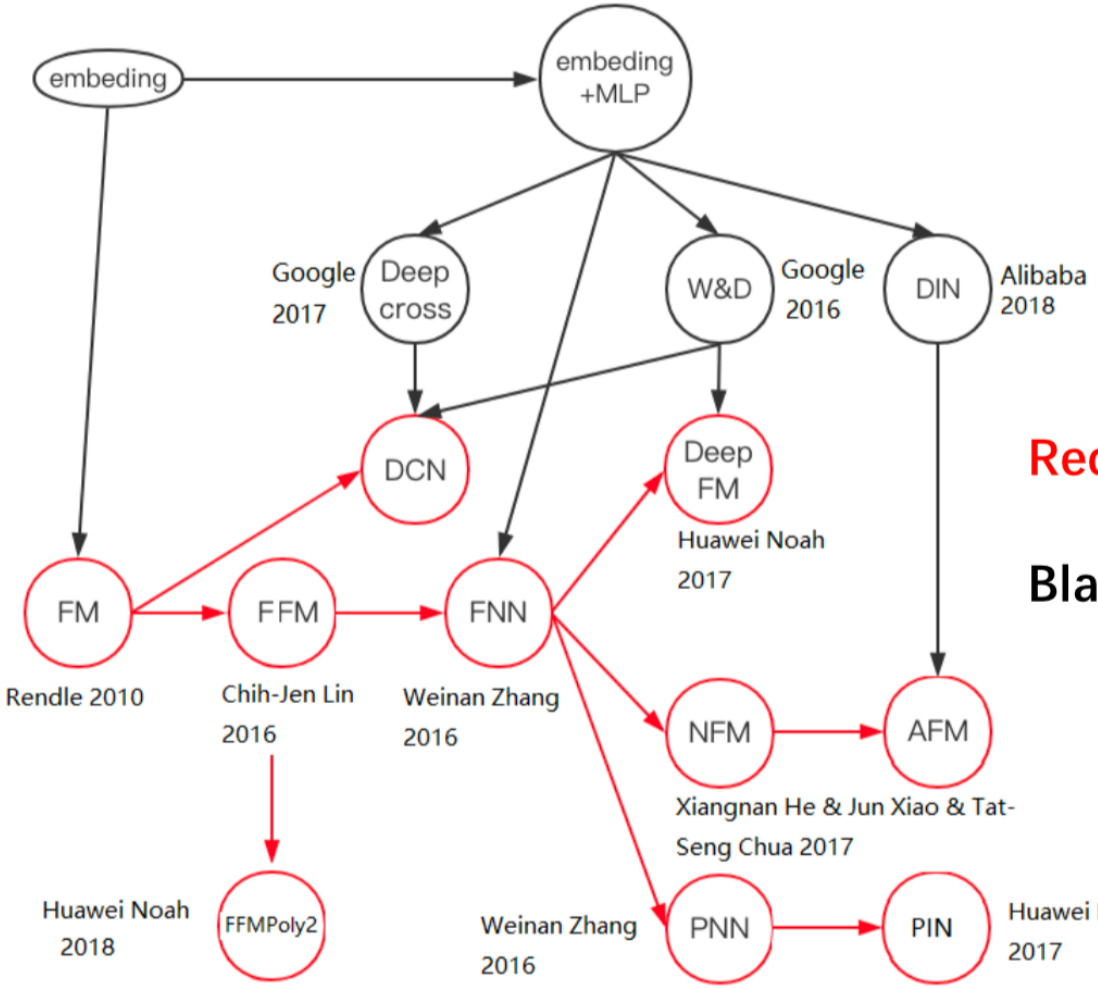
DeepFM

딥러닝을 활용한 추천시스템 발전 과정

Evolution of deep learning for recommender system



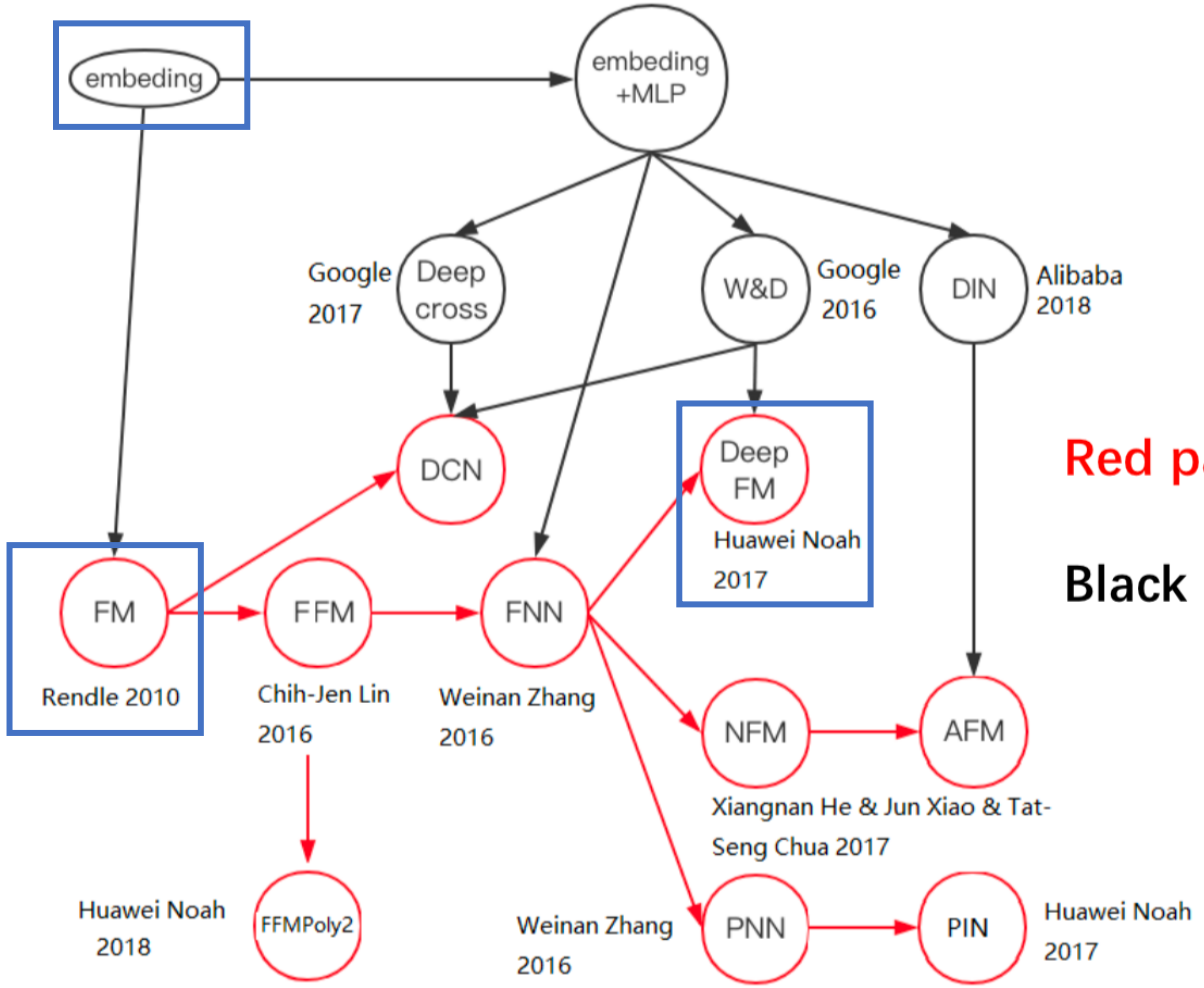
Red path : FM path

Black path : embedding + MLP path

PAI PUBLIC AI

딥러닝을 활용한 추천시스템 발전 과정

Evolution of deep learning for recommender system



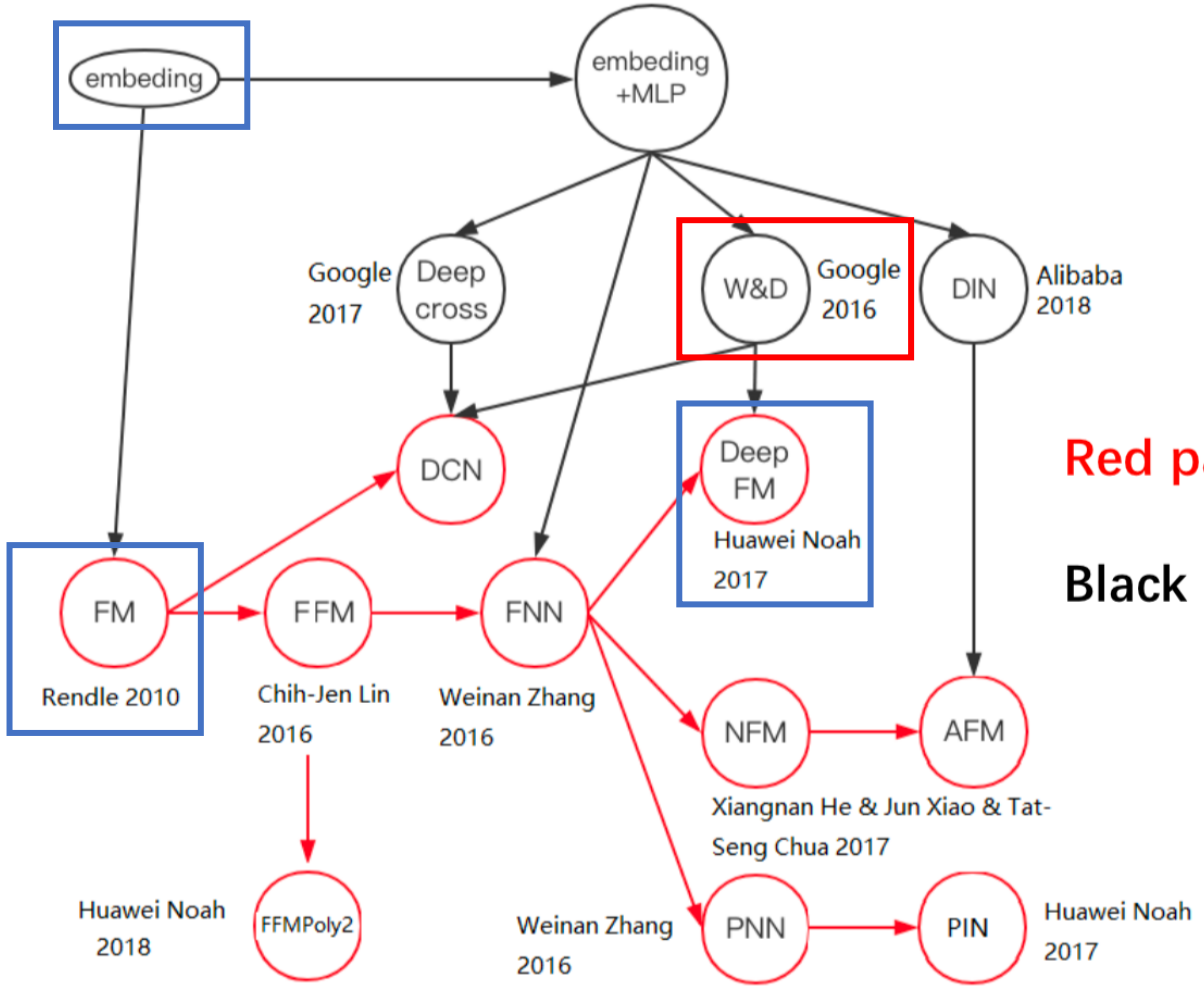
Red path : FM path

Black path : embedding + MLP path

PAI PUBLIC AI

딥러닝을 활용한 추천시스템 발전 과정

Evolution of deep learning for recommender system



DeepFM은 Wide & Deep Network을 계승한 모델

Red path : FM path

Black path : embedding + MLP path

PAI PUBLIC AI

Wide & Deep Learning

Google Play App 추천 시스템에 활용된 알고리즘

PAI
PUBLIC AI

Wide & Deep Learning가 다루는 문제

ABSTRACT

Generalized linear models with nonlinear feature transformations are widely used for large-scale regression and classification problems with sparse inputs. Memorization of feature interactions through a *wide* set of cross-product feature transformations are effective and interpretable, while generalization requires more feature engineering effort. With less feature engineering, *deep* neural networks can generalize better to unseen feature combinations through low-dimensional dense embeddings learned for the sparse features. However, deep neural networks with embeddings can over-generalize and recommend less relevant items when the user-item interactions are sparse and high-rank. In this paper, we present Wide & Deep learning—jointly trained wide linear models and deep neural networks—to combine the benefits of memorization and generalization for recommender systems. We productionized and evaluated the system on Google Play, a commercial mobile app store with over one billion active users and over one million apps. Online experiment results show that Wide & Deep significantly increased app acquisitions compared with wide-only and deep-only models. We have also open-sourced our implementation in TensorFlow.

