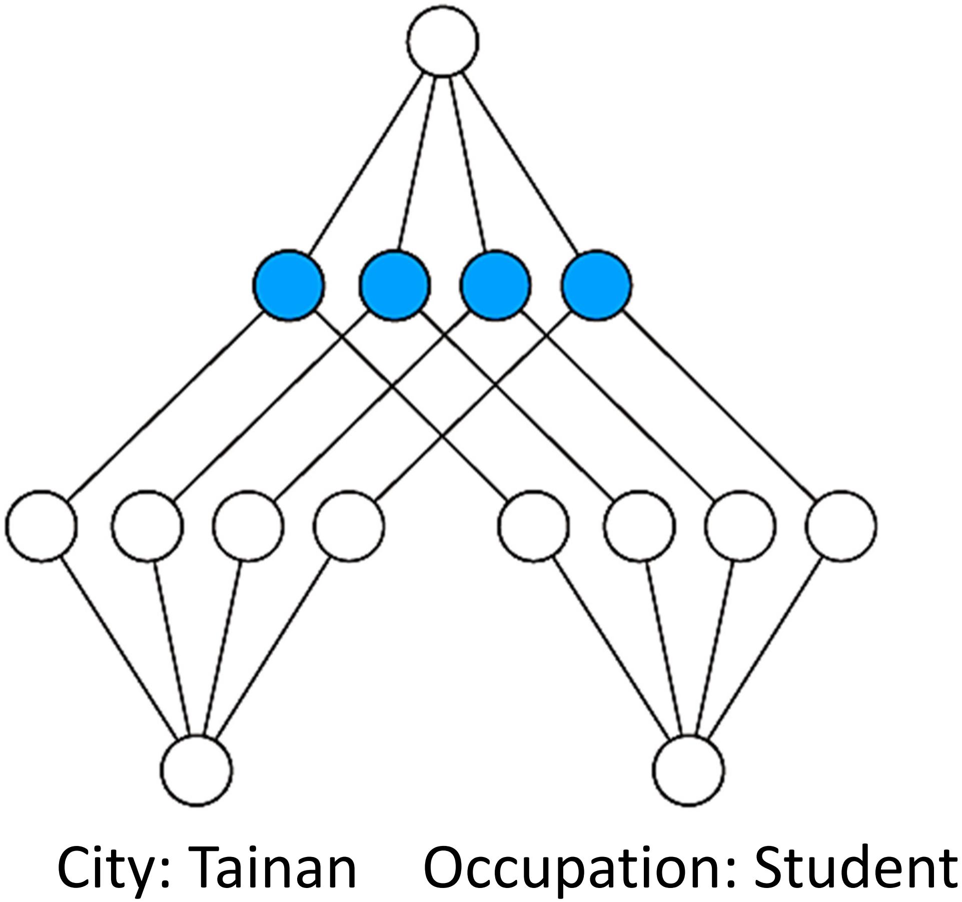
Low-dim dense embeddings significantly reduce model parameters

