Wide & Deep Learning for Recommender Systems

By Heng-Tze Cheng et al in Google Inc

Lab Intern 정재원

Contents

Abstract

What is wide & deep

- Generalized linear models with nonlinear feature transformations

-> large-scale regression and classification problems with sparse inputs.

    - Simple & scalability

 # Wide set of cross-product feature transformation

- Strength: <u>**memorization**</u> of feature interaction
- Weak: need more feature engineering

 # DNN (Deep Neural Network)

- Strength: <u>**generalizing**</u> by low-dimensional dense embedding learned for the sparse features (less feature engineering)
- Weak: over generalize -> recommend less relevant items (interaction sparse & high-rank)

Wide & Deep learning: jointly train wide linear & deep neural networks for recommender system

<u>**(benefit of Memorization & Generalization)**</u>

Introduction

Intro

\# Recommender system = search ranking system

- main task: find relevant items & rank based on action (click, purchase etc)
- input query: set of user and contextual information
- output: ranked list of items

\# Use generalized linear model (logistic regression) for massive-scale online recommendation and ranking system

why? simple, scalable, interpretable

\# Goal: Achieve both **Memorization & Generalization** (similar to general search ranking system)

Memorization & **Generalization**

Introduction

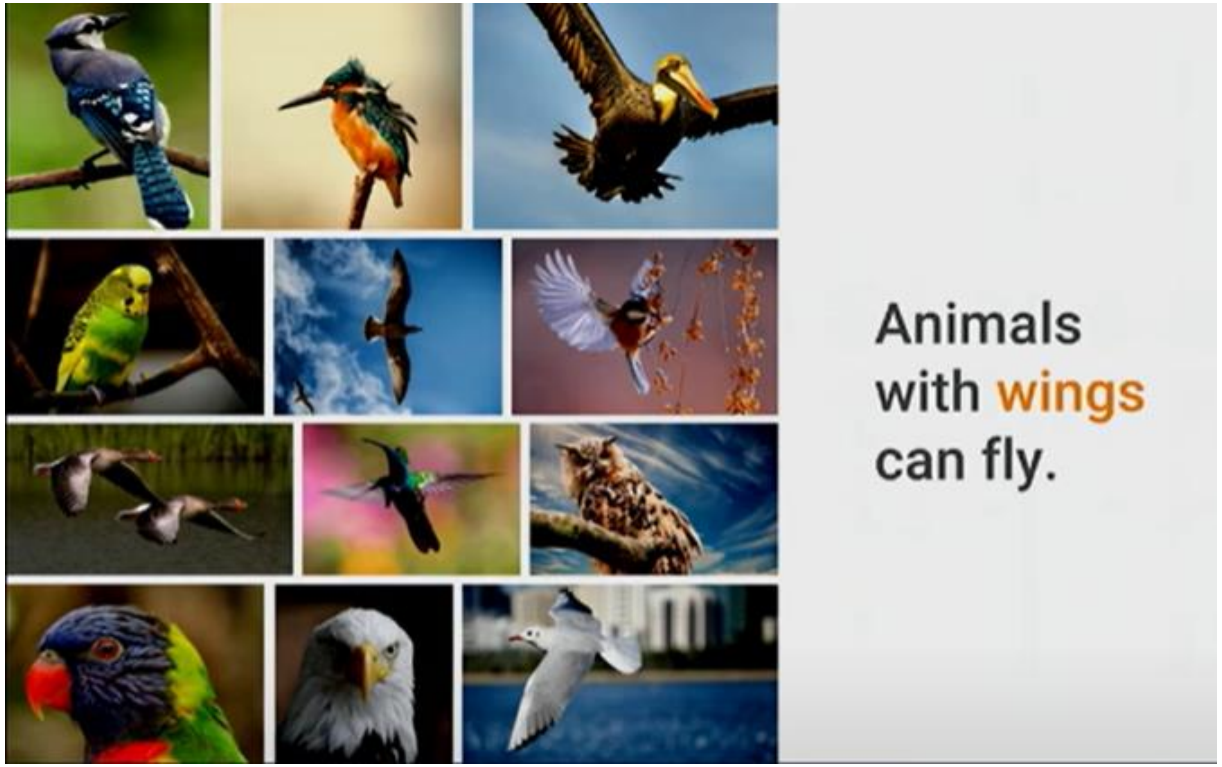
Intro

Memorization

Introduction

Intro

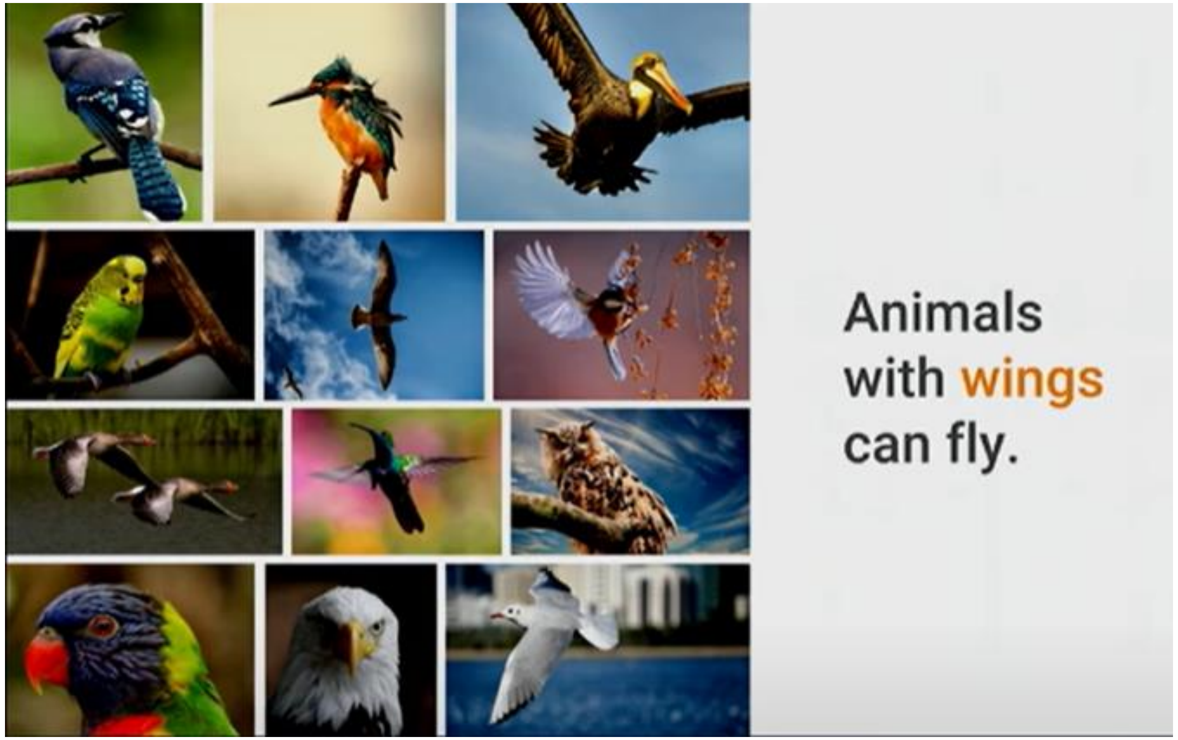
Generalization



Introduction

Intro

Memorization & Generalization



Introduction

Memorization

1. Memorization: learning frequency of co-occurrence items (features) & exploiting correlation in historical data

-> more topical & **directly relevant items (already performed)**

- Use cross-product transformation over sparse features
- Explain how co-occurrence of feature pair correlates with the target label

ex) AND(user_installed_app="Netflix", impression_app="pandora")

-> 1 if the user installed Netfix and then is later shown Pandora.

