DeepFM: A Factorization-Machine based Neural Network for CTR Prediction

김연아

2021.05.17

Click-through rate

- **Click-through rate**
  - The probability a user will click on a recommended item

Recomm. Items	CTR	Rank
Item 1	0.65	2
Item 2	0.22	3
Item 3	0.94	1

Implicit feature interactions

- **Learn implicit feature interactions behind user click behaviors**

- **Order of interactions**

Examples (apps market)

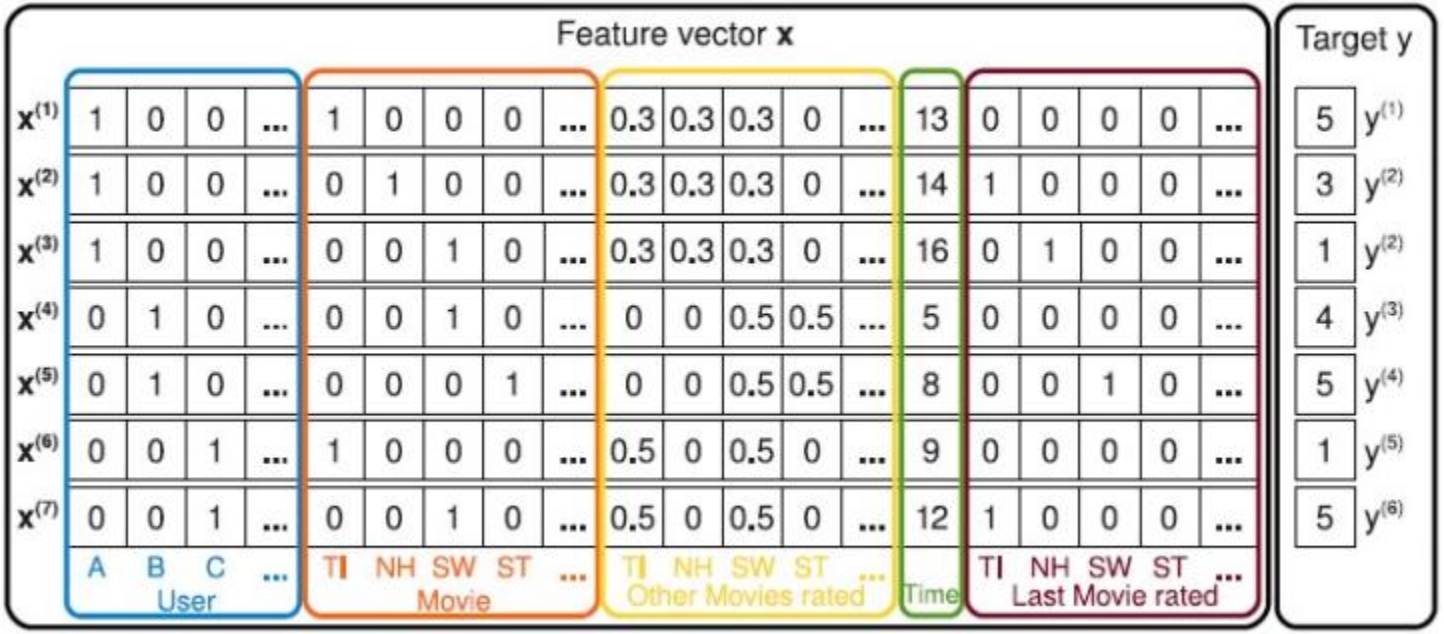
- "people often download apps for **food delivery** at **meal-time**" → order-2 interaction
  - Interaction between app category and time-stamp

- "**male** **teenagers** like **shooting games and RPG games**" → order-3 interaction
  - Interaction between app category, user gender, and age

→ Considering **low- and high-order feature interactions** simultaneously brings additional improvement in recommendation

Factorization Machines (FM)

- Pairwise feature interactions as inner product of latent vectors between features and show very promising results
  - High-order feature interaction → order-2 feature interactions (complexity problem)



Implicit feature interactions

- **Learn implicit feature interactions behind user click behaviors**

- **Order of interactions**

Examples (apps market)

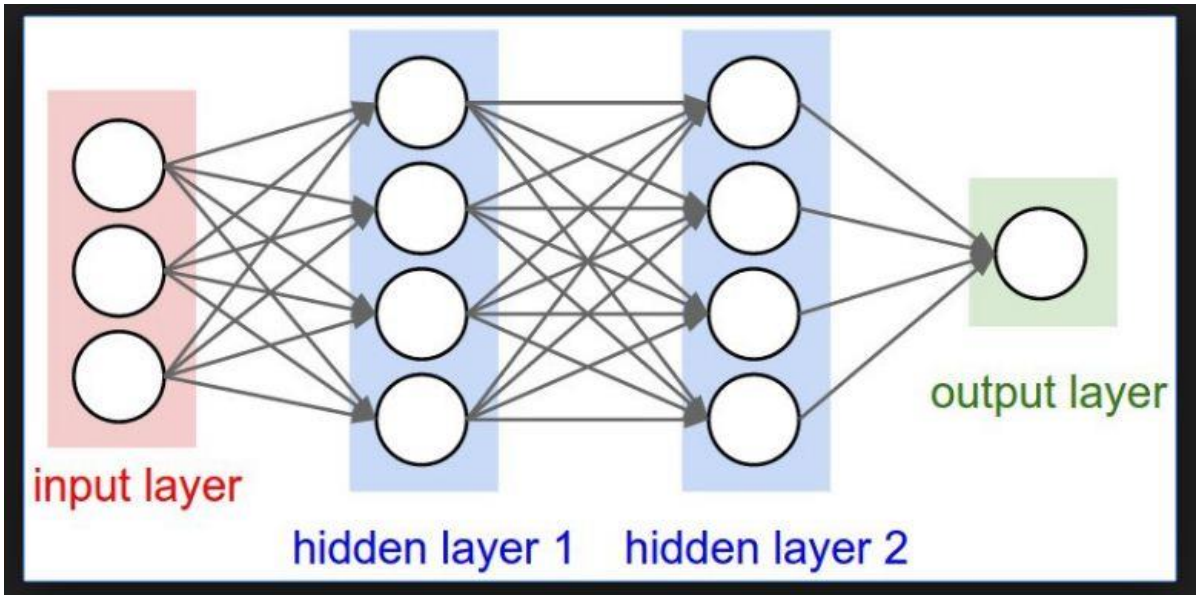
- "people often purchase **diaper** and **beer** together" → order-2 interaction
- Diaper and beer…?