FNN Performance

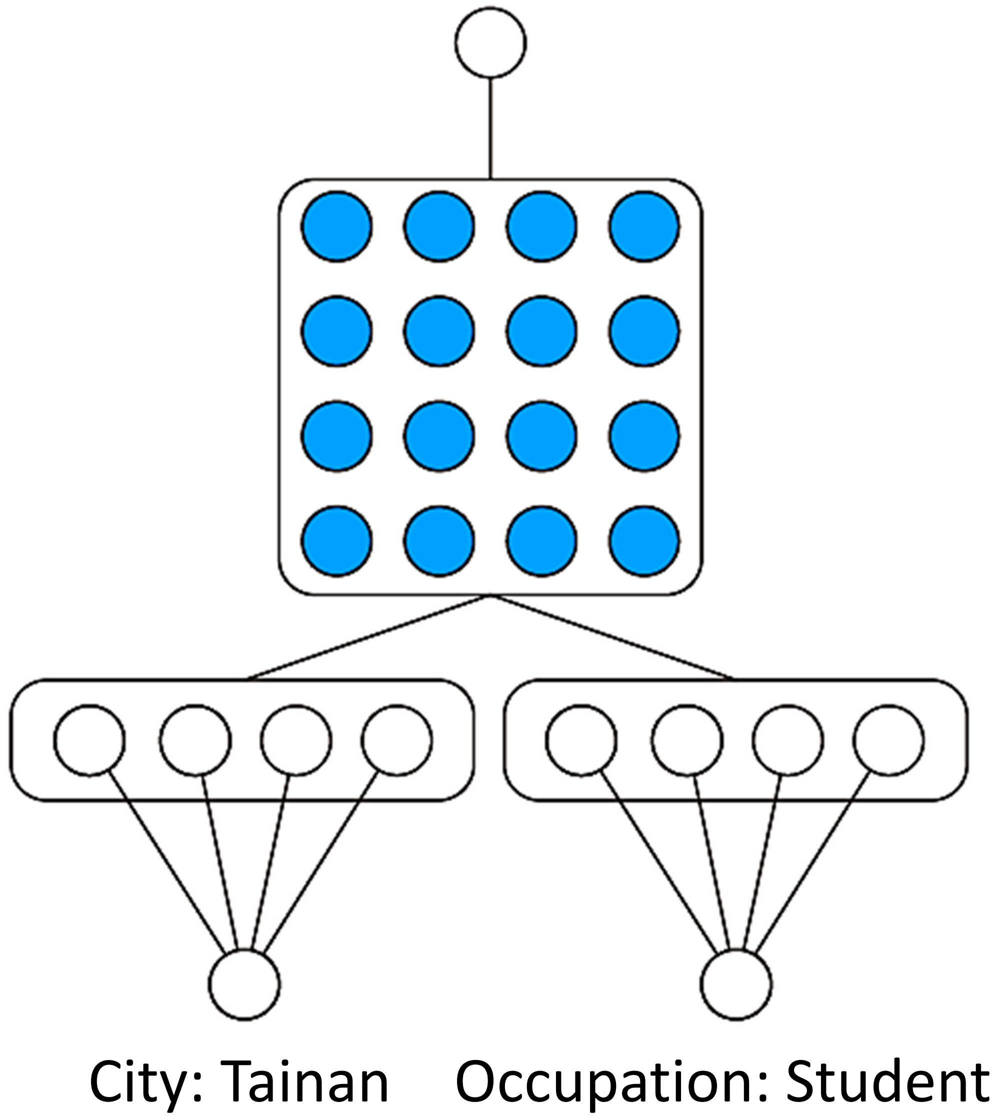
- 19.5M user-item interactions
- CTR prediction measured by AUC scores
  - $FNN > FM \sim LR$
- Can we further improve FNN?
  - Consider "multiplication" (inner product) in FM
  - FNN (NN-based methods) use "addition" to combine vectors

Campaign	LR	FM	FNN
1458	70.42%	70.21%	**70.52%**
2259	69.66%	69.73%	**69.74%**
2261	62.03%	60.97%	**62.99%**
2997	60.77%	60.87%	**61.41%**
3386	80.30%	79.05%	**80.56%**
all	68.81%	68.18%	**70.70%**

Product Operations as Feature Interactions



Inner Product Operation



Outer Product Operation

IPNN and OPNN

- Inner product
  - $n(n-1)/2$ inteactions

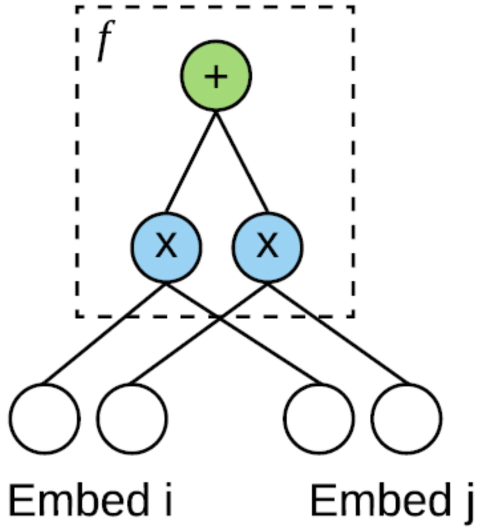
$$\mathbf{p}^\top = [\langle \mathbf{v}_1, \mathbf{v}_2 \rangle, \ldots, \langle \mathbf{v}_{n-1}, \mathbf{v}_n \rangle],$$
$$\hat{y}_{IPNN} = \text{net}([\mathbf{v}_1, \ldots, \mathbf{v}_n, p_{1,2}, \ldots, p_{n-1,n}]).$$

