

Wide & Deep Learning for Recommender Systems

22.08.13 이지현

Contents

- Introduction
- Recommender system overview
- Wide & Deep learning
- System implementation
- Experiments & Results
- Conclusion

개요

- 2016년 구글이 발표한 추천랭킹 알고리즘
- 구글 플레이 앱 추천에 활용
 - 구체적으로, user 의 검색 query 를 바탕으로, candidate 앱을 정렬하는데 적용됨.
- 간단한 아이디어로 성능을 개선시킨 범용 추천엔진
 - Memorization & Generalization
 - Google play 추천 엔진 개선
 - TensorFlow Open Source

<https://arxiv.org> > cs ▾

[1606.07792] Wide & Deep Learning for Recommender Systems

HT Cheng 저술 · 2016 · 2424회 인용 — In this paper, we present **Wide & Deep learning**—jointly trained **wide linear models** and **deep neural networks**—to combine the benefits of ...

함께 검색한 항목

Wide and deep Wide and deep code
DeepFM Wide and Deep 구현
Wide and Deep GitHub Wide and Deep Cross feature

Introduction

- 일반적인 검색 랭킹 문제와 유사하게, 추천 시스템에서의 한 가지 과제는 Memorization 과 Generalization 둘 모두를 달성하는 것.
- 사람이 학습하는 과정



갈매기는 날 수 있다



비둘기는 날 수 있다



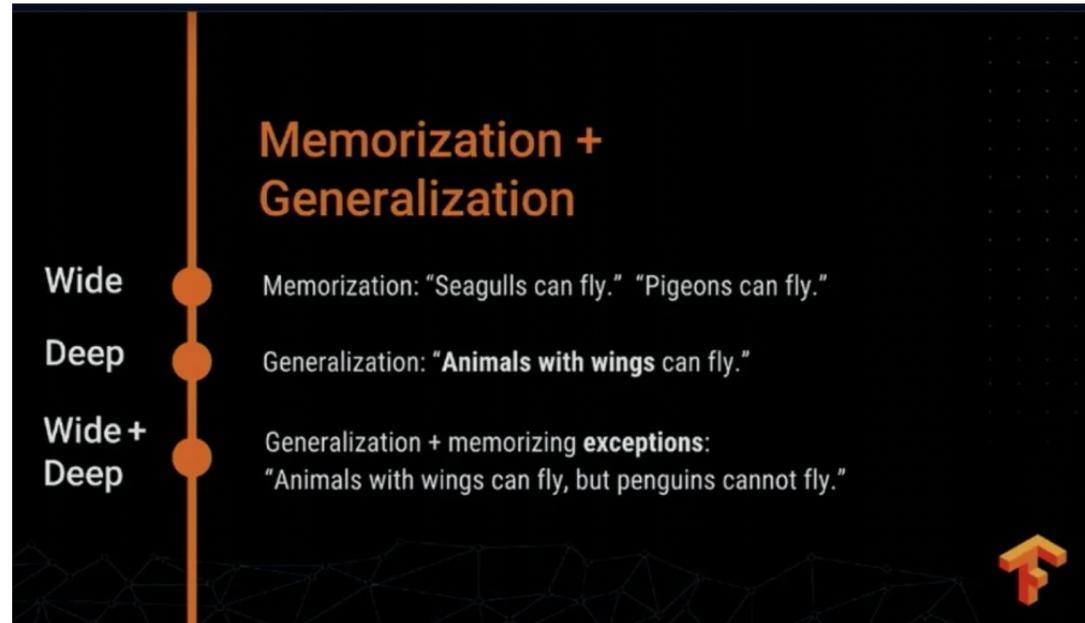
날개 있는 동물
은 날 수 있다



Penguins...
Well, at least
they try.

펭귄..?

Introduction

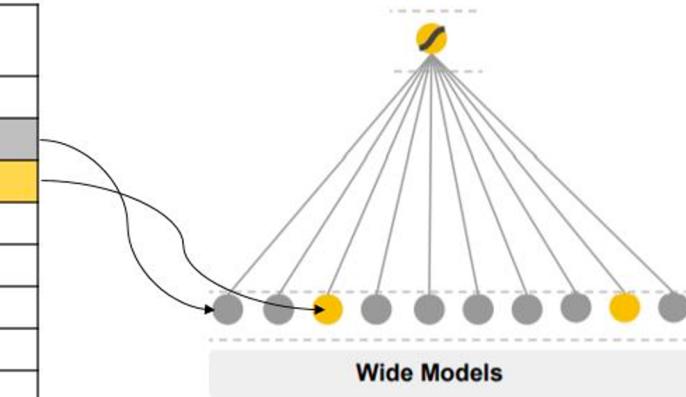


- Memorization 에 특화된 linear wide 모델
- Generalization 에 특화된 non-linear deep 모델
- Wide & Deep 모델

Introduction

- Wide 모델 (Memorization)
 - user 가 설치한 앱 feature 와, user 가 열람한 앱 feature 간의 interaction 을 input 으로 사용
 - 이 때 앱은 A, B, C 총 3개만 존재하며, user 가 설치한 앱과 클릭한 앱은 아래와 같음.
- $\text{user_install_app} = [A, B]$
- $\text{user_impression_app} = [A, C]$

Install	Impression	(Install, Impression)	Install x Impression
A	A	(1,1)	1
A	B	(1,0)	0
A	C	(1,1)	1
B	A	(1,1)	1
B	B	(1,0)	0
B	C	(1,1)	1
C	A	(0,1)	0
C	B	(0,0)	0
C	C	(0,1)	0