&

→ Most features hidden in data and difficult to identify needs to be **captured automatically** by **machine learning**

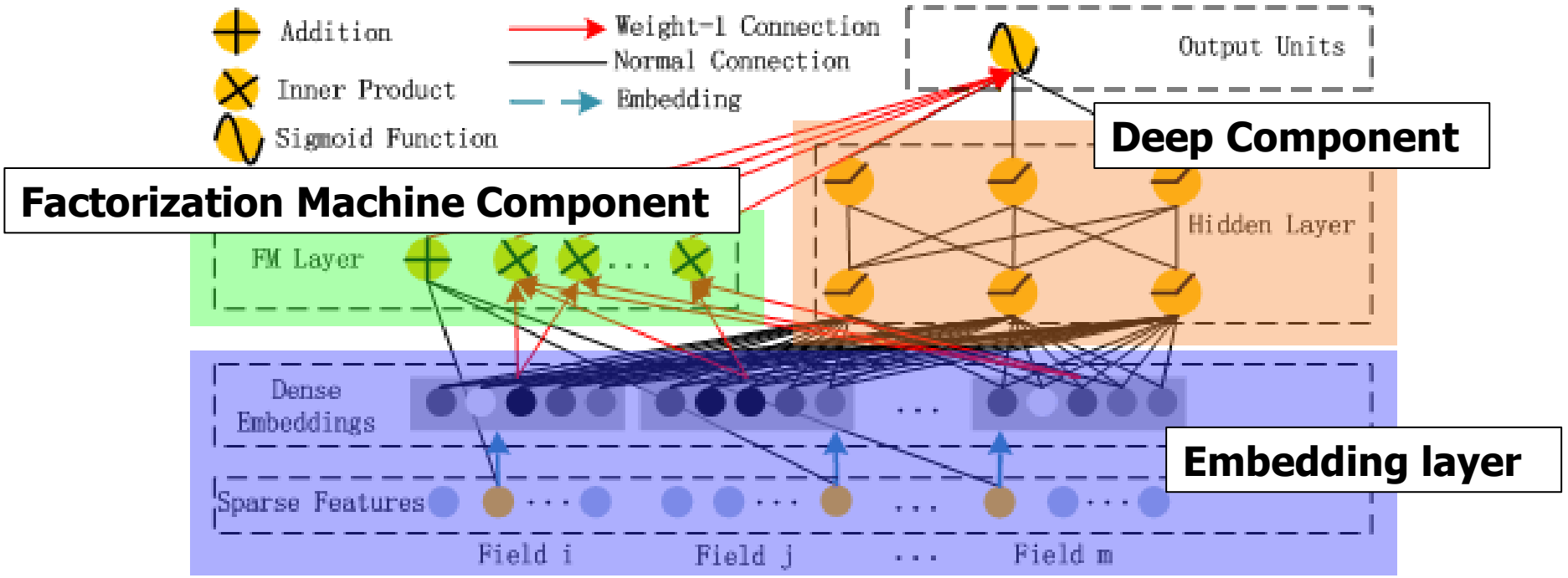
Deep Neural Networks

- Powerful approach to learn feature representation
- Learn sophisticated feature interactions



DeepFM

- **Factorization-Machine based neural network**
  - Learn both low- and high-order feature interactions
  - Without any feature engineering of the input