- Outer product
  - $d^2 \cdot \frac{n(n-1)}{2}$ interactions

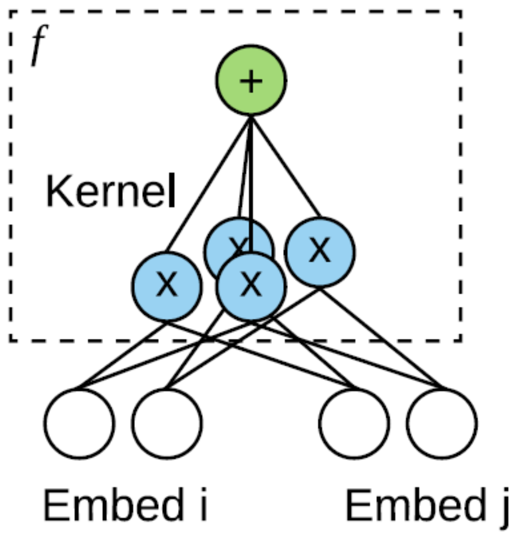
$$\mathbf{p}^\top = [\langle \mathbf{v}_1, \mathbf{v}_2 \rangle_{\phi_{1,2}}, \ldots, \langle \mathbf{v}_{n-1}, \mathbf{v}_n \rangle_{\phi_{n-1,n}}],$$
$$\hat{y}_{OPNN} = \text{net}([\mathbf{v}_1, \ldots, \mathbf{v}_n, p_{1,2}, \ldots, p_{n-1,n}]).$$

n: number of fields/features
d: embedding dimension

Inner Product



Outer Product



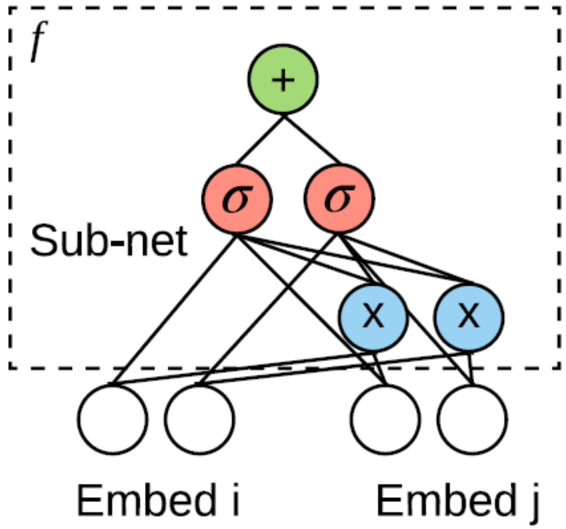
PIN: Product-network in Network

- A sub-network to learn feature interactions
  - Every field (i, j) has one sub-net
  - Two hidden layers are used
- Each sub-network takes an additional product term as input