Introduction

Wide & Deep / Memorization

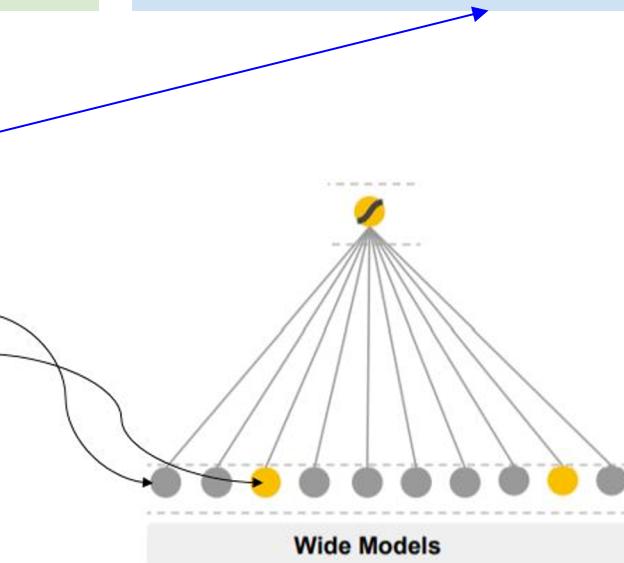
모든 앱의 combination 을 표현하면 총 9가지,
이 때 1이 되는 경우는 4가지

user 가 A 앱을 설치했고, 동시에 C 앱을 봤다면 $(A, C) = (1, 1)$ 로 표현가능하고, 두 값을 곱하면 1이 됨

해당 방식은 1이 되는 모든 경우를 학습하기 때문에 memorization에 강하고, user의 특이 취향이 반영된 niche combination을 학습하기에 탁월한 반면, 0이 되는 pair는 학습이 불가능함

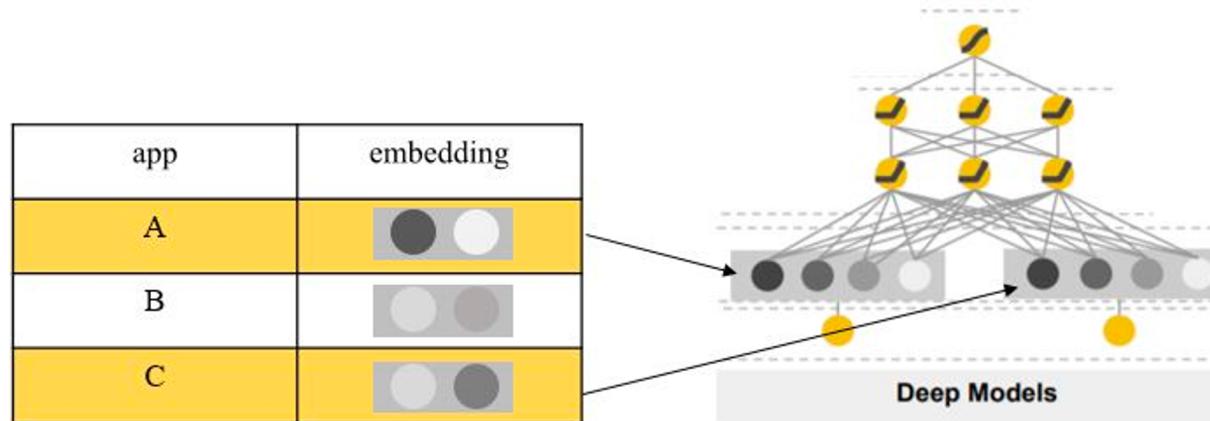
- $\text{user_install_app} = [A, B]$
- $\text{user_impression_app} = [A, C]$

Install	Impression	(Install, Impression)	Install x Impression
A	A	(1,1)	1
A	B	(1,0)	0
A	C	(1,1)	1
B	A	(1,1)	1
B	B	(1,0)	0
B	C	(1,1)	1
C	A	(0,1)	0
C	B	(0,0)	0
C	C	(0,1)	0



Introduction

- Deep 모델 (Generalization)
 - A, B, C 앱을 동일한 임베딩 공간 (여기서는 2차원 가정)에 표현
 - 따라서 (C, B) 처럼 pair 가 없는 관계도 결국 같은 임베딩 공간 내에서 표현이 되기 때문에 학습 가능



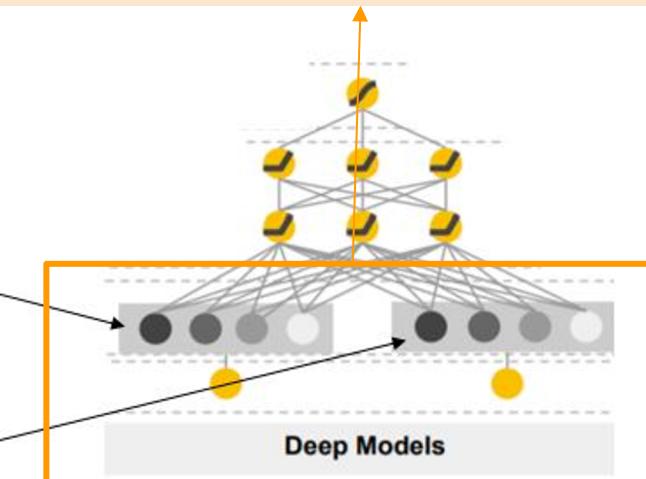
Introduction

- Deep 모델 (Generalization)