Wide & Deep Learning가 다루는 문제

ABSTRACT

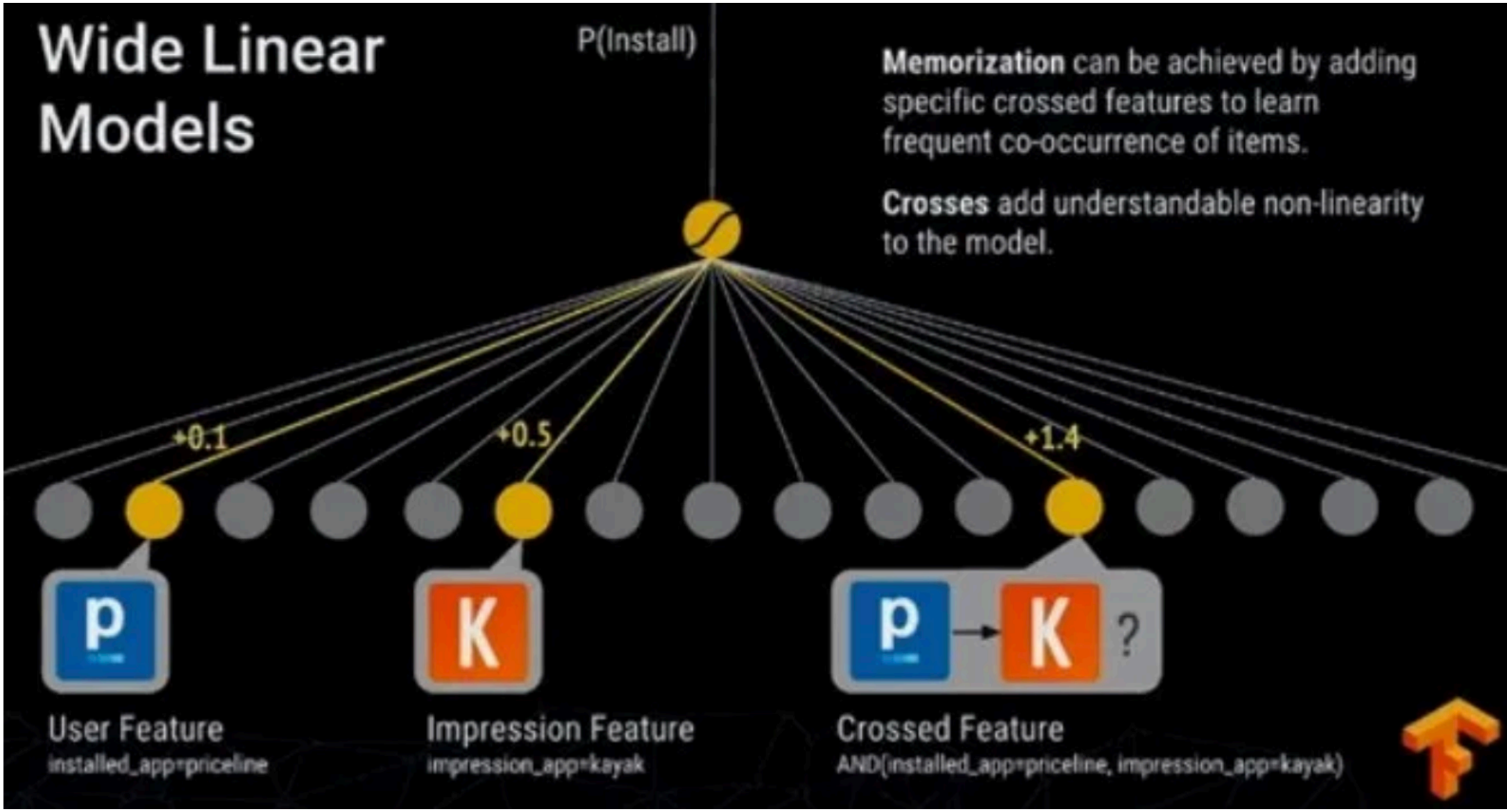
Generalized linear models with nonlinear feature transformations are widely used for large-scale regression and classification problems with sparse inputs. Memorization of feature interactions through a *wide* set of cross-product feature transformations are effective and interpretable, while generalization requires more feature engineering effort. With less feature engineering, *deep* neural networks can generalize better to unseen feature combinations through low-dimensional dense embeddings learned for the sparse features. However, deep neural networks with embeddings can over-generalize and recommend less relevant items when the user-item interactions are sparse and high-rank. In this paper, we present Wide & Deep learning—jointly trained wide linear models and deep neural networks—to combine the benefits of memorization and generalization for recommender systems. We productionized and evaluated the system on Google Play, a commercial mobile app store with over one billion active users and over one million apps. Online experiment results show that Wide & Deep significantly increased app acquisitions compared with wide-only and deep-only models. We have also open-sourced our implementation in TensorFlow.