Element-wise product

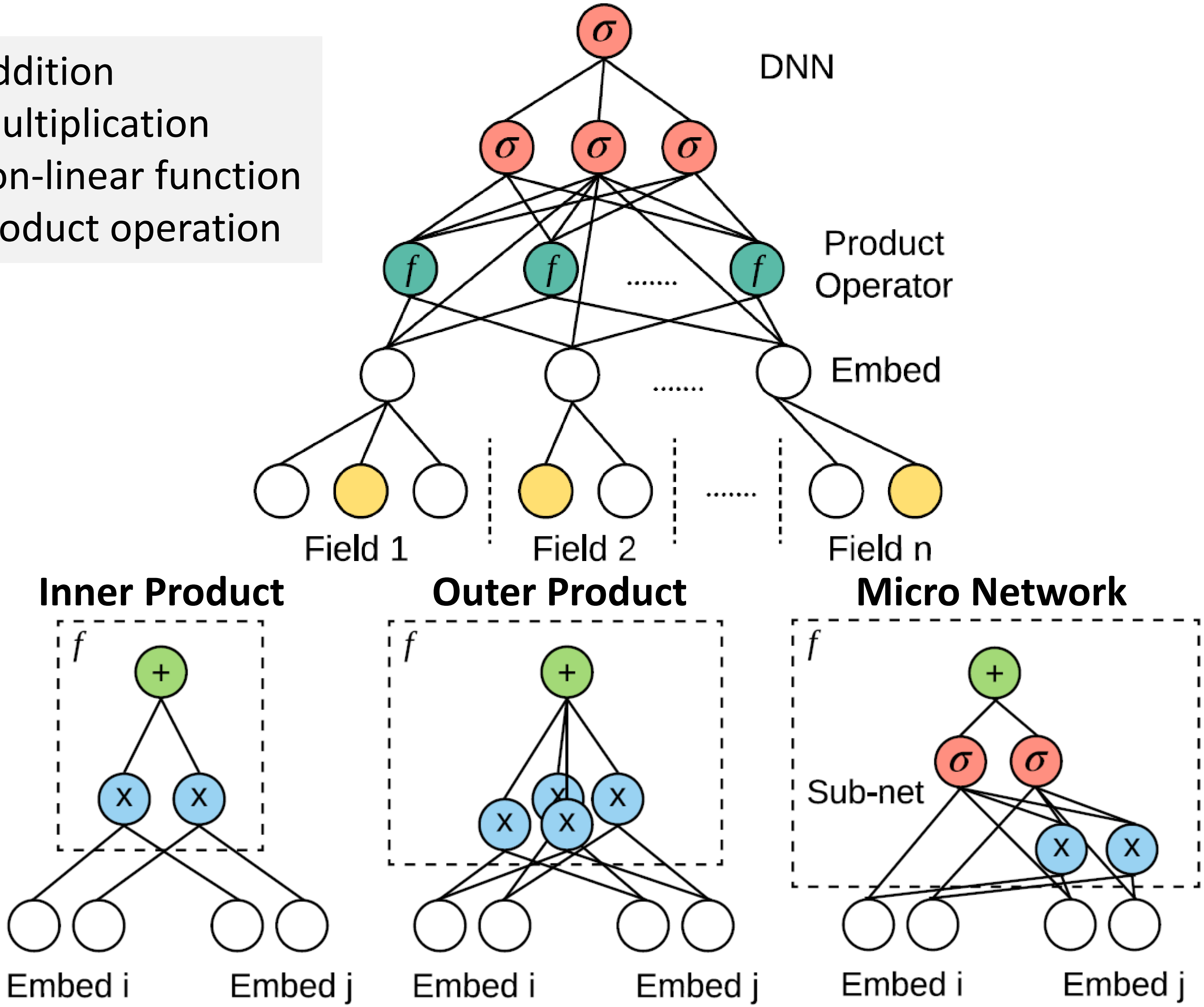
$$
\begin{aligned}
\mathbf{v}_{i,j} &= [\mathbf{v}_i, \mathbf{v}_j, \underline{\mathbf{v}_i \odot \mathbf{v}_j}], \\
f_{i,j}(\mathbf{v}_i, \mathbf{v}_j) &= \sigma(\mathbf{v}_{i,j}^{\top} \mathbf{w}_{i,j}^{1} + \mathbf{b}_{i,j}^{1})^{\top} \mathbf{w}_{i,j}^{2} + \mathbf{b}_{i,j}^{2}, \\
\mathbf{p}_{i,j} &= f_{i,j}(\mathbf{v}_i, \mathbf{v}_j), j \neq i, \\
\hat{y}_{PIN} &= \mathrm{net}([\mathbf{p}_{1,2}, \ldots, \mathbf{p}_{n-1,n}]),
\end{aligned}
$$

Micro Network



Product-based Neural Networks (PNN)

"+": addition
"×": multiplication
"σ": non-linear function
"f": product operation