- A, B, C 앱을 동일한 임베딩 공간 (여기서는 2차원)에 넣어두면
- 따라서 (C, B)처럼 pair 가 없는 관계에 대해서도 학습 가능

두 feature 간 interaction 은 multi-layer 가 만든 non-linear 공간에서 표현됨. 따라서, generalization 에 강한 반면, 희소한 앱들은 학습이 잘 안됨.

app	embedding
A	[● ○]
B	[○ ○]
C	[○ ●]



Recommender system overview

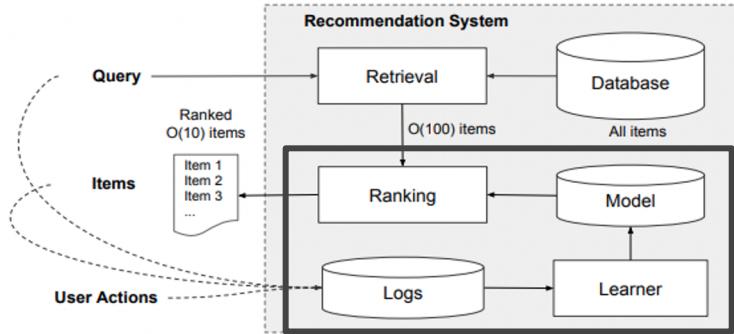


Figure 2: Overview of the recommender system.

- User 의 검색 query 가 들어오면, database 로부터 해당 query 에 적합한 후보 앱들을 retrieval (검색)
- 이어 ranking 알고리즘을 통해 후보 앱들의 점수를 매겨 정렬
 - 점수란, 구체적으로 user 정보 x 가 주어졌을 때, user 가 y 앱에 action 할 확률인 $p(y|x)$
 - feature x : user features (e.g. country, language, demographics), contextual features (e.g. device, hour of the day, day of the week) and impression features (e.g., app age, historical statistics of an app)
- Wide & Deep 은 ranking 알고리즘으로, 여기서는 이런 후보 앱들의 점수를 매기는 데 사용됨

Wide & Deep Learning

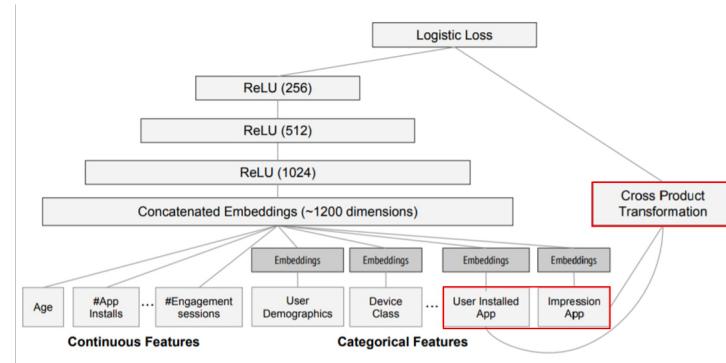
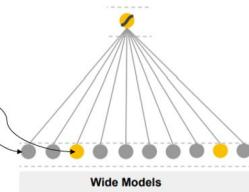
The wide component

- user_installed_app, user_impression_app 두 feature 를 cross-product 한 결과 x 를 input 으로 사용
- Wide 모델 수식

$$y = \mathbf{w}^T \mathbf{x} + b$$

where, $\mathbf{x} = [x_1, x_2, \dots, x_d]$
 $\mathbf{w} = [w_1, w_2, \dots, w_d]$
 b = bias

Install	Impression	(Install, Impression)	Install x Impression
A	A	(1,1)	1
A	B	(1,0)	0
A	C	(1,1)	1
B	A	(1,1)	1
B	B	(1,0)	0
B	C	(1,1)	1
C	A	(0,1)	0
C	B	(0,0)	0
C	C	(0,1)	0



- 논문에서 설명하는 cross-product transformation 는 선형대수의 cross-product 과는 다른 개념

Wide & Deep Learning

Cross product transformation example

- 예를 들어 gender, education level, language 3개의 feature 가 있다고 가정
- 모두 binary 로 가정하며, 다음과 같음