$$y = w_1x_1 + w_2x_2 + w_3x_3 + w_{1,2}x_1x_2 + w_{1,3}x_1x_3 + w_{2,3}x_2x_3$$

cross-product feature transformatio

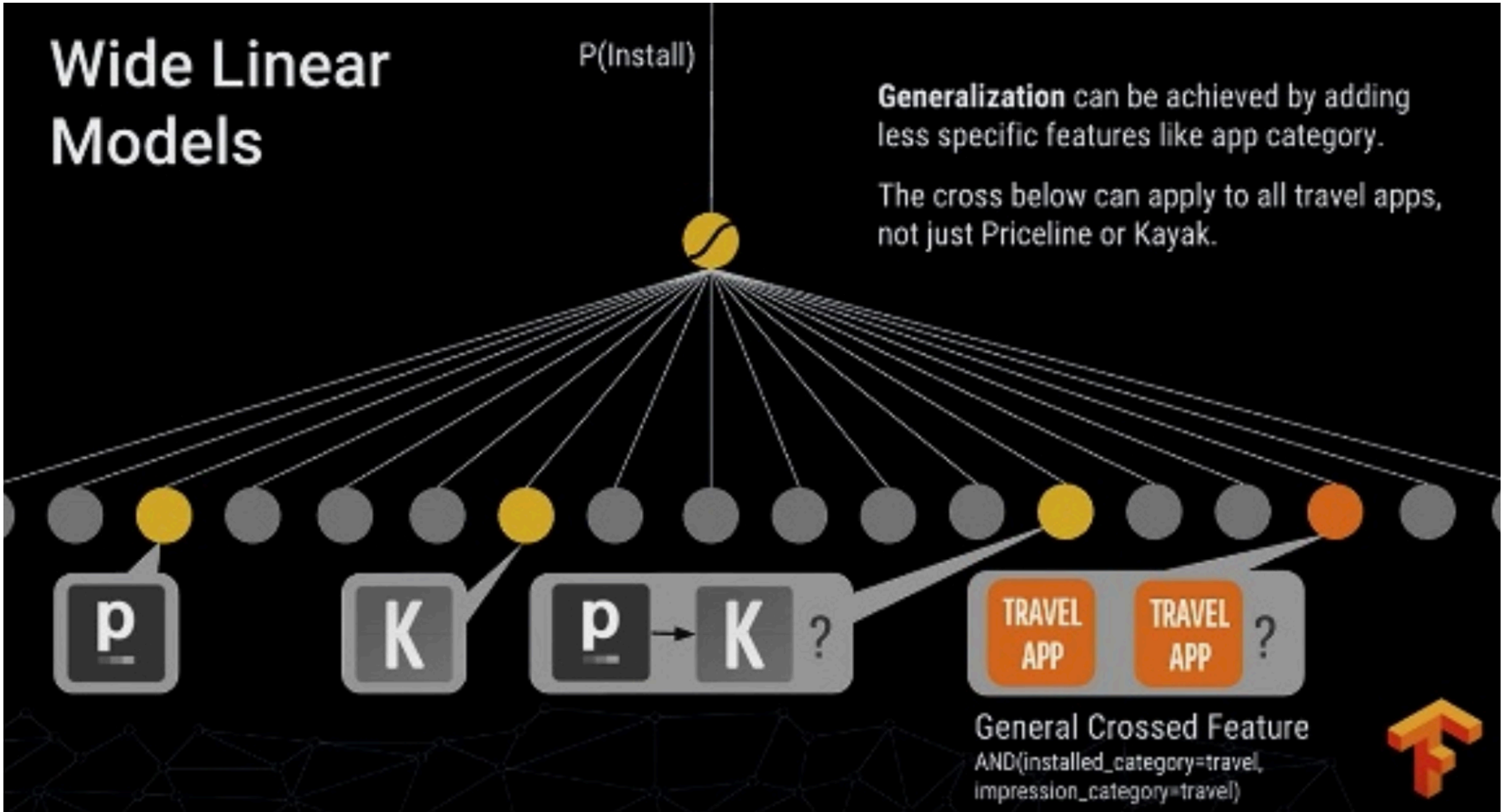
PAI
PUBLIC AI

Wide & Deep Learning



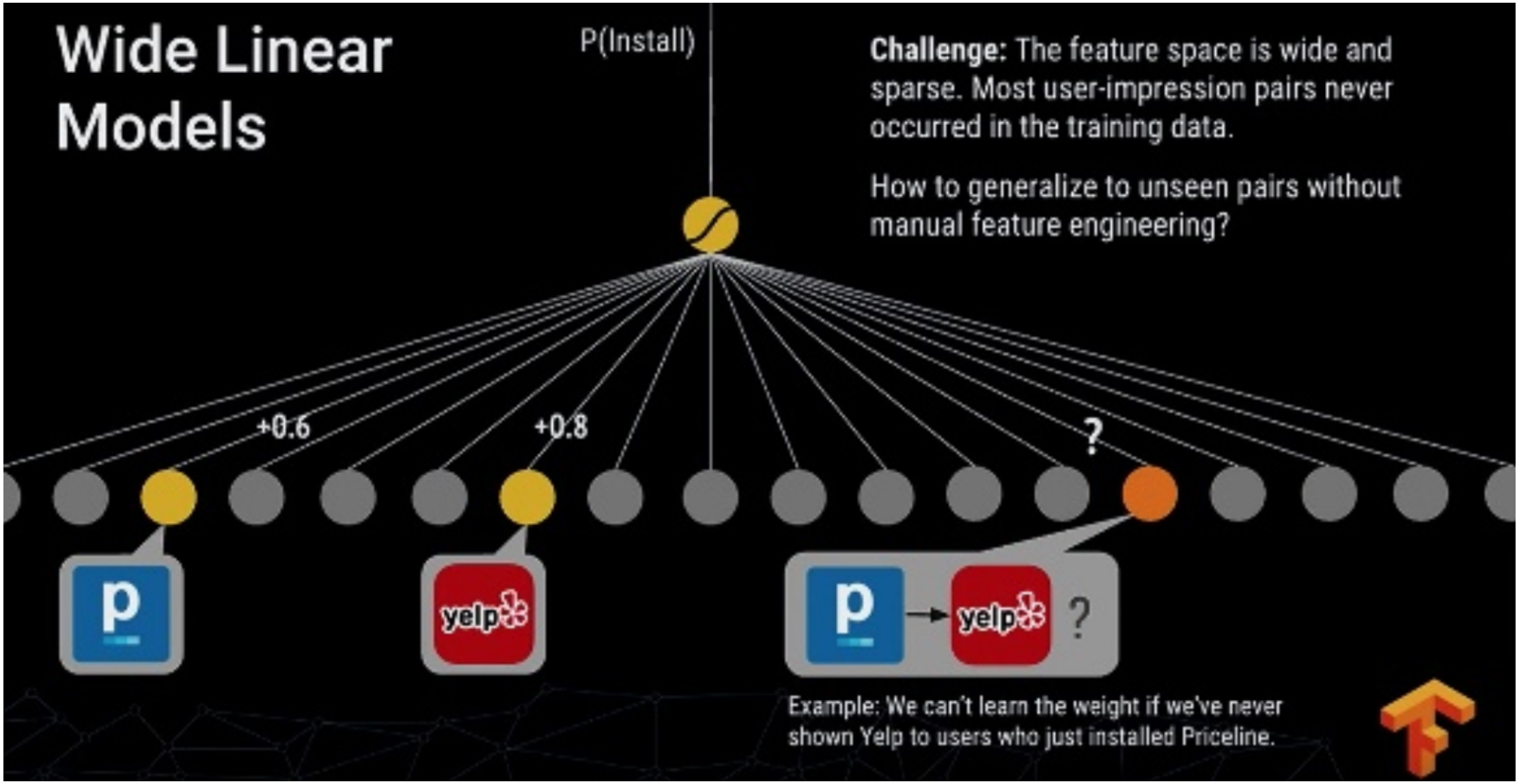
PAI
PUBLIC AI

Wide & Deep Learning



PAI
PUBLIC AI

Wide & Deep Learning



PAI
PUBLIC AI

[recap] 명목형 변수를 위한 선형회귀 with 상호작용

$$y = \sigma(W_{\text{고객번호}}X_{\text{고객번호}} + W_{\text{나이}}X_{\text{나이}} + W_{\text{성별}}X_{\text{성별}} + W_{\text{영화번호}}X_{\text{영화번호}} + W_{\text{연도}}X_{\text{연도}} + W_{\text{장르}}X_{\text{장르}} + W_0)$$

상호작용도 학습시키고 싶다! => 상호작용을 학습하기 위한 가중치도 필요

상호작용 가짓 수

(고객, 나이) = 943 * 12
(고객, 성별) = 943 * 2
(고객, 영화) = 943 * 1682
(고객, 연도) = 943 * 77
(고객, 장르) = 943 * 20

(나이, 성별) = 12 * 2
(나이, 영화) = 12 * 1682
(나이, 연도) = 12 * 77
(나이, 장르) = 12 * 20
(성별, 영화) = 2 * 1682

(성별, 연도) = 2 * 77
(성별, 장르) = 2 * 20
(영화, 연도) = 1682 * 77
(영화, 장르) = 1682 * 20
(연도, 장르) = 77 * 20

총 상호작용 가짓수 = 1,880,423 개 → 가중치가 지나치게 많다!

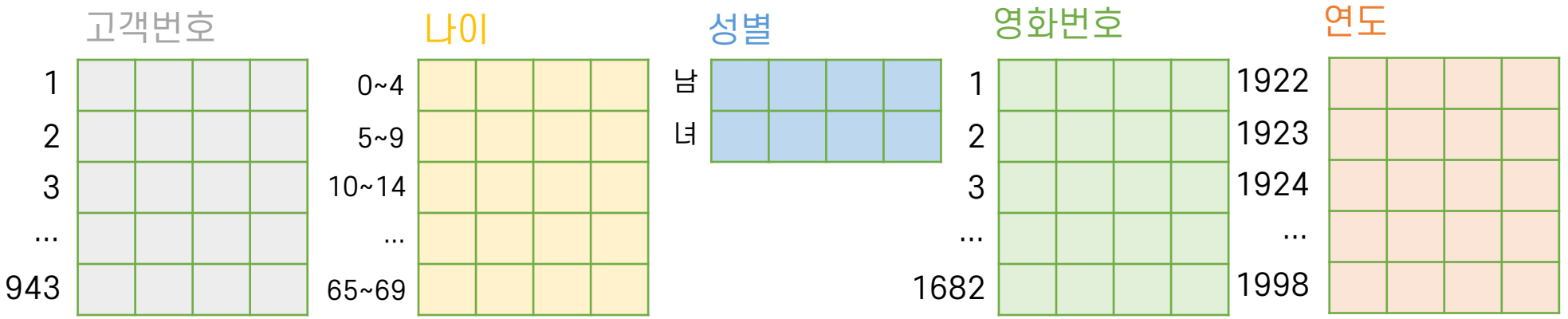
PAI PUBLIC AI

[recap] Factorization Machine의 방법

$$y = \sigma(W_{\text{고객번호}}X_{\text{고객번호}} + W_{\text{나이}}X_{\text{나이}} + W_{\text{성별}}X_{\text{성별}} + W_{\text{영화번호}}X_{\text{영화번호}} + W_{\text{연도}}X_{\text{연도}} + W_0)$$