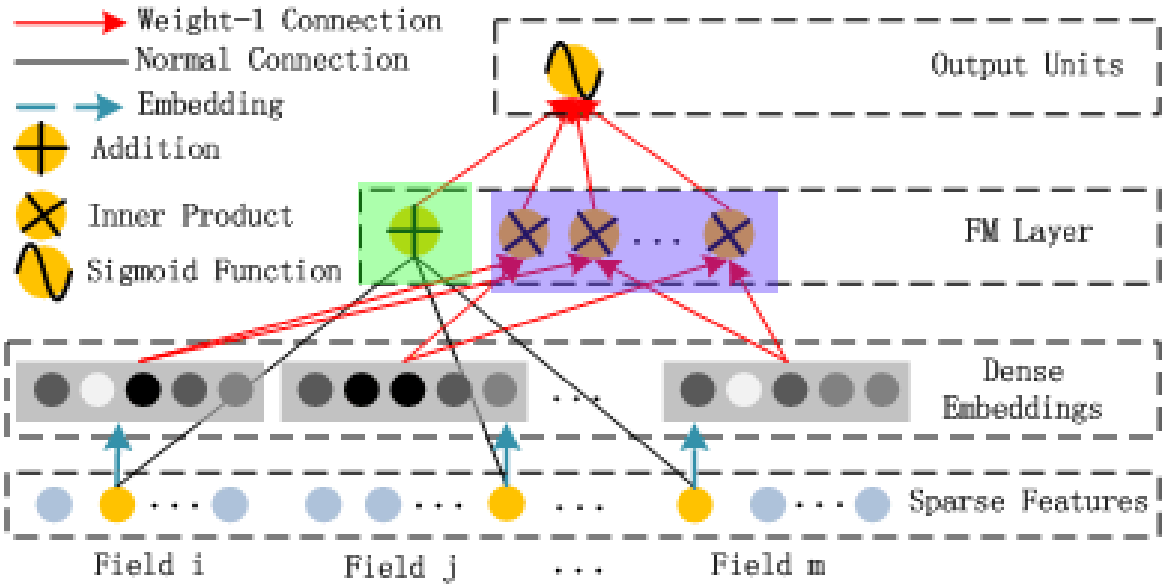


Factorization Machine Component

Inner product unit: order-2 feature interactions

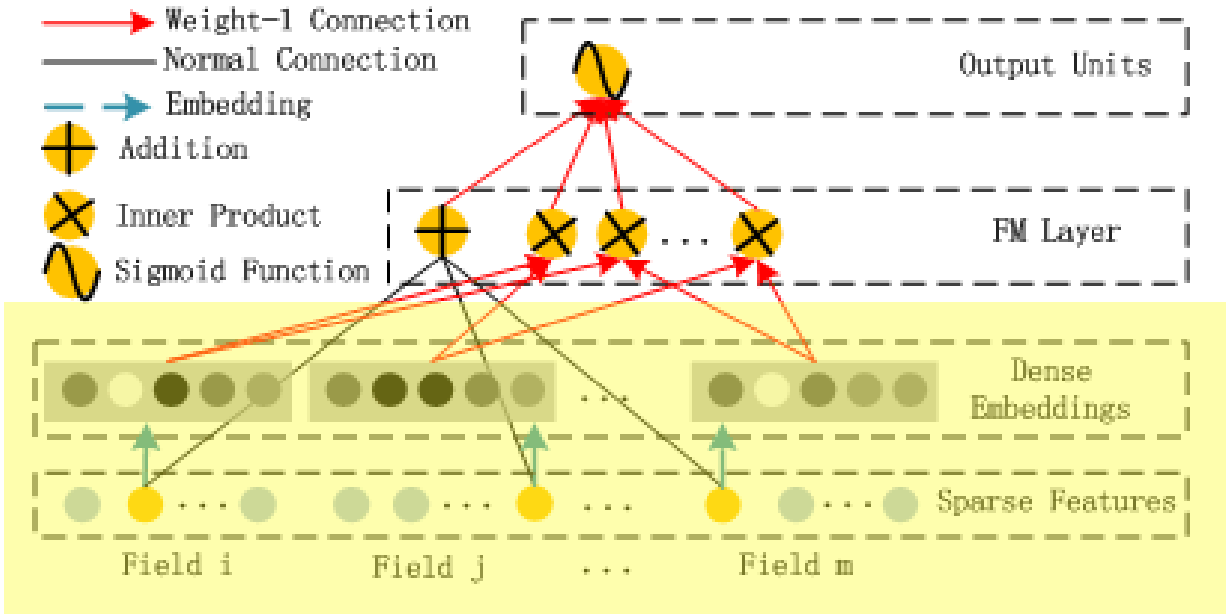
$$y_{FM} = \langle w, x \rangle + \sum_{j_1=1}^{d} \sum_{j_2=j_1+1}^{d} \langle V_i, V_j \rangle x_{j_1} \cdot x_{j_2}, \qquad (2)$$

Addition unit:
Order-1 features



Inputs

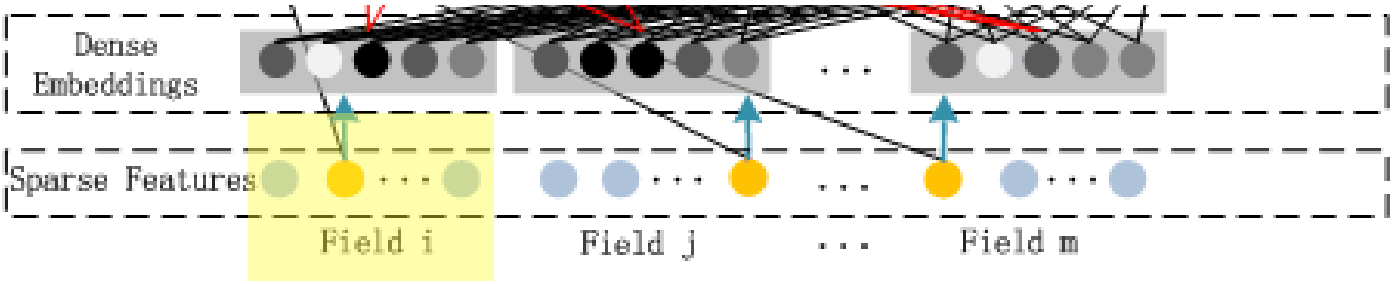
- **Input of CTR prediction**
  - The raw feature input vector for CTR prediction is usually…
    - highly sparse
    - super high-dimensional
    - categorical-continuous-mixed
    - grouped in fields (gender, location, age, …)



?