$$\text{gender} = [\text{male}, \text{female}] = [1, 0]$$

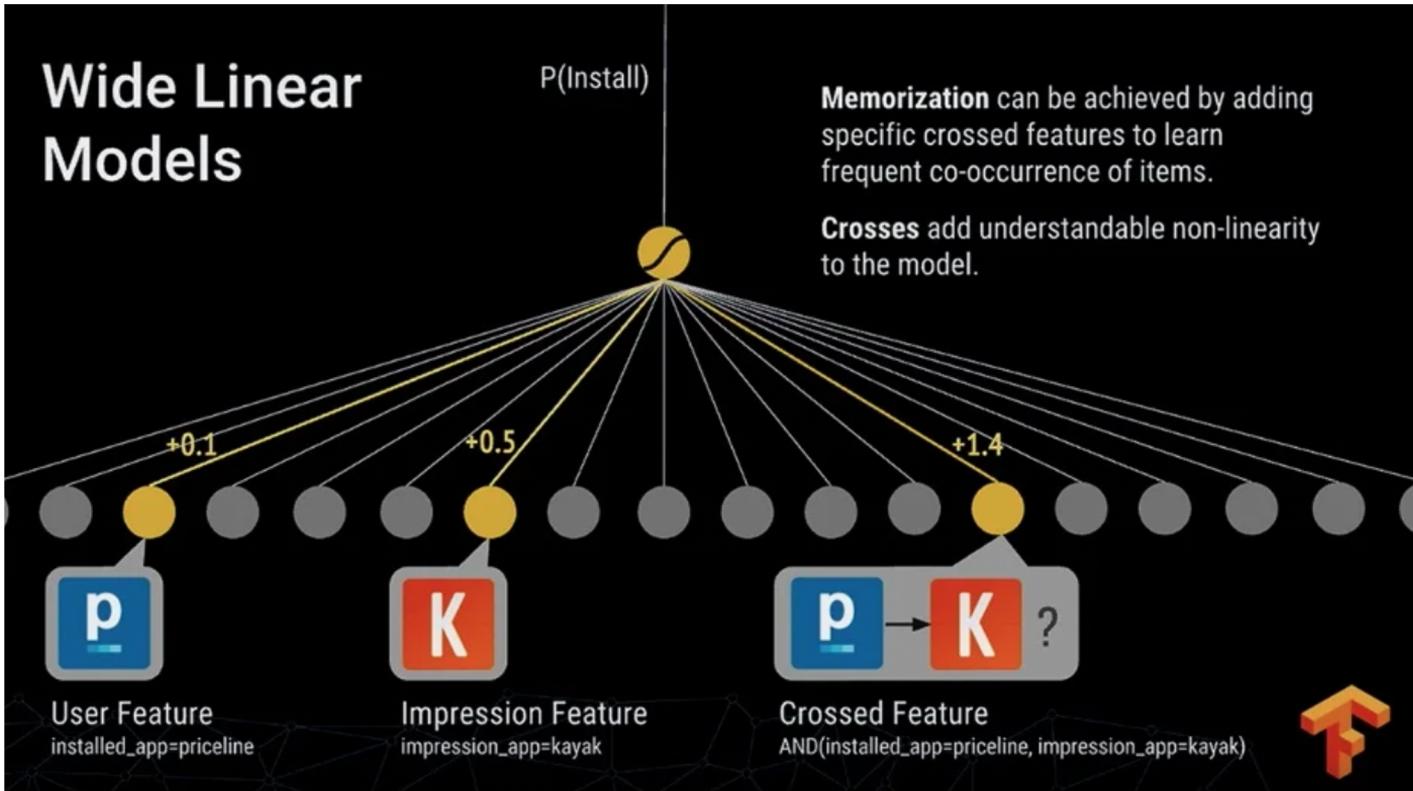
$$\text{education} = [\text{high}, \text{low}] = [1, 0]$$

$$\text{language} = [\text{eng}, \text{kor}] = [1, 0]$$

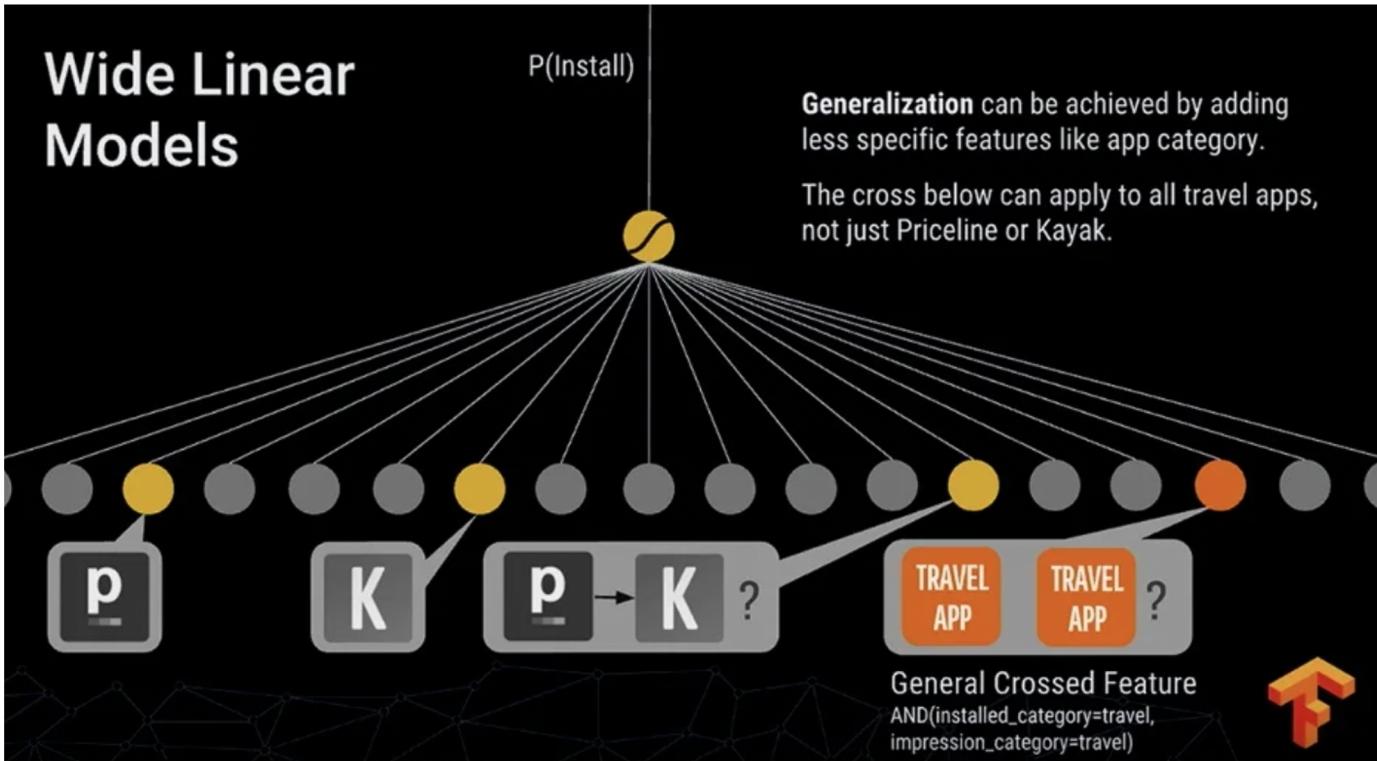
- user A 가 male 이면서 education level 은 low 에 해당하고 언어는 kor 를 사용한다면, user A 벡터 x 는 다음
 $\overbrace{\mathbf{x}}^{\text{벡터 } \mathbf{x}} = [\text{gender}, \text{education}, \text{language}] = [\text{male}, \text{low}, \text{kor}] = [1, 0, 0]$
- 2가지 경우의 feature transformation ($k=2$) 방법이 있다고 가정하 $\phi_2(\mathbf{x})$ 가 gender 와 language 의 조합을 나타내는 transformation 이라면, 다음과 같이 표현 가능