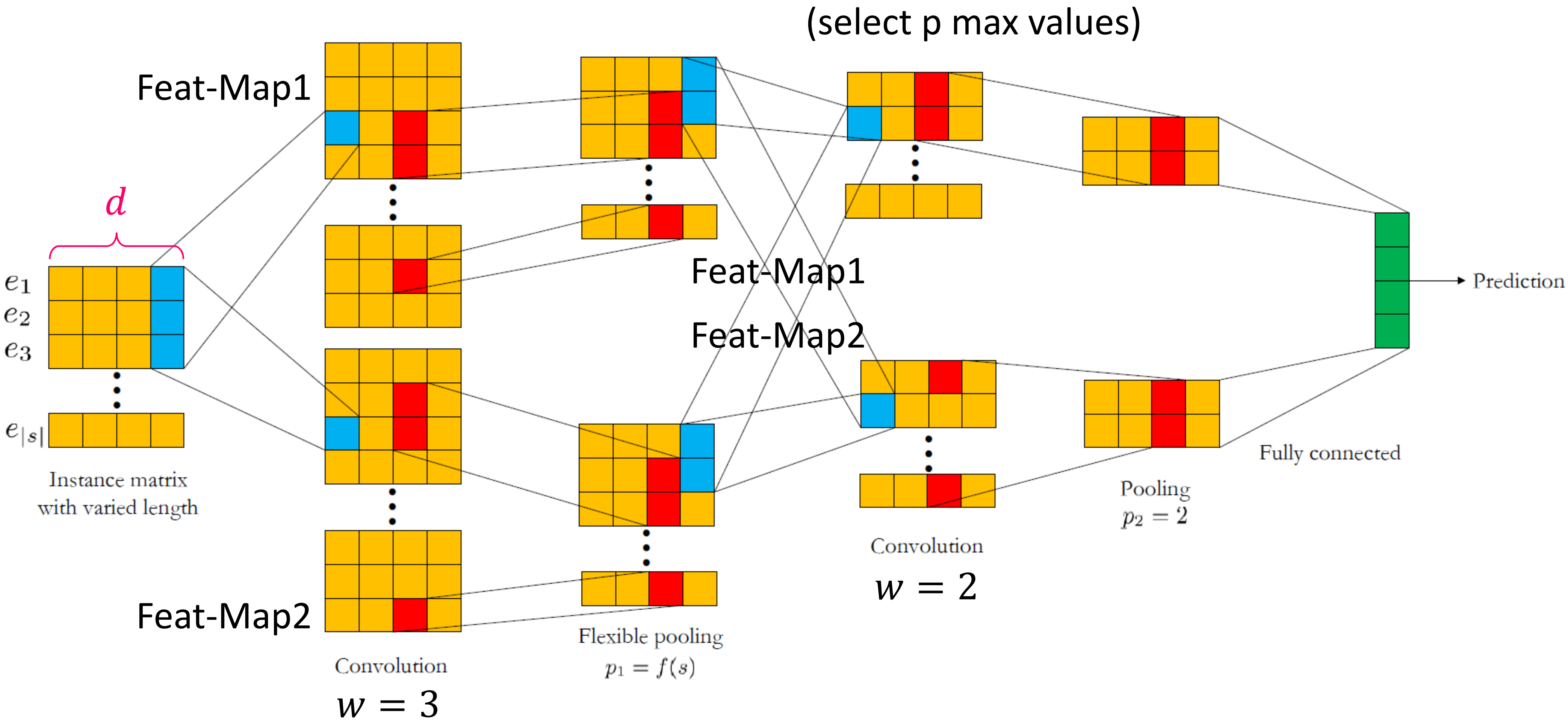
Performance of PNN

- PNN (especially PIN) generates the best performance

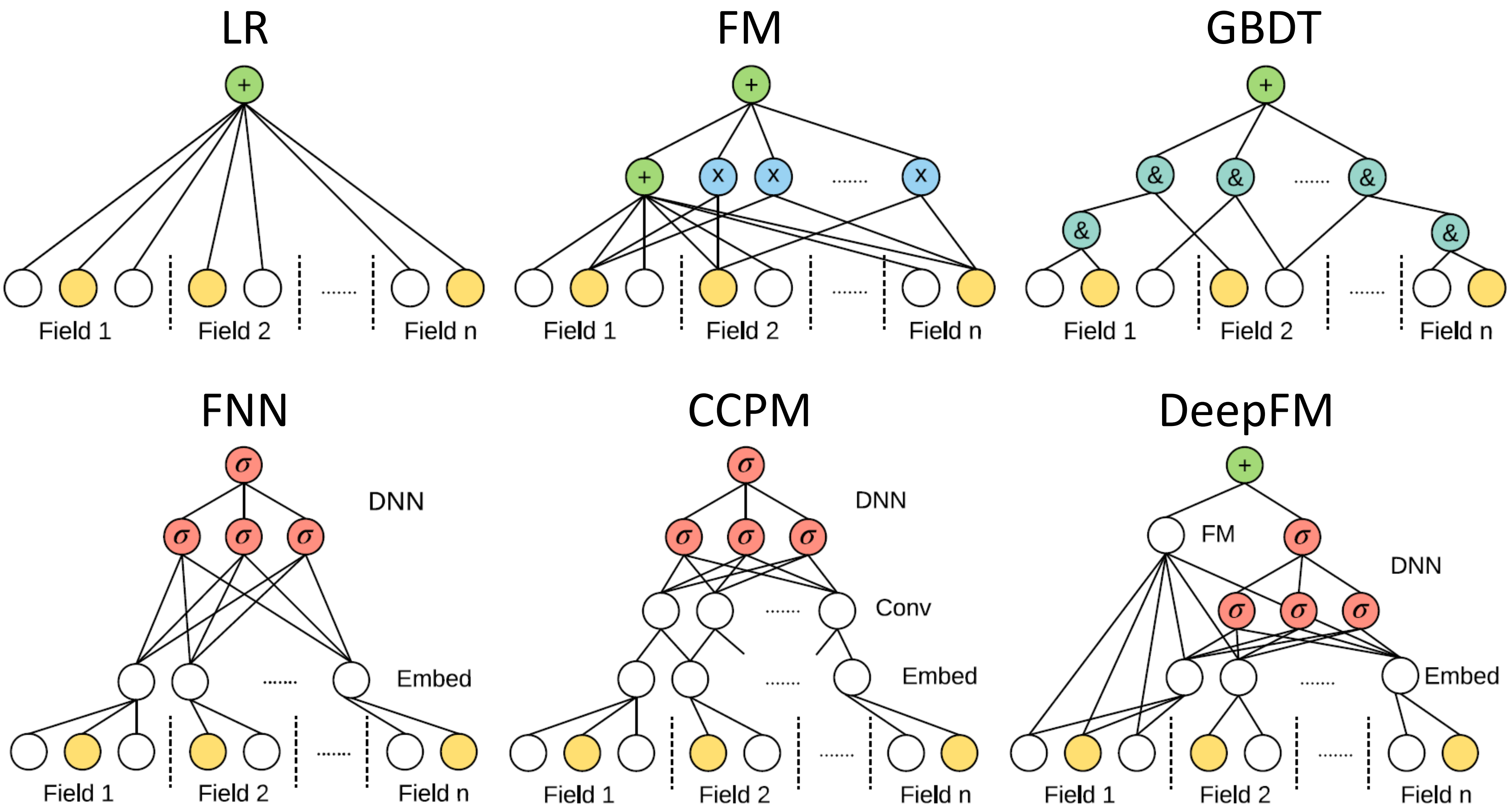
	Criteo		Avazu		iPinYou		Huawei	
Model	AUC (%)	Log Loss	AUC (%)	Log Loss	AUC (%)	Log Loss	AUC (%)	Log Loss
LR	78.00	0.5631	76.76	0.3868	76.38	0.005691	86.40	0.02648
GBDT	78.62	0.5560	77.53	0.3824	76.90	0.005578	86.45	0.02656
FM	79.09	0.5500	77.93	0.3805	77.17	0.005595	86.78	0.02633
FFM	79.80	0.5438	78.31	0.3781	76.18	0.005695	87.04	0.02626
CCPM	79.55	0.5469	78.12	0.3800	77.53	0.005640	86.92	0.02633
FNN	79.87	0.5428	78.30	0.3778	77.82	0.005573	86.83	0.02629
AFM	79.13	0.5517	78.06	0.3794	77.71	0.005562	86.89	0.02649
DeepFM	79.91	0.5423	78.36	0.3777	77.92	0.005588	87.15	0.02618
KFM	79.85	0.5427	78.40	0.3775	76.90	0.005630	87.00	0.02624
NIFM	79.80	0.5437	78.13	0.3788	77.07	0.005607	87.16	0.02620
IPNN	80.13	0.5399	78.68	0.3757	78.17	0.005549	87.27	0.02617
OPNN	80.17	0.5394	78.71	0.3756	78.21	0.005563	87.28	0.02617
PIN	**80.21**	**0.5390**	**78.72**	**0.3755**	**78.22**	**0.005547**	**87.30**	**0.02614**

Convolutional Click Prediction Model (CCPM)

- Each instance s (user, query, item, time, cat., device, etc) has $|s|$ fields/elements → generating d-dim embedding layer $e_1, e_2, \ldots, e_{|s|}$
  - An instance can be an item of a user, or a sequence of items of a user
- Use **1-D convolution** ($w \times d$) with ***p*-max pooling** to learn features



NN-based RecSys



"+": addition, "×": multiplication, "&": feature combination

References / Code

Models	Papers	#Cites
FNN	Deep Learning over Multi-field Categorical Data: A Case Study on User Response Prediction (ECIR 2016) https://github.com/wnzhang/deep-ctr	**164**
IPNN OPNN	Product-based Neural Networks for User Response Prediction (ICDM 2016) https://github.com/Atomu2014/product-nets	**135**
PIN	Product-Based Neural Networks for User Response Prediction over Multi-Field Categorical Data (ACM TOIS 2018) https://github.com/Atomu2014/product-nets-distributed	**30**
CCPM	A Convolutional Click Prediction Model (CIKM 2015) https://github.com/Hirosora/LightCTR	**60**