Inputs

- **Input of CTR prediction**
  - The raw feature input vector for CTR prediction is usually…
    - highly sparse
    - super high-dimensional
    - categorical-continuous-mixed
    - Grouped in fields (gender, location, age, …)



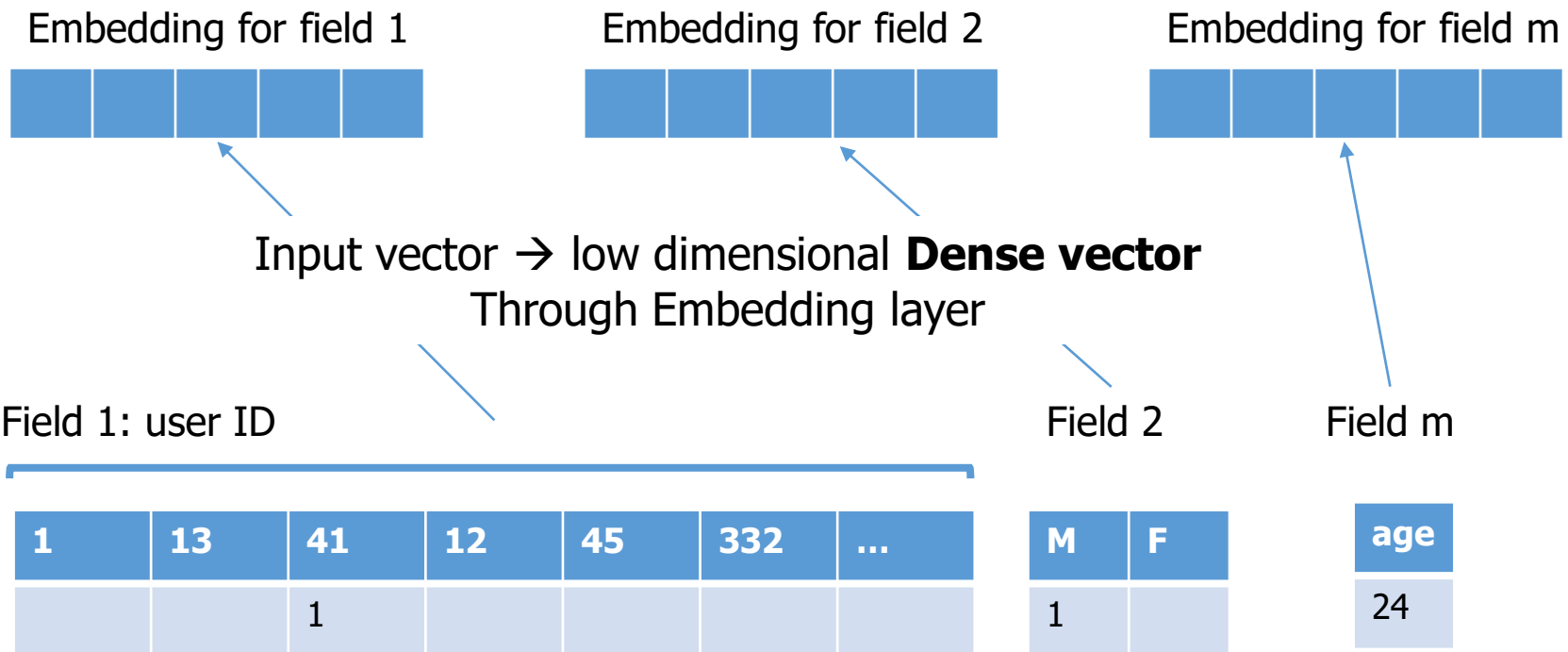
Ex)For an app store of **billion** users…

Field 1: user ID

1	13	41	12	45	332	…	1253	3512	2333
		1							

Embedding layer

- **Features of embedding layer**
  - While the lengths of different input field vectors can be different, their embeddings are of the same size (k).
  - The latent feature vectors (V) in FM now server as network weights which are learned and used to compress the input field vectors to the embedding vectors.

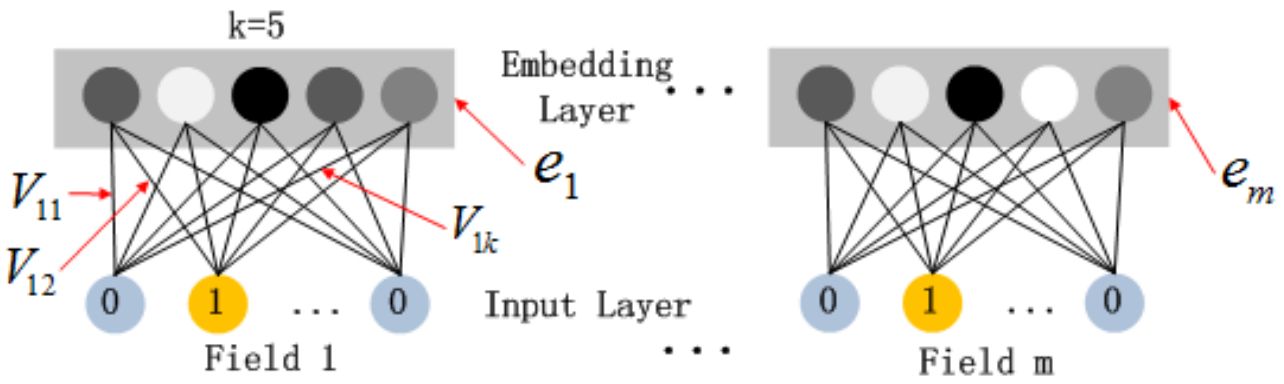


Embedding layer

- **Features of embedding layer**
  - While the lengths of different input field vectors can be different, their embeddings are of the same size (k).
  - The latent feature vectors (V) in FM now server as network weights which are learned and used to compress the input field vectors to the embedding vectors.