$$\phi_2(\mathbf{x}) = x_1^{c_{21}} x_2^{c_{22}} x_3^{c_{23}} = 1^1 0^0 0^1 = 0, \quad 0^0 \equiv 1$$

Wide & Deep Learning



Wide & Deep Learning



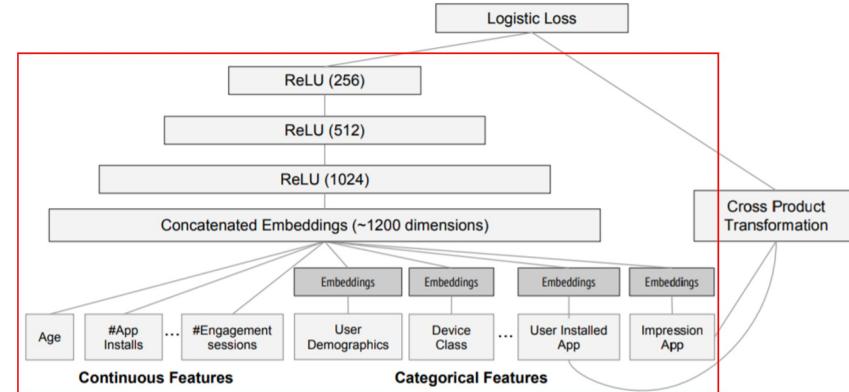
Wide & Deep Learning

The Deep component

- Continuous feature 와 embedding 된 categorical feature 를 concat 하여 input 으로 사용

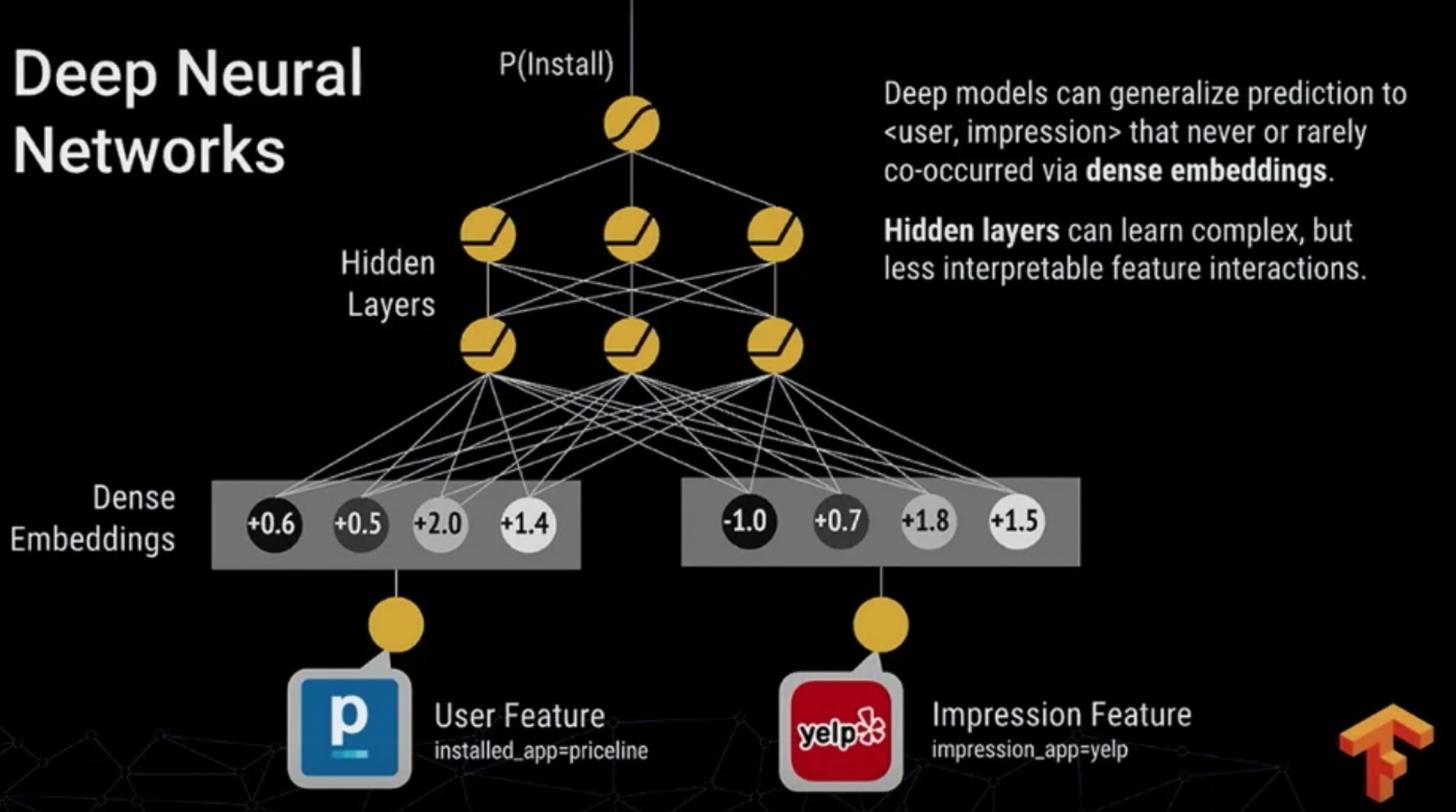
$$\mathbf{a}^{(l)} = f(W^{(l-1)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l-1)})$$