상호작용도 학습시키고 싶다! => 상호작용을 계산하기 위한 임베딩 행렬(Factor)을 각각 구성하자

상호작용 임베딩 행렬



Wide & Deep Learning의 방법 (1) wide part

$$y = \sigma(W_{\text{고객번호}}X_{\text{고객번호}} + W_{\text{나이}}X_{\text{나이}} + W_{\text{성별}}X_{\text{성별}} + W_{\text{영화번호}}X_{\text{영화번호}} + W_{\text{연도}}X_{\text{연도}} + W_0)$$

데이터 분석을 통해, 강한 상호작용을 나타내는 Feature을 우선 도출하자 -> (1) Wide Part

상호작용 가짓 수

(고객, 나이) = 943 * 12
(고객, 성별) = 943 * 2
(고객, 영화) = 943 * 1682
(고객, 연도) = 943 * 77
(고객, 장르) = 943 * 20

(나이, 성별) = 12 * 2
(나이, 영화) = 12 * 1682
(나이, 연도) = 12 * 77
(나이, 장르) = 12 * 20
(성별, 영화) = 2 * 1682

(성별, 연도) = 2 * 77
(성별, 장르) = 2 * 20
(영화, 연도) = 1682 * 77
(영화, 장르) = 1682 * 20
(연도, 장르) = 77 * 20

PAI
PUBLIC AI

Wide & Deep Learning의 방법 (1) wide part

$$y = \sigma(W_{\text{고객번호}} X_{\text{고객번호}} + W_{\text{나이}} X_{\text{나이}} + W_{\text{성별}} X_{\text{성별}} + W_{\text{영화번호}} X_{\text{영화번호}} + W_{\text{연도}} X_{\text{연도}} + W_0)$$

데이터 분석을 통해, 강한 상호작용을 나타내는 Feature을 우선 도출하자 -> (1) Wide Part

상호작용 가짓 수

~~(고객, 나이) = 943 * 12~~
~~(고객, 성별) = 943 * 2~~
~~(고객, 영화) = 943 * 1682~~
~~(고객, 연도) = 943 * 77~~
(고객, 장르) = 943 * 20

~~(나이, 성별) = 12 * 2~~
~~(나이, 영화) = 12 * 1682~~
(나이, 연도) = 12 * 77
~~(나이, 장르) = 12 * 20~~
(성별, 영화) = 2 * 1682

~~(성별, 연도) = 2 * 77~~
~~(성별, 장르) = 2 * 20~~
~~(영화, 연도) = 1682 * 77~~
~~(영화, 장르) = 1682 * 20~~
~~(연도, 장르) = 77 * 20~~

PAI
PUBLIC AI

Wide & Deep Learning의 방법 (1) wide part

$$y = \sigma(W_{\text{고객번호}}X_{\text{고객번호}} + W_{\text{나이}}X_{\text{나이}} + W_{\text{성별}}X_{\text{성별}} + W_{\text{영화번호}}X_{\text{영화번호}} + W_{\text{연도}}X_{\text{연도}} + W_0)$$

데이터 분석을 통해, 강한 상호작용을 나타내는 Feature을 우선 도출하자 -> (1) Wide Part

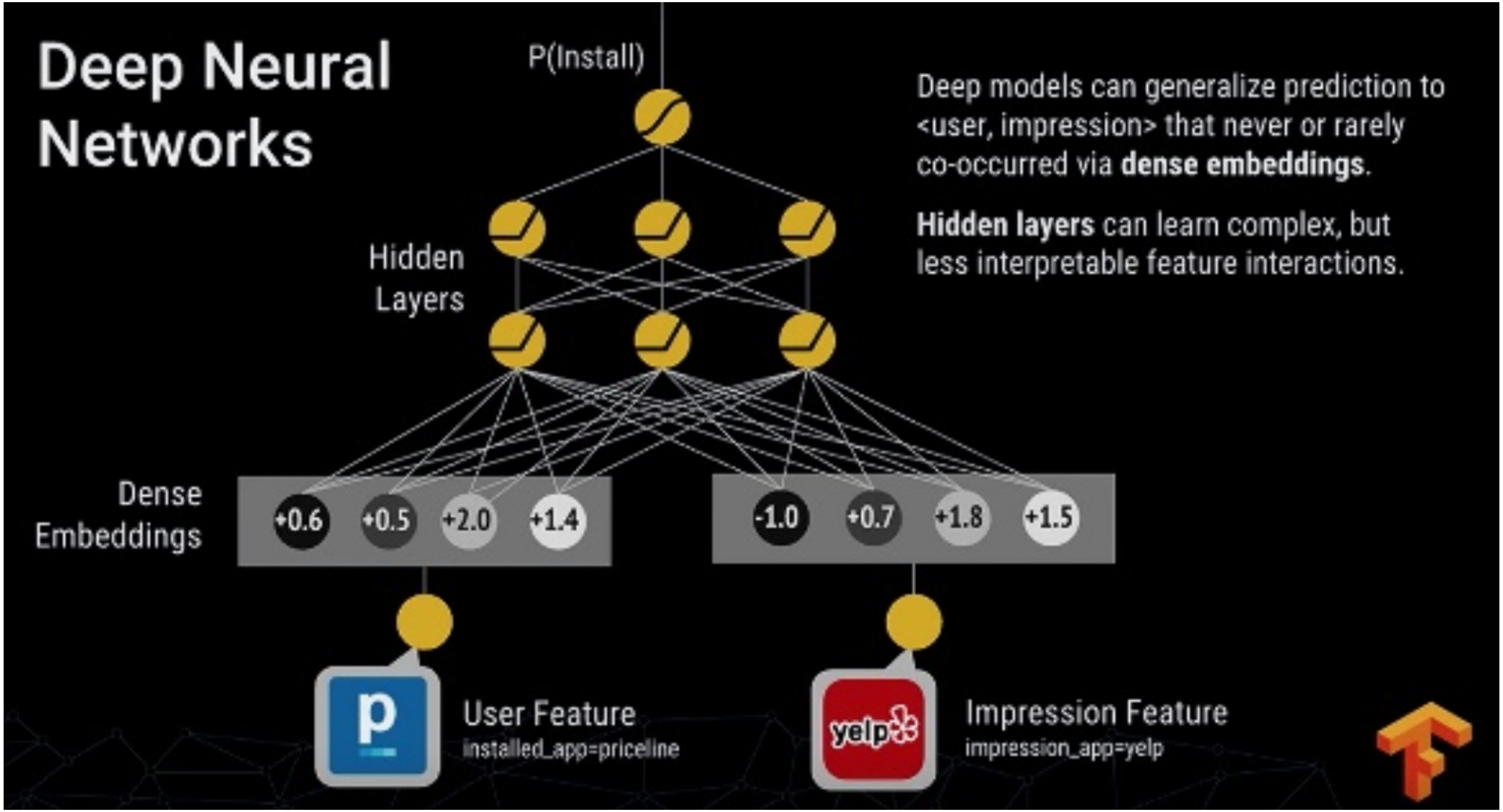
상호작용 가짓 수

(고객, 나이) = 943 * 12	(나이, 성별) = 12 * 2	(성별, 연도) = 2 * 77
(고객, 성별) = 943 * 2	(나이, 영화) = 12 * 1682	(성별, 장르) = 2 * 20
(고객, 영화) = 943 * 1682	(나이, 연도) = 12 * 77	(영화, 연도) = 1682 * 77
(고객, 연도) = 943 * 77	(나이, 장르) = 12 * 20	(영화, 장르) = 1682 * 20
(고객, 장르) = 943 * 20	(성별, 영화) = 2 * 1682	(연도, 장르) = 77 * 20