- Transformation can't generalize unseen pair (= learn only performed pair)

=> **Embedding based model can** ex) FM (Factorization Machine), DNN (Deep Neural Network)

Introduction

Generalization

2. Generalization: explore new combination based on transitivity of correlation

-> **improve diversity** of recommended items

- Use less granular feature(= have less detail)
- Manual engineering needed ex) Netflix -> category: video

ex) AND(user_installed_category=video, impression_category=music)

- Query-item matrix sparse& high rank (= specific preference, niche items)

-> no interaction between pair but make embedding -> recommend not relevant items

⇒ **Linear model with cross-product transformation memorize these "exception rules" with fewer parameters**

Wide & Deep Recommendation system

Recommendation system overview

\# User action

- Visit app -> **generate query** (include user & context features)
- Action in app (click, purchase etc) -> system return **list of apps (= impressions)**

=> **These action recorded in logs (= training data)**

\# Recommendation system

1. retrieval (검색): return candidates items by ML & human-defined rule
2. ranking: ranks all item by their score(= P(y | **x**))
   - y = user action label
   - x = features
     - user features (country, language)
     - contextual features (device, hour of day)
     - impression features (app age, historical statistic of an app)

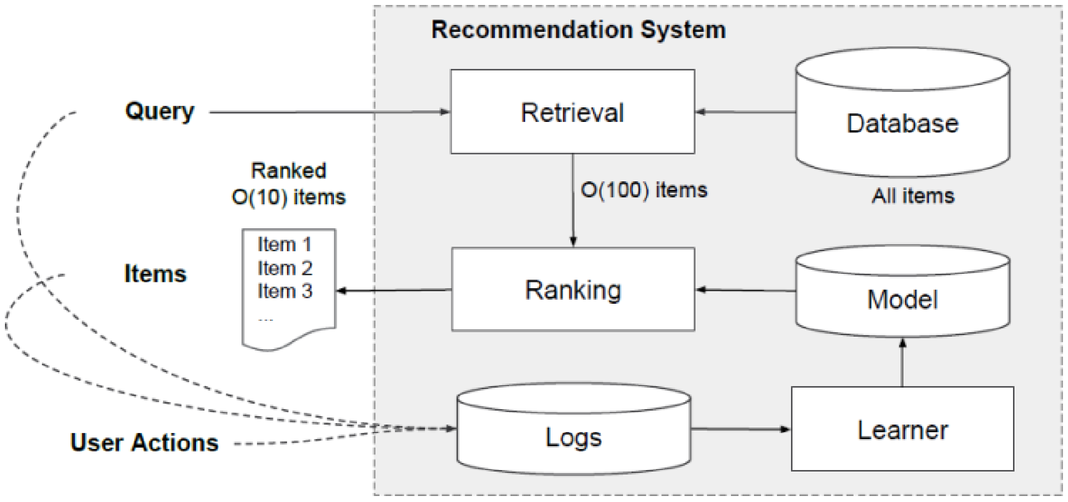


Figure 2: Overview of the recommender system.

Wide & Deep Recommendation system

Recommendation system overview

Wide & Deep Recommendation system's component

- Wide: Generalized linear model with cross-product transformation
- Deep: Neural Network model

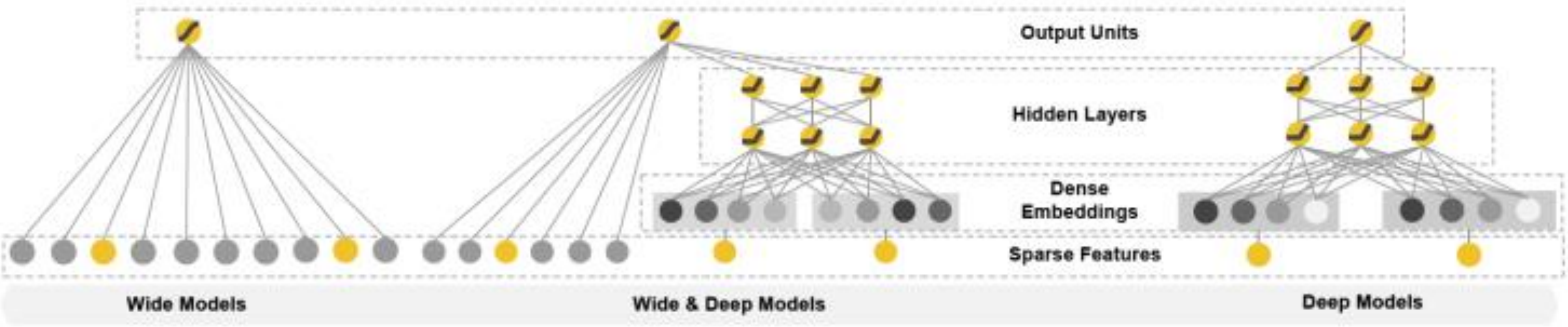


Figure 1: The spectrum of Wide & Deep models.