- Activation function : ReLU



Wide & Deep Learning

Deep Neural Networks



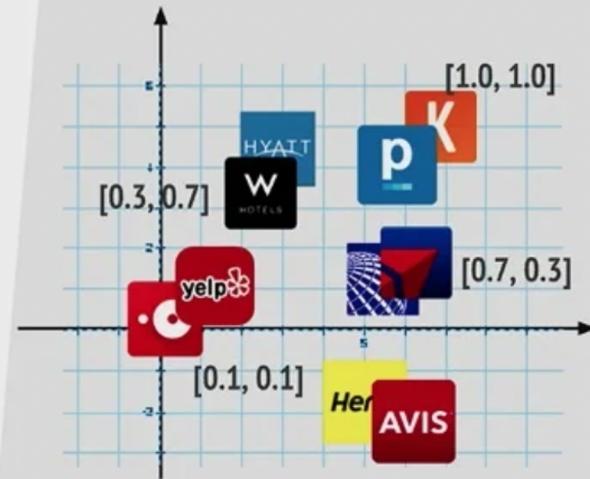
Wide & Deep Learning

Wide Model Weights

		Impression App (N)					
		K	globe	W HOTELS	AVIS	yelp	
User Installed App (M)		p	1.0	0.7	0.7	0.7	?
		HYATT	0.7	1.0	0.3	0.3	?
Hertz	0.7	0.3	1.0	0.3	0.3	0.3	
•○	0.3	0.3	?	0.3	0.3	1.0	

Wide model nonzero weights: 22

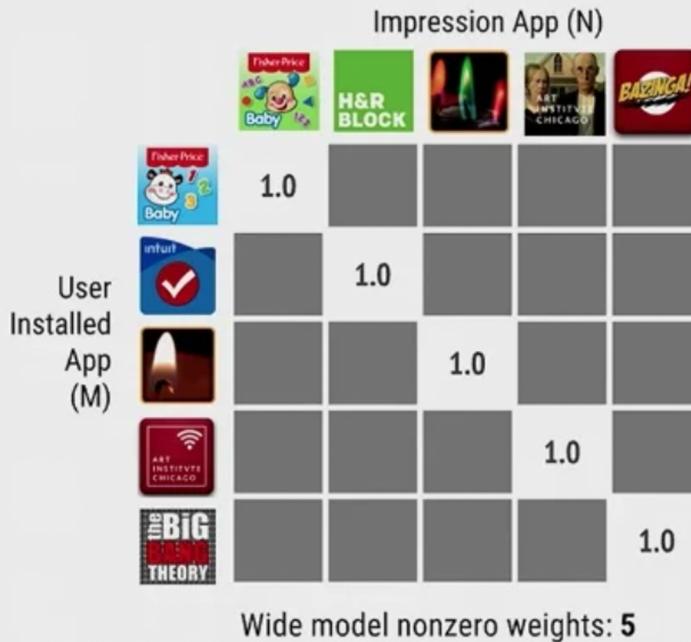
Deep Model Embeddings



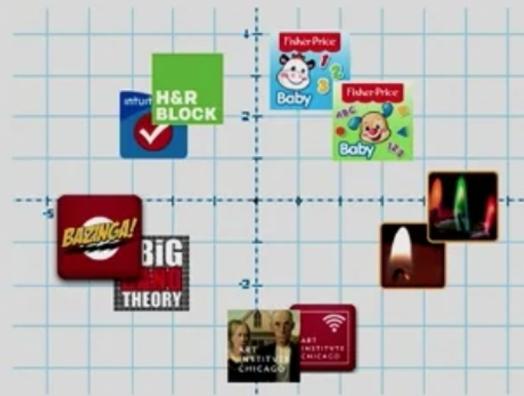
Deep model embeddings:
 $D \times (M+N) = 2 \times (5+5) = 20$

Wide & Deep Learning

Wide Model Weights



Deep Model Embeddings



Deep model embeddings:
 $D \times (M+N) = 2 \times (5+5) = 20$

Wide & Deep Learning

	특징	장점	단점
Memorization (Wide)	<ul style="list-style-type: none">• 히스토리 데이터• 직접 연관 데이터	<ul style="list-style-type: none">• 간단함• 주제 식별	<ul style="list-style-type: none">• 학습 데이터에 있는 것들만 학습 가능• 뻔한 추천
Generalization (Deep)	<ul style="list-style-type: none">• Dense vector embedding• 새로운 feature 조합 탐색	<ul style="list-style-type: none">• 다양성 개선• 추가 feature engineering이 필요없음	<ul style="list-style-type: none">• Sparse & High-rank 의 경우, 추천 성능이 떨어짐• 관련 없는 추천

Wide & Deep Learning Q1.