943*20 + 12*77 + 2*1682 = 23,148개

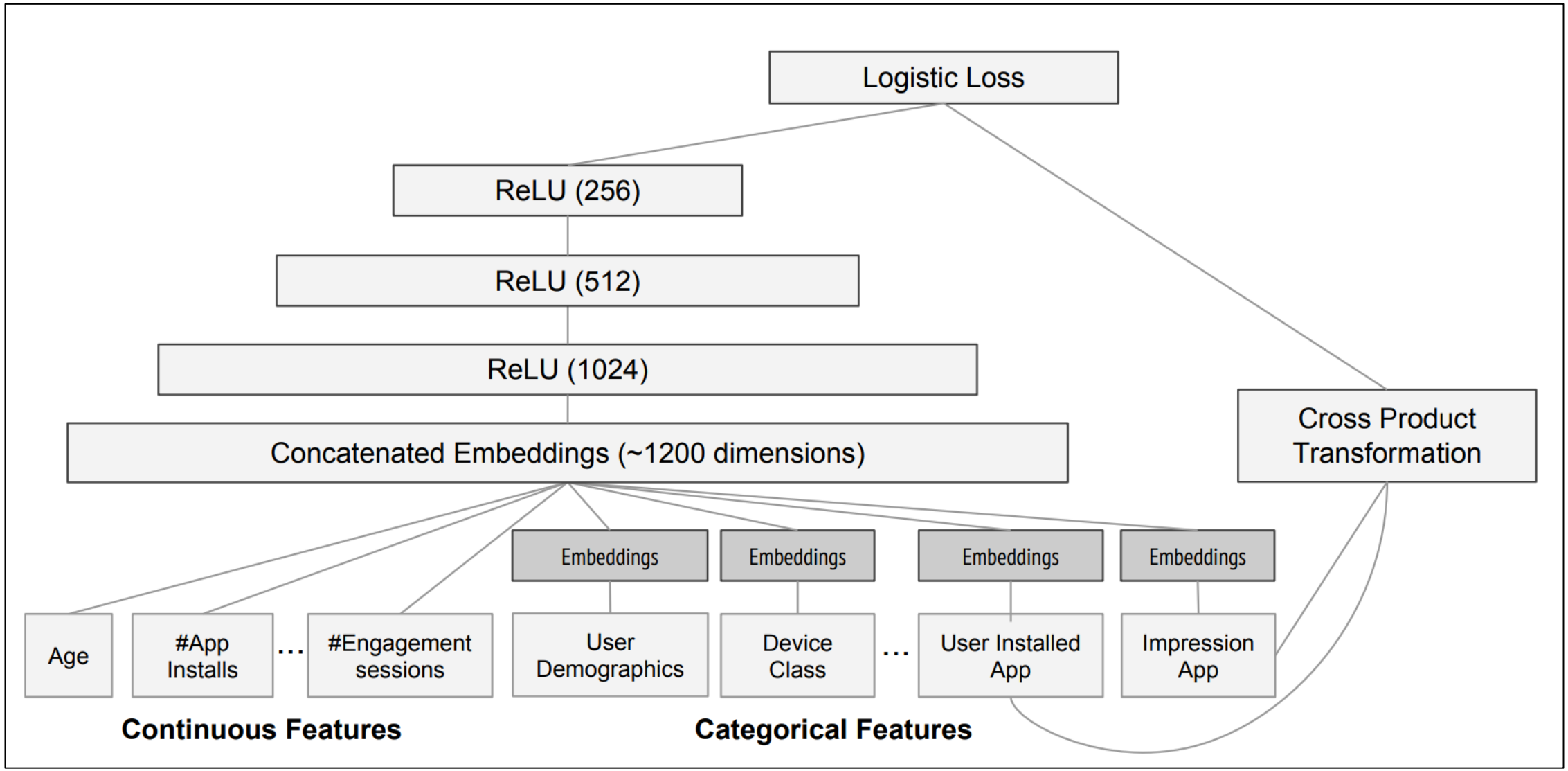
PAI PUBLIC AI

Wide & Deep Learning의 방법 (2) deep part



PAI
PUBLIC AI

Wide & Deep Learning의 구조



PAI
PUBLIC AI

Wide & Deep Learning의 의미

Memorization & Generalization

	특징	장점	단점
Memorization (Wide)	• 히스토리 데이터 • 직접 연관 데이터	• 간단함 • 주제 식별	• 학습 데이터에 있는 것들만 학습 가능 • 뻔한 추천
Generalization (Deep)	• Dense vector embedding • 새로운 feature 조합 탐색	• 다양성 개선 • 추가 feature engineering이 필요 없음	• Sparse & High-rank의 경우 추천 성능이 떨어짐 • 관련 없는 추천

Wide & Deep Learning의 코드

https://github.com/tensorflow/models/tree/master/official/r1/wide_deep

```
# Wide columns and deep columns.
base_columns = [
    education, marital_status, relationship, workclass, occupation,
    age_buckets,
]

crossed_columns = [
    tf.feature_column.crossed_column(
        ['education', 'occupation'], hash_bucket_size=_HASH_BUCKET_SIZE),
    tf.feature_column.crossed_column(
        [age_buckets, 'education', 'occupation'],
        hash_bucket_size=_HASH_BUCKET_SIZE),
]

wide_columns = base_columns + crossed_columns

deep_columns = [
    age,
    education_num,
    capital_gain,
    capital_loss,
    hours_per_week,
    tf.feature_column.indicator_column(workclass),
    tf.feature_column.indicator_column(education),
    tf.feature_column.indicator_column(marital_status),
    tf.feature_column.indicator_column(relationship),
    # To show an example of embedding
    tf.feature_column.embedding_column(occupation, dimension=8),
]

return wide_columns, deep_columns
```