Output of the embedding layer

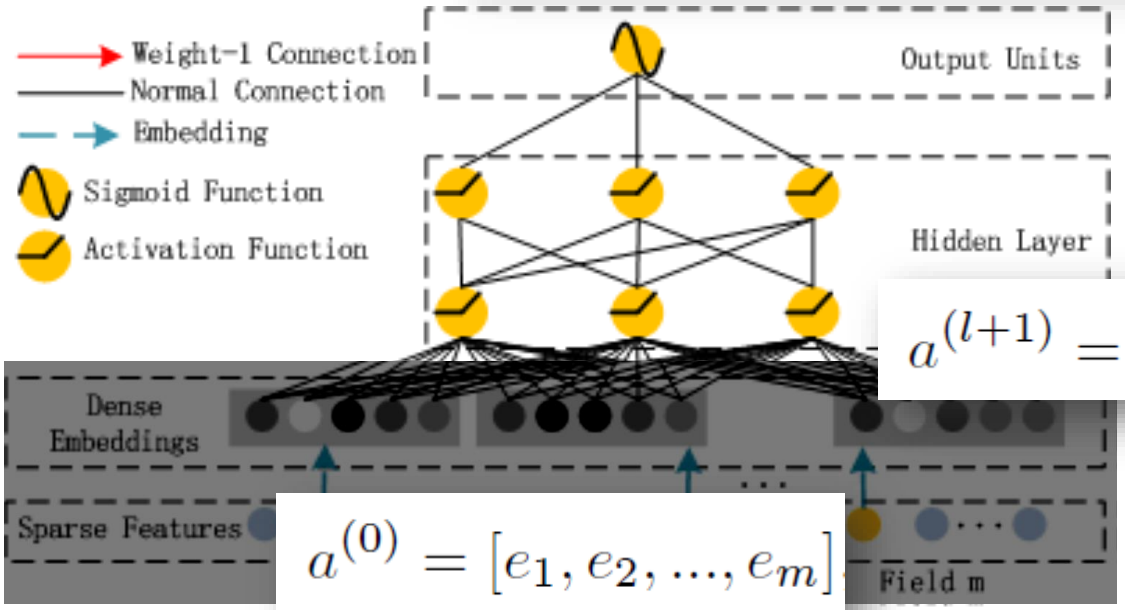
$$a^{(0)} = [e_1, e_2, \ldots, e_m]$$



Deep Component

- **Feed-forward neural network**

$$y_{DNN} = \sigma(W^{|H|+1} \cdot a^H + b^{|H|+1})$$



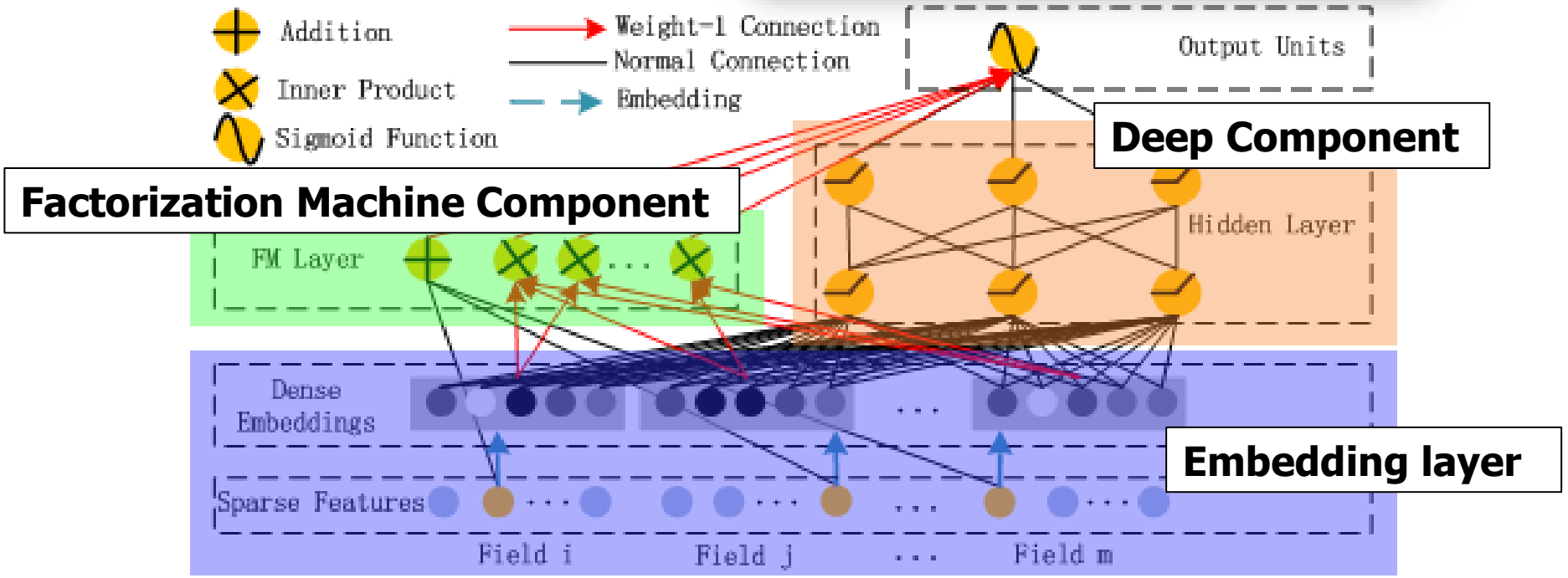
$$a^{(l+1)} = \sigma(W^{(l)} a^{(l)} + b^{(l)})$$