Wide model 의 input 입력 순서는 정의되어 있는건가?

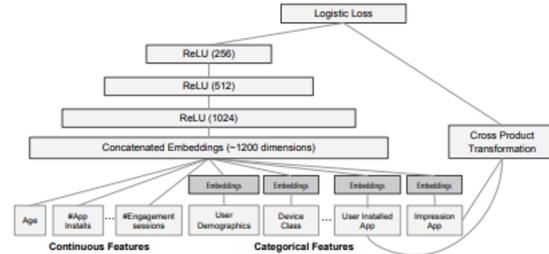


Figure 4: Wide & Deep model structure for apps recommendation.

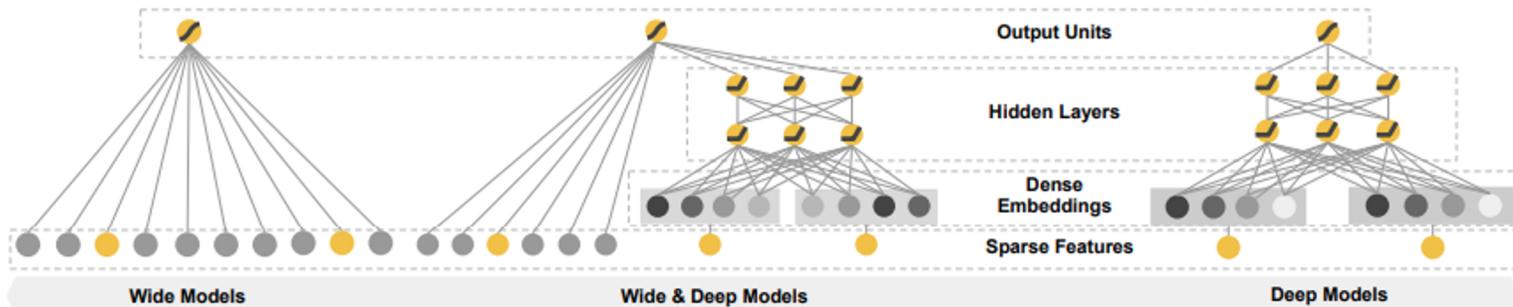


Figure 1: The spectrum of Wide & Deep models.

Wide & Deep Learning Q2.

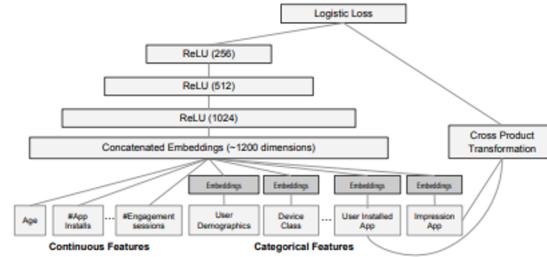


Figure 4: Wide & Deep model structure for apps recommendation.

Continuous features 의 경우 embedding 을 하지 않고 어떻게 concat 하는거지?

Vocabularies, which are tables mapping categorical feature strings to integer IDs, are also generated in this stage. The system computes the ID space for all the string features that occurred more than a minimum number of times. Continuous real-valued features are normalized to $[0, 1]$ by mapping a feature value x to its cumulative distribution function $P(X \leq x)$, divided into n_q quantiles. The normalized value is $\frac{i-1}{n_a - 1}$ for values in the i -th quantiles. Quantile boundaries

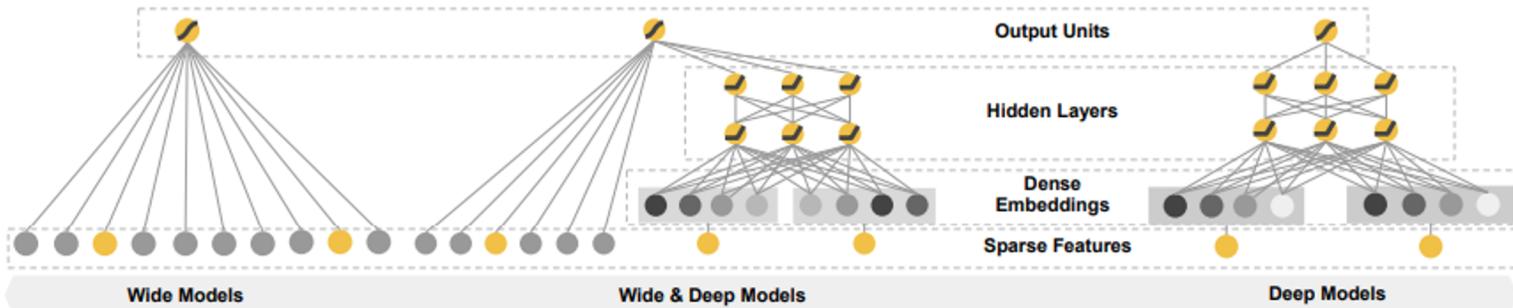


Figure 1: The spectrum of Wide & Deep models.