PAI
PUBLIC AI

Wide & Deep Learning의 코드

https://github.com/tensorflow/models/tree/master/official/r1/wide_deep

```
# Wide columns and deep columns.
base_columns = [
    education, marital_status, relationship, workclass, occupation,
    age_buckets,
]

crossed_columns = [
    tf.feature_column.crossed_column(
        ['education', 'occupation'], hash_bucket_size=_HASH_BUCKET_SIZE),
    tf.feature_column.crossed_column(
        [age_buckets, 'education', 'occupation'],
        hash_bucket_size=_HASH_BUCKET_SIZE),
]

wide_columns = base_columns + crossed_columns

deep_columns = [
    age,
    education_num,
    capital_gain,
    capital_loss,
    hours_per_week,
    tf.feature_column.indicator_column(workclass),
    tf.feature_column.indicator_column(education),
    tf.feature_column.indicator_column(marital_status),
    tf.feature_column.indicator_column(relationship),
    # To show an example of embedding
    tf.feature_column.embedding_column(occupation, dimension=8),
]

return wide_columns, deep_columns
```

각 Column마다 wide part인지 deep part인지를 지정해주어야 함

- Hand-Craft Data Engineering이 많이 듦.
- 논문에서는 각 피쳐 별로 어떻게 wide와 deep으로 나누는지 나와있지 않음

PAI
PUBLIC AI

Wide & Deep Learning의 코드

https://chromium.googlesource.com/external/github.com/tensorflow/tensorflow/+/r0.10/tensorflow/g3doc/tutorials/wide_and_deep/index.md

The Wide Model: Linear Model with Crossed Feature Columns

The wide model is a linear model with a wide set of sparse and crossed feature columns:

```
wide_columns = [
  gender, native_country, education, occupation, workclass, marital_status, relationship, age_buckets,
  tf.contrib.layers.crossed_column([education, occupation], hash_bucket_size=int(1e4)),
  tf.contrib.layers.crossed_column([native_country, occupation], hash_bucket_size=int(1e4)),
  tf.contrib.layers.crossed_column([age_buckets, race, occupation], hash_bucket_size=int(1e6))]
```

Wide models with crossed feature columns can memorize sparse interactions between features effectively. That being said, one limitation of crossed feature columns is that they do not generalize to feature combinations that have not appeared in the training data. Let's add a deep model with embeddings to fix that.

DeepFM : Wide & Deep Learning을 개선

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

Abstract

Learning sophisticated feature interactions behind user behaviors is critical in maximizing CTR for recommender systems. Despite great progress, existing methods seem to have a strong bias towards low- or high-order interactions, or require expertise feature engineering. In this paper, we show that it is possible to derive an end-to-end learning model that emphasizes both low- and high-order feature interactions. The proposed model, DeepFM, combines the power of factorization machines for recommendation and deep learning for feature learning in a new neural network architecture. Compared to the latest Wide & Deep model from Google, DeepFM has a shared input to its "wide" and "deep" parts, with no need of feature engineering besides raw features. Comprehensive experiments are conducted to demonstrate the effectiveness and efficiency of DeepFM over the existing models for CTR prediction, on both benchmark data and commercial data.

PAI
PUBLIC AI

DeepFM : Wide & Deep Learning을 개선

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

Abstract

Learning sophisticated feature interactions behind user behaviors is critical in maximizing CTR for recommender systems. Despite great progress, existing methods seem to have a strong bias towards low- or high-order interactions, or require expertise feature engineering. In this paper, we show that it is possible to derive an end-to-end learning model that emphasizes both low- and high-order feature interactions. The proposed model, DeepFM, combines the power of factorization machines for recommendation and deep learning for feature learning in a new neural network architecture. Compared to the latest Wide & Deep model from Google, DeepFM has a shared input to its "wide" and "deep" parts, with no need of feature engineering besides raw features. Comprehensive experiments are conducted to demonstrate the effectiveness and efficiency of DeepFM over the existing models for CTR prediction, on both benchmark data and commercial data.

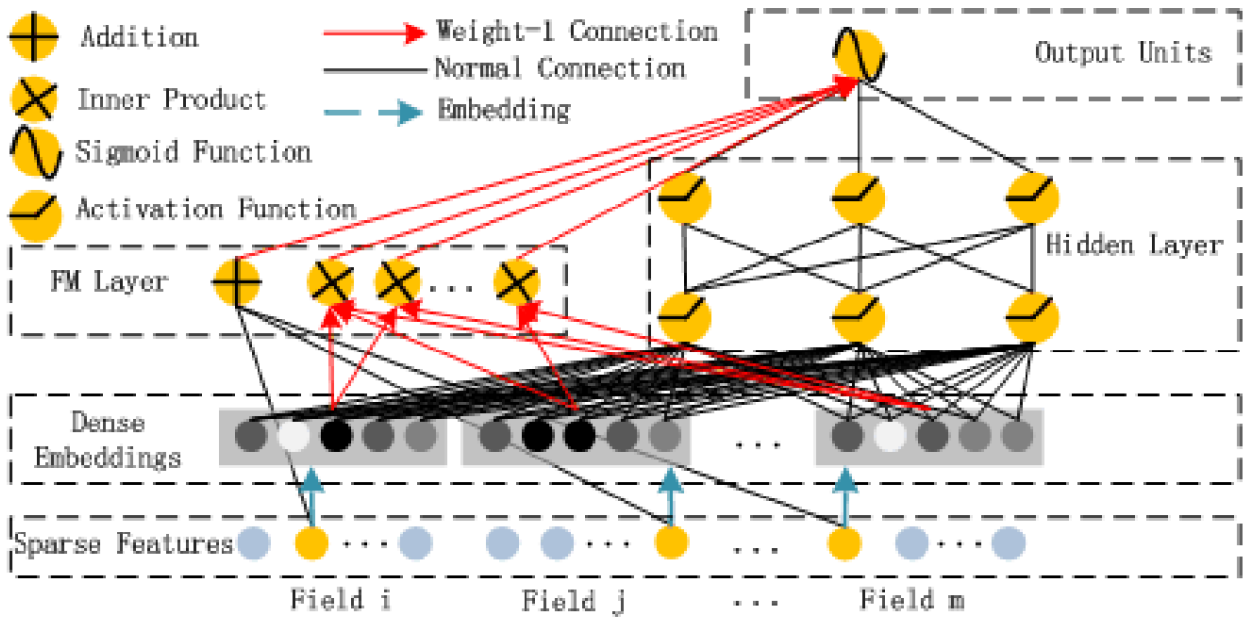


Figure 1: Wide & deep architecture of DeepFM. The wide and deep component share the same input raw feature vector, which enables DeepFM to learn low- and high-order feature interactions simultaneously from the input raw features.

PAI PUBLIC AI

DeepFM의 특징 (1) : FM와 Deep의 분할

FM Component

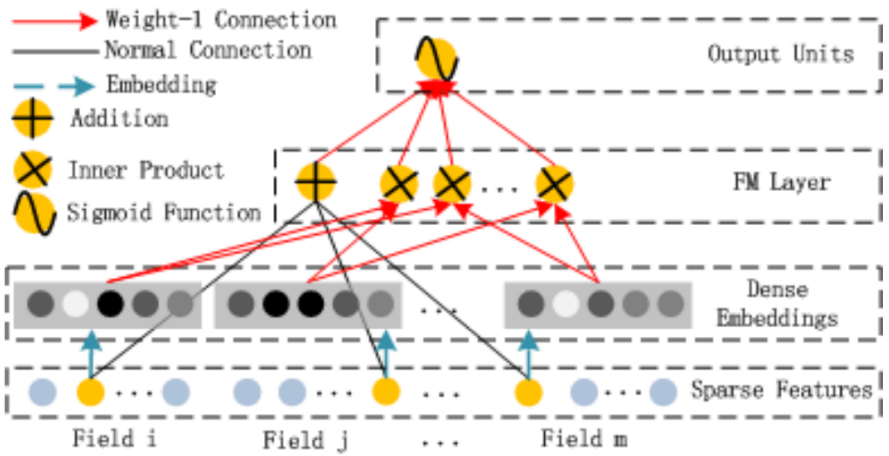


Figure 2: The architecture of FM.