Wide & Deep Recommendation system

Learning – The Wide Component

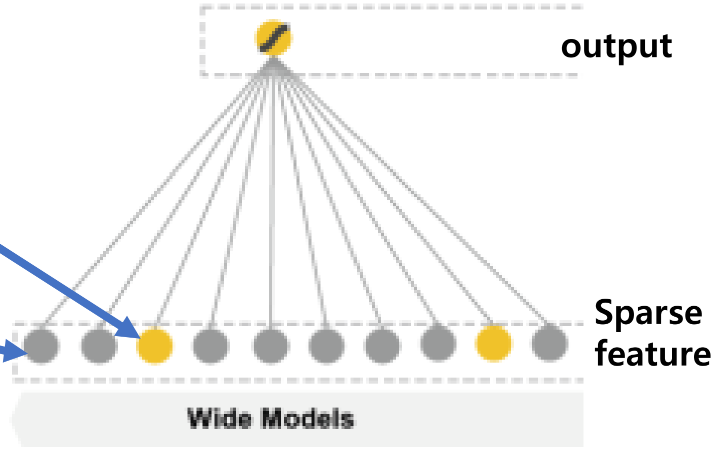
- Generalized linear model: $y = w^T x + b$
  - feature x = raw feature + cross-product-transformed feature
- **Cross-product transformation**
  - Equation: $\phi_k(\mathbf{x}) = \prod_{i=1}^{d} x_i^{c_{ki}} \quad c_{ki} \in \{0,1\}$, $c_{ki} = \begin{cases} 1, & i-th\ feature\ is\ part\ of\ k-th\ transformation \\ 0, & otherwise \end{cases}$
  - Example
    - Features: $Gender = [male, female] = [1,0]$, $age = [young\ , old] = [1,0]$, $Country = [\mathrm{kor}, usa] = [1,0]$
    - If 1st (k =1)transformation = "AND(gender=male, age=young)" (using gender & age features)
      - $\varphi_1 = x_1^{c_{11}}\ x_2^{c_{12}}\ x_3^{c_{13}} = x_1^1\ x_2^1\ x_3^0$ (c_{ki} = [1,1,0])
      - If user's feature vector x = [male, young, usa] = [1,1,0] => $\varphi_1 = 1^1 1^1\ 1^0 = 1$
  - Capture **binary features' interaction** & add non-linearity to generalized linear model

Wide & Deep Recommendation system

Learning – The Wide Component

- In Wide model, Cross-product transformation by using installation & impression features
  - Let) user_installed_app = [A,B] & user_impressed_app = [A,C]

Install	Impression	(install, impression)	Target
A	A	(1,1)	1
A	B	(1,0)	0
…	…	…	…
C	B	(0,0)	0
C	C	(0,1)	0

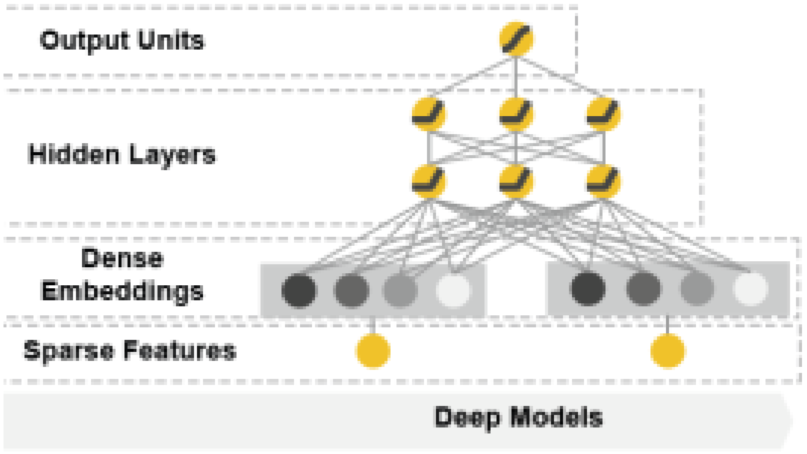


  - Model train all data which target value is 1
    - Good **memorization** & can train **niche combination and user's specific preference**
    - Can't train all data which target value is 0 -> **weak generalization**