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

DeepFM

- **Factorization-Machine based neural network**

Predicted CTR $\hat{y} = sigmoid(y_{FM} + y_{DNN})$



Previous models

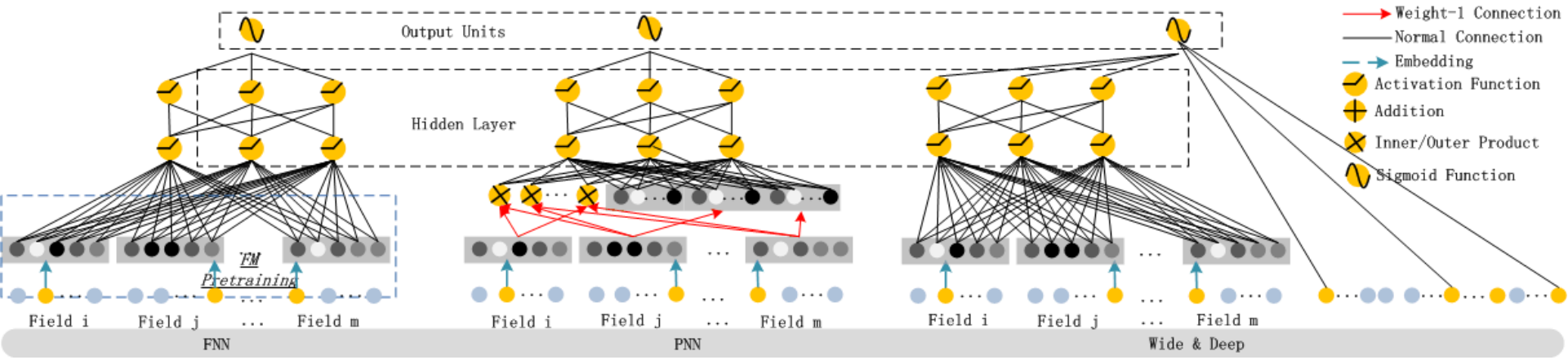
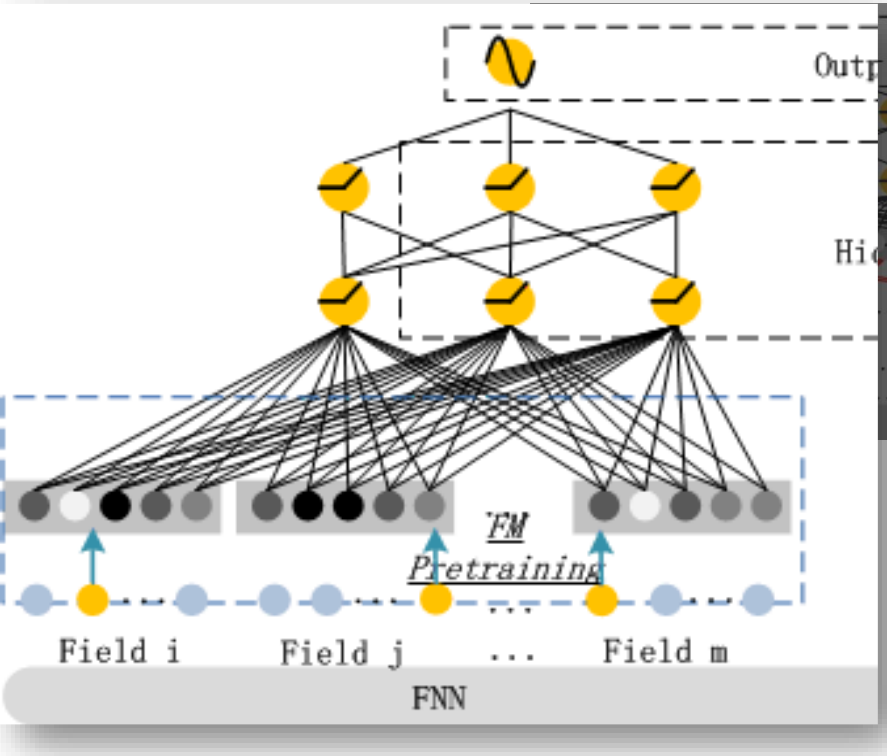