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

Deep Component

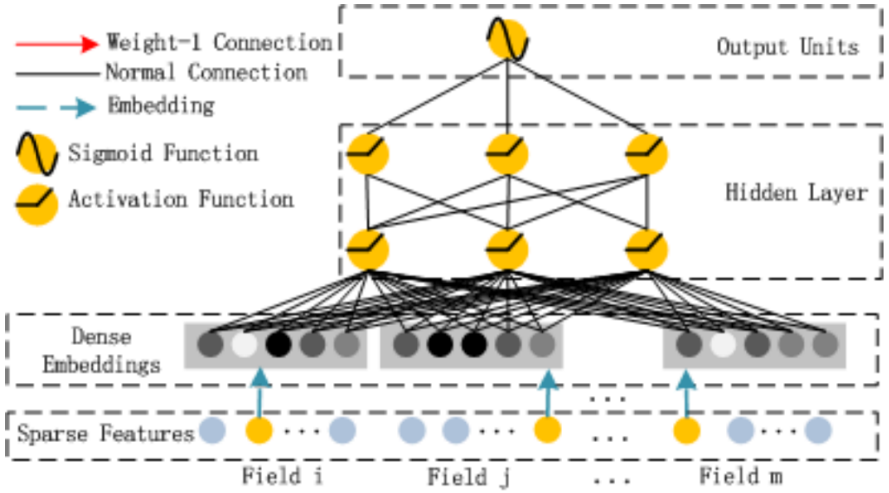


Figure 3: The architecture of DNN.

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

PAI PUBLIC AI

DeepFM의 특징 (2) : 임베딩 행렬의 공유

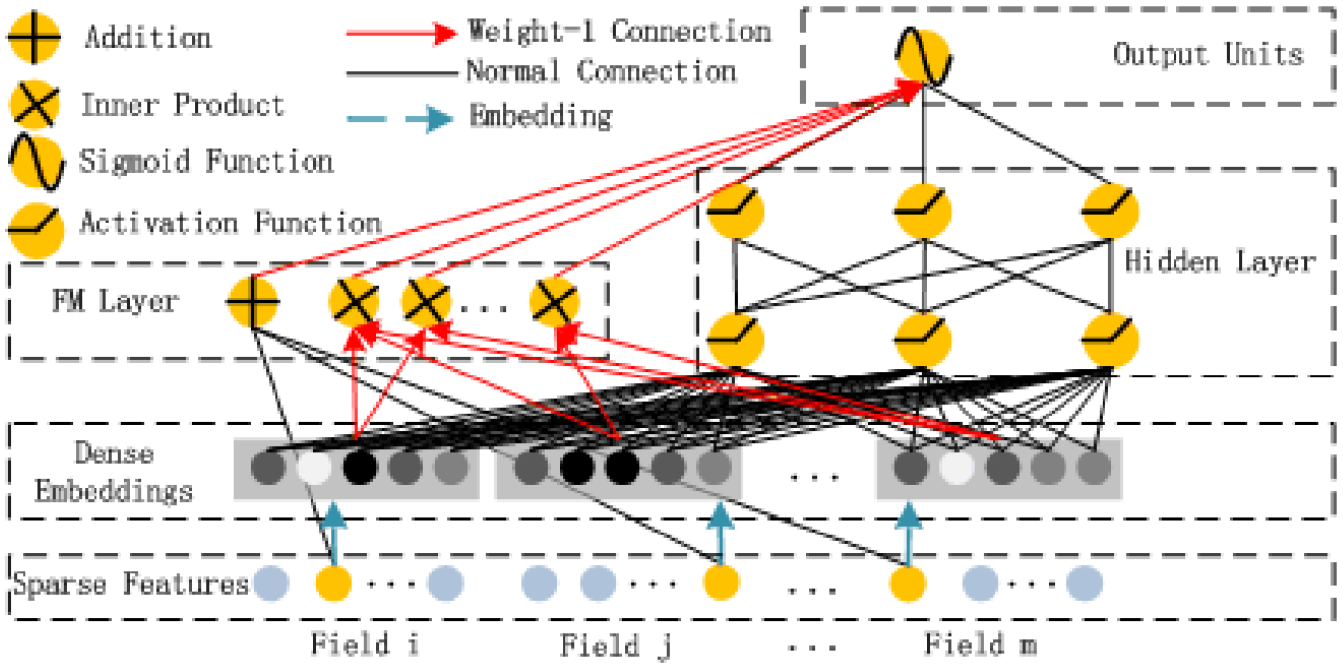


Figure 1: Wide & deep architecture of DeepFM. The wide and deep component share the same input raw feature vector, which enables DeepFM to learn low- and high-order feature interactions simultaneously from the input raw features.

DeepFM과 다른 네트워크의 비교

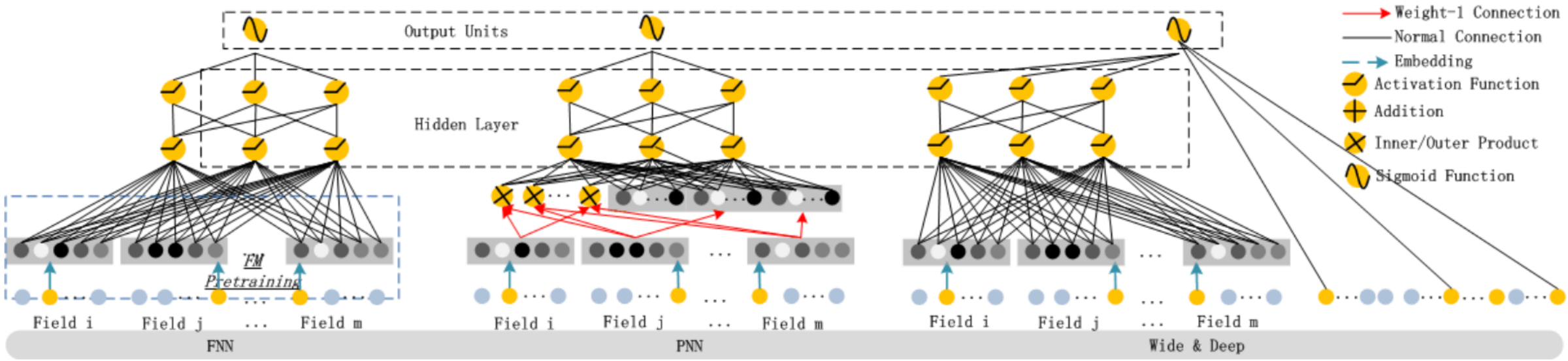


Figure 5: The architectures of existing deep models for CTR prediction: FNN, PNN, Wide & Deep Model

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	√	√	√	√