Wide & Deep Recommendation system

Learning – The Deep Component

- Feedforward Neural Network
- High dimension Categorical string features -> low dimension and dense real value embedding vector
  - Often embedding O(10) to O(100)
- Equation: $a^{(l+1)} = f(W^{(l)}a^{(l)} + b^{(l)})$
  - l = layer number, f = activation function (= ReLUs)
  - $a^{(l)}$, $b^{(l)}$, $W^{(l)}$ = activations, biases, weights at l-th layer
- Embedding all pair of data = (install, impression)
  - **Train unseen pair(=new combination) by embedding vector (generalization)**
  - **Too sparse & high rank feature -> make less relevant recommendation (over generalized)**



App	2-D Embedding
A	
B	
C	

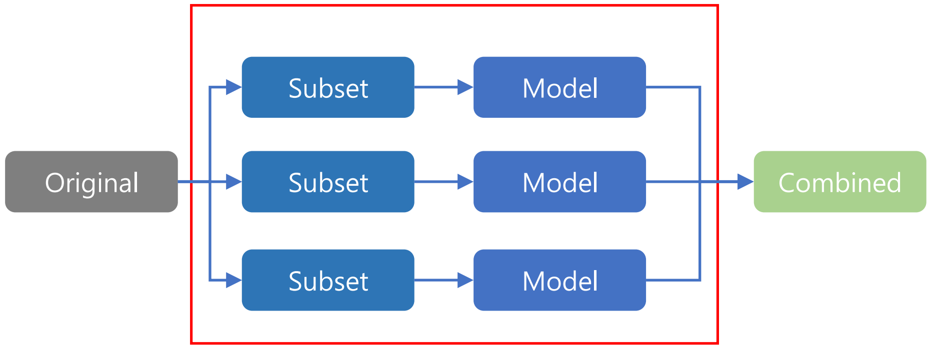
Wide & Deep Recommendation system

Joint Training of Wide & Deep Model

Ensemble

- Train separately without knowing each other
- Only combine at inference time (= not training time)
- Disjoint train -> size larger (more feature and more transformation)

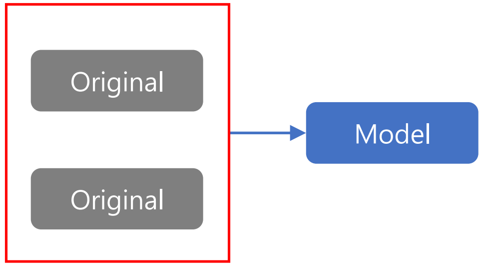
Ensemble training



Joint training

- **Optimize all parameter simultaneously (both wide and deep) at training time**
- Deep part with small number of Wide part's cross-product transformation **(size smaller)**
  - **Wide part complement weak point of deep part (= over generalization)**

Joint training



Wide & Deep Recommendation system

Joint Training of Wide & Deep Model

 # Joint training of Wide & Deep model

- **Back propagating gradient simultaneously both the Wide and Deep part by Mini Batch Gradient**
  - Wide part by **FTRL(Online Gradient Descent + Regularized Dual Average) algorithm** with L1 regularization
    - Online Gradient Descent: stochastic gradient descent but use **"most recent data"**
  - Deep part by **Ada-Grad algorithm** (= learning rate decay each element)

$$h \leftarrow h + \frac{\partial L}{\partial W} \odot \frac{\partial L}{\partial W}, \; \odot\text{: elementwise prouct}$$

$$W \leftarrow W - \eta \frac{1}{\sqrt{h}} \frac{\partial L}{\partial W}$$

[Ada-Grad Equation]