Table 1: Comparison of deep models for CTR prediction

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

Previous models



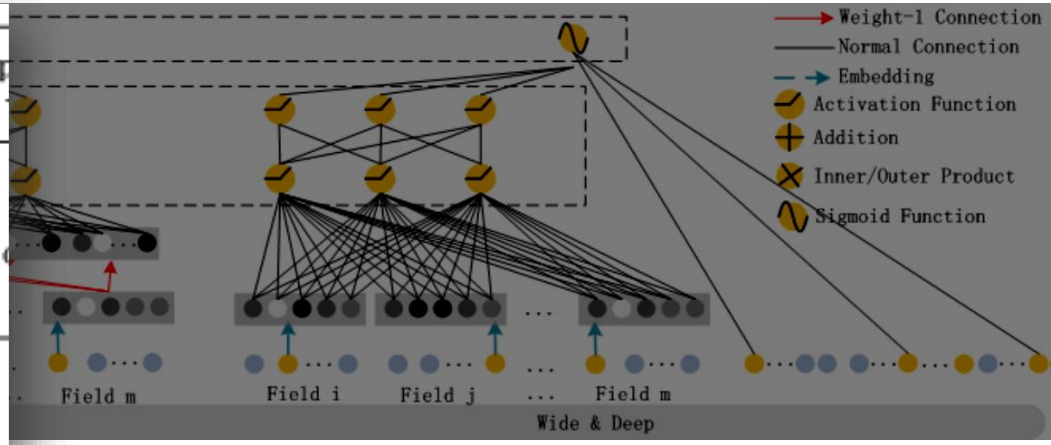


Table 1: Comparison of deep models for CTR prediction

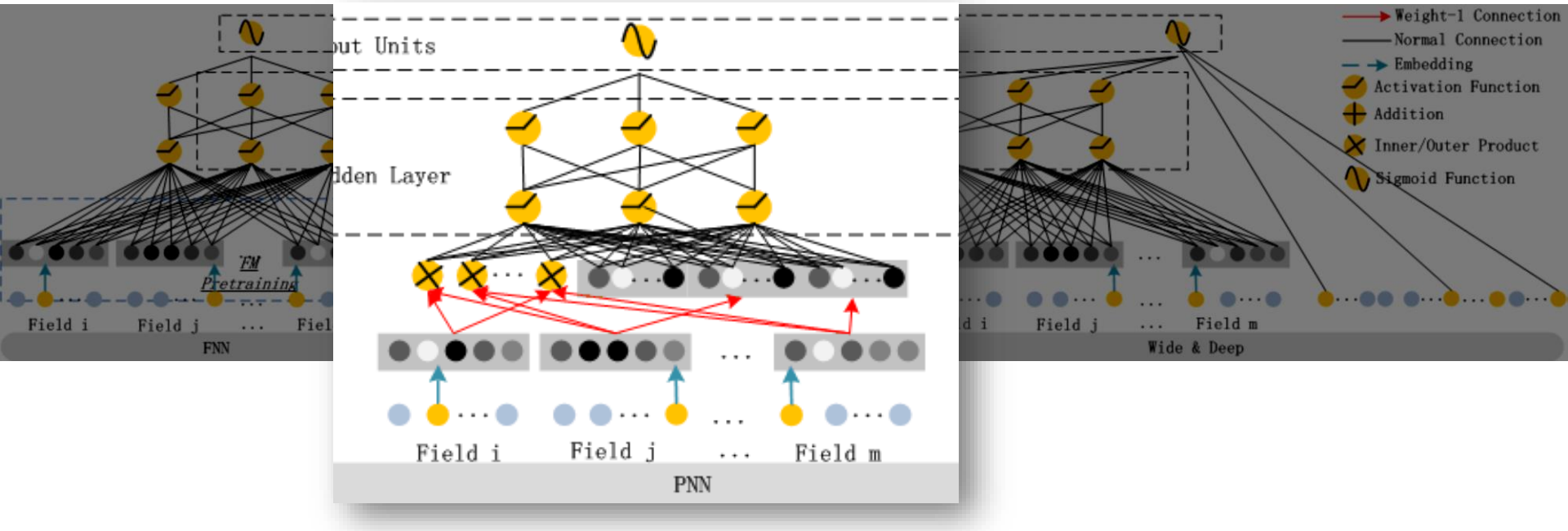
	No Pre-training	High-order Features	Low-order Features	No Feature Engineering
FNN	×	√	×	√
PNN	√	√	×	√
Wide & Deep	√	√	√	×
DeepFM	√	√	√	√

Factorization-machine supported Neural Network (FNN)

- Neural network based → high-order interaction modeling

- Need pre-training with FM
- Hard to capture low-order interaction

Previous models



Product-based Neural Network (PNN)

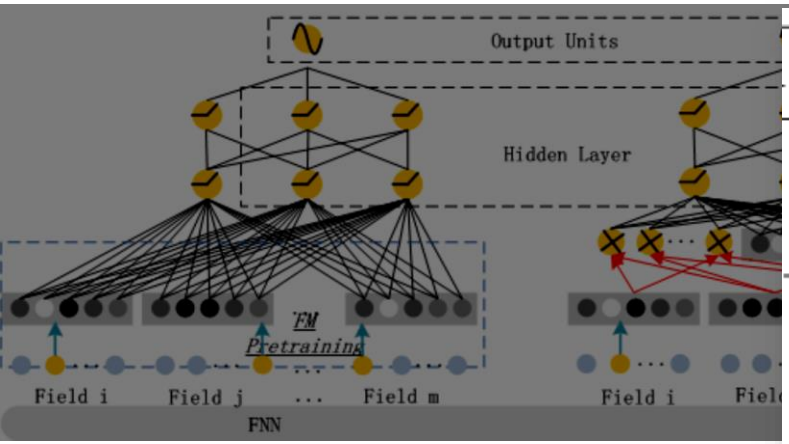
- Neural network based → high-order interaction modeling