Wide & Deep Learning

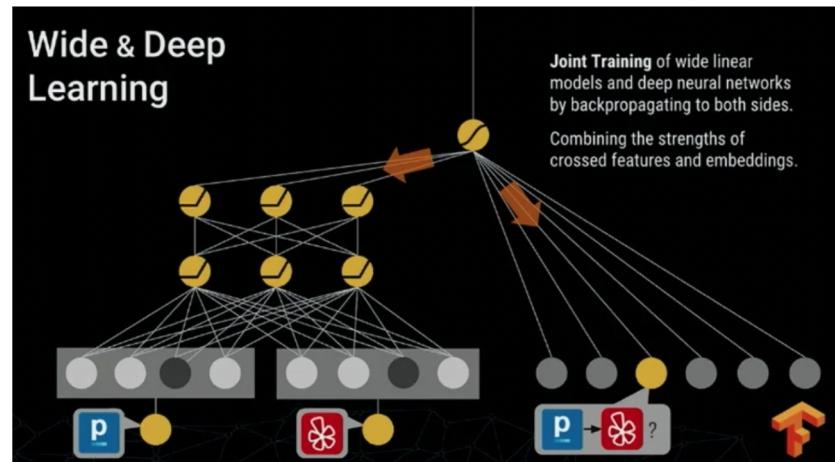
Joint training of wide & deep model

- 여러 개의 모델을 결합하는 ensemble 과는 달리, output 의 gradient 를 wide 와 deep 모델에 동시에 backpropagation
 - wide model : optimizer 로 online learning 방식인 Follow-the-regularized-leader (FTRL) 알고리즘 (L1 regularization)
 - deep model : optimizer 로 Adagrad
- 하나의 공통 로지스틱 손실 함수

The combined model is illustrated in Figure 1 (center). For a logistic regression problem, the model's prediction is:

$$P(Y = 1|\mathbf{x}) = \sigma(\mathbf{w}_{wide}^T \mathbf{x} + \phi(\mathbf{x}) + \mathbf{w}_{deep}^T \mathbf{a}^{(l_f)} + b) \quad (3)$$

where Y is the binary class label, $\sigma(\cdot)$ is the sigmoid function, $\phi(\mathbf{x})$ are the cross product transformations of the original features \mathbf{x} , and b is the bias term. \mathbf{w}_{wide} is the vector of all wide model weights, and \mathbf{w}_{deep} are the weights applied on the final activations $\mathbf{a}^{(l_f)}$.



Wide & Deep Learning

Joint training vs Ensemble

- 결합 방식
 - Ensemble 은 개별 모델들이 서로 모르는채로 따로 훈련을 하고, 이들의 예측은 훈련 시간이 아닌 추론 시간에만 결합됨
 - 이와 반대로, Joint training 은 training 과정에 합계의 가중치 뿐만 아니라 wide&deep part 를 동시에 고려하여 모든 파라미터들을 동시에 최적화.
- 모델 사이즈
 - Ensemble 의 경우, 훈련이 상호배제로 수행되기 때문에, 양상을 작업에서 합리적인 정확도를 달성하기 위해서 각 개별 모델 크기가 일반적으로 더 커야 함 (e.g. 더 많은 feature 와 transformations)
 - Joint training 의 경우, wide part 는 full size wide 모델보다 적은 수의 cross-product feature transformations 로 deep part 약점을 보완하기만 하면 됨

System implementation

앱 추천 파이프라인 구현은 다음과 같이 3단계로 구성됨

- 데이터 생성 (data generation)
- 모델 훈련 (model training)
 - Warm-starting system
- 모델 서빙 (model serving)

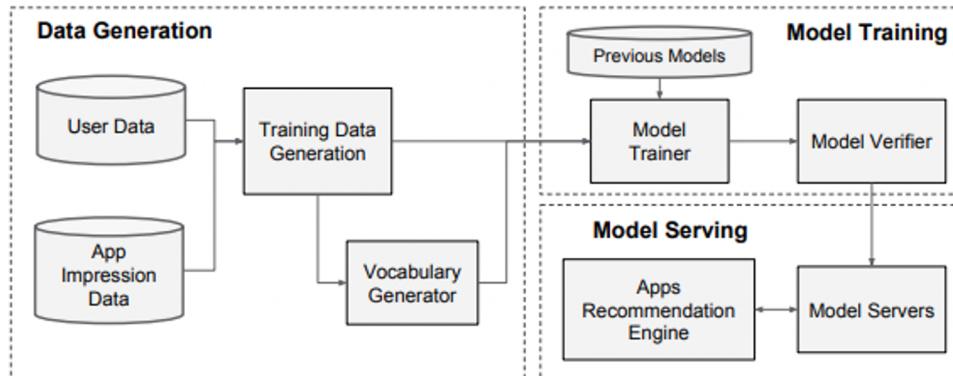


Figure 3: Apps recommendation pipeline overview.

Experiments & Results

실제 추천 시스템에서 Wide & Deep Learning 의 효과를 평가하기 위해, 실시간 실험 진행하였으며, 두 가지 측면에서 시스템 평가

- 앱 획득 (App Acquisitions)
- 서빙 성능 (Serving Performance)
- **App Acquisitions**
 - 3주 동안 A/B testing 프레임워크에서 실시간 온라인 실험 수행
 - 대조군 (Control) 그룹의 경우, 유저의 1%가 무작위로 선택했고, 이전 버전의 랭킹 모델