- **Combined weighted sum of output log odds (common logistic loss function)**
  - Equation: $P(Y=1|\mathbf{x}) = \sigma(\mathbf{w}_{wide}^T[\mathbf{x}, \phi(\mathbf{x})] + \mathbf{w}_{deep}^T a^{(l_f)} + b)$
    - Y = binary class label, $a^{(l_f)}$, b, $W^{(T)}$ = final activations, biases, weights
    - σ = sigmoid function, $\varphi(x)$ = cross-product transformation of original feature x

System Implementation

Apps Recommendation pipeline overview

- 3 stages exist
  1. Data Generation
  2. Model Training
  3. Model Serving

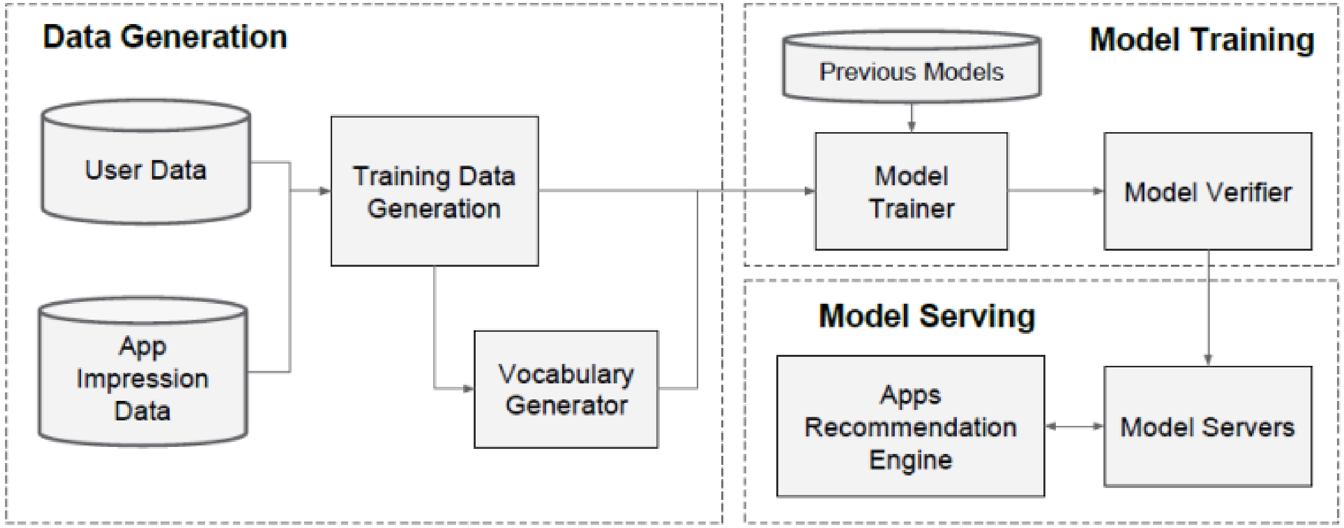


Figure 3: Apps recommendation pipeline overview.

System Implementation

1) Data Generation

- The label = app acquisition: 1 if the impressed app was installed

- Vocabularies: mapping categorical string feature -> integer ID, normalized [0, 1]
    - $normalized\ value\ in\ i - th\ quantiles = \frac{i-1}{n_q - 1}$, $\mathrm{n_q}$ quantiles

System Implementation

2) Model Training

1. Embedding
    - wide component -> cross product transformation
    - categorical string features -> integer ID = Vocabularies
    - continuous feature