- Hard to capture low-order interaction

Table 1: Comparison of deep models for CTR prediction

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

Previous models



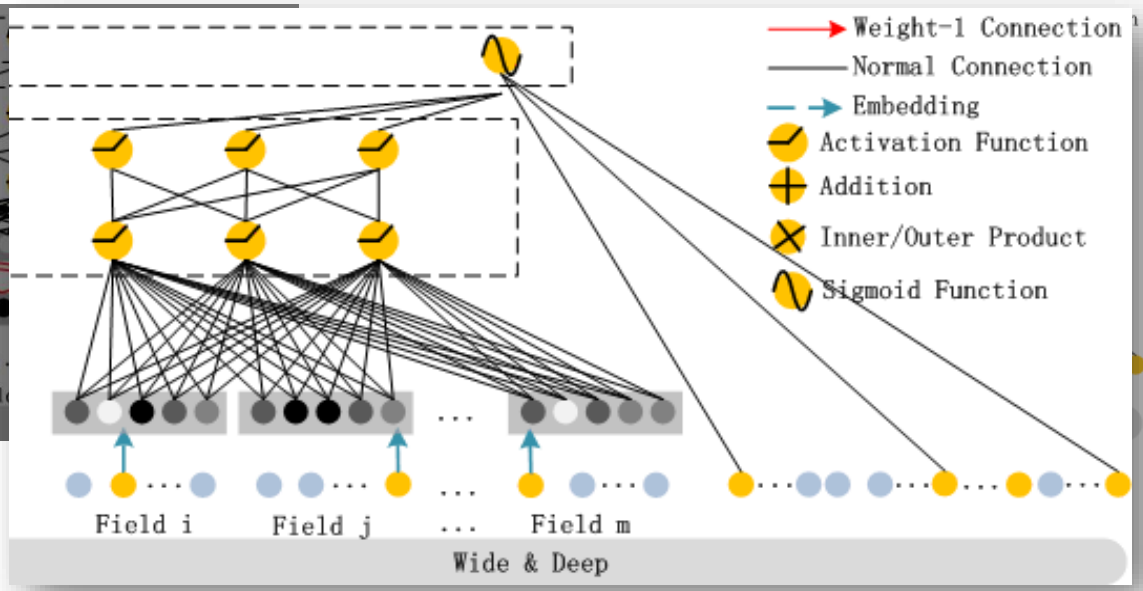


Table 1: Comparison of deep models for CTR prediction

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

Wide & Deep

- Combines a linear (wide) model for low-order interaction and a deep model for high-order interaction

- Require feature engineering

Experiments

- **Datasets**
  - **Criteo Dataset**
    45 million users' click records
    13 continuous, 26 categorical features
  - **Company dataset (app store)**
    REAL industrial CTR prediction
    7 consecutive days of users' click records & next 1 day for testing
    App features, user features, context features

Experiments

- **Time Efficiency comparison**

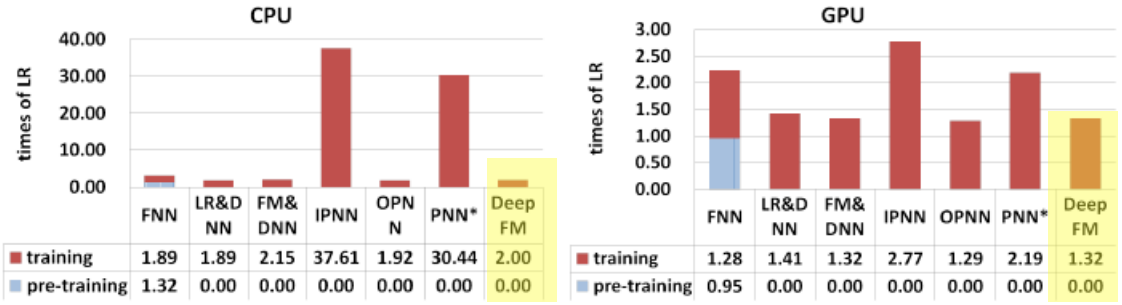


Figure 6: Time comparison.

- **Effectiveness Comparison**

Table 2: Performance on CTR prediction.

	Company∗		Criteo	
	AUC	LogLoss	AUC	LogLoss
LR	0.8640	0.02648	0.7686	0.47762
FM	0.8678	0.02633	0.7892	0.46077
FNN	0.8683	0.02629	0.7963	0.45738
IPNN	0.8664	0.02637	0.7972	0.45323
OPNN	0.8658	0.02641	0.7982	0.45256
PNN∗	0.8672	0.02636	0.7987	0.45214
LR & DNN	0.8673	0.02634	0.7981	0.46772
FM & DNN	0.8661	0.02640	0.7850	0.45382
DeepFM	**0.8715**	**0.02618**	**0.8007**	**0.45083**

Hyper-parameter study

- **Activation function**
  - ReLU & tanh > sigmoid
- **Dropout**
  - 0.6 ~ 0.9
  - Adding randomness to model → strengthen model's robustness
- **Number of neurons per layer**
  - Complexity
  - Increasing the number of neurons does not always bring benefit
- **Number of hidden layers**
  - Increasing number of hidden layers → degraded performance = Overfitting
- **Network shape**
  - Constant (200-200-200 neurons per layer) is the best

Conclusions

- **FM component + Deep component**

- **Performance improvement from …**
  - **Does not** need any pre-training
  - Learns **both high- and low-** order feature interactions
  - Introduces a sharing strategy of feature embedding to **avoid feature engineering**