2. Concatenate all embedding together

3. Concatenated vector into 3 ReLUs layers => logistic output unit

4. Joint training with common logistic loss function

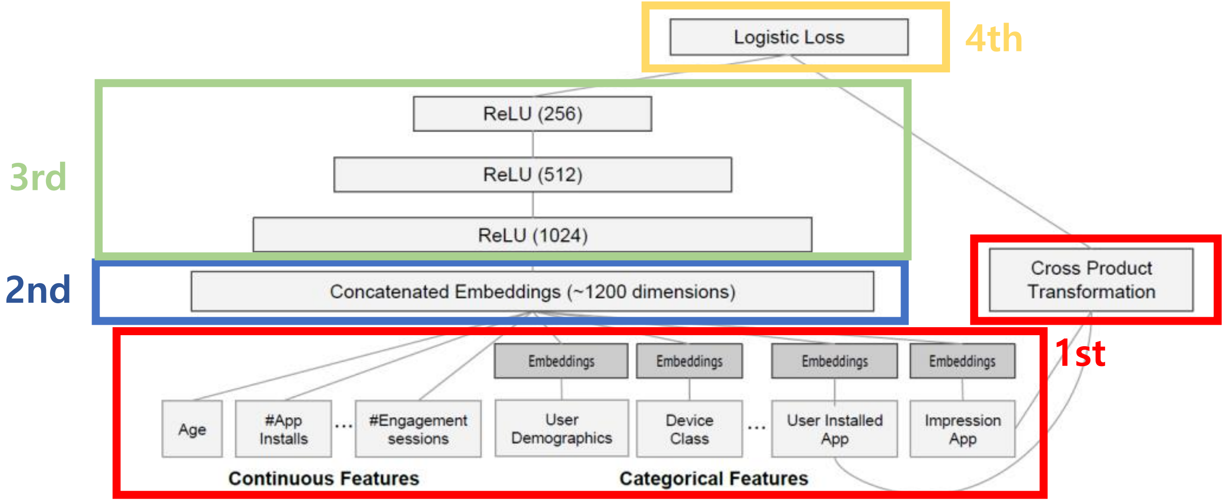


Figure 4: Wide & Deep model structure for apps recommendation.

- Retrain every time is expensive and delays (because every time new data arrival)

-> implement warm-starting system (= initialize embedding & weight from previous model)

System Implementation

3) Model Serving

- Receive set of app candidates from retrieval system

- Score calculated by Wide & Deep model

- To optimize performance time, multithreading parallelism by running smaller batches in parallel
  - multithread = multiple processing server
  - parallel processing with multiple server -> decrease performance time

Experiment Result

1) App Acquisitions

Compare Wide & deep with only wide, only deep

- Online: randomly select 1% users each group & implement A/B test
  - significant increase of gain with Wide & Deep

- Offline: not have impact than online but slightly increase than others
  - offline -> fixed label
  - online -> generate new exploratory recommendation, learn from new user

[A/B test]

Group	Control	experiment
Indicators	5,000	6,000(20%↑)

Table 1: Offline & online metrics of different models. Online Acquisition Gain is relative to the control.

Model	Offline AUC	Online Acquisition Gain
Wide (control)	0.726	0%
Deep	0.722	+2.9%
Wide & Deep	0.728	+3.9%

Experiment Result

2) Serving Performance

Compare Wide & deep along batch size & # of threads

- At same throughput, more threads and smaller batch size -> lower latency
  - Throughput = the amount processed by digital data transmission per unit hour (= Batch size x # of threads)
  - Latency = delay time

Table 2: Serving latency vs. batch size and threads.

Batch size	Number of Threads	Serving Latency (ms)
200	1	31
100	2	17
50	4	14

Related Work

- Wide & Deep's idea from FM(Factorization machines)
  - FM: add generalization to linear model by factorizing the interaction

- Joint training
  - In NLP reduce RNN's complexity by learning direct weight between input and output
  - It also apply to graphical models

- Previous recommender system used content information & CF(collaborative filtering) for rating matrix
  - Wide & Deep is different (use jointly training with user and impression data)

Conclusion

- **Keyword in recommendation system**
  - **Memorization:** learning co-occurrence and correlation **-> more topical & directly relevant**
  - **Generalization:** explore unseen combination **-> improve diversity**
- **Component of model**
  - **Wide linear model** -> **Memorize** sparse feature by using cross product transformation
  - **Deep Neural Network(DNN)** -> **Generalize** unseen feature interaction by low dimensional embedding

=> Wide & Deep model -> Combine these models' strengths

- Experiment at Google Play, (a massive-scale commercial app store)

=> Wide & Deep make significant result compared to only deep, only-wide model in Online experiment

Thank you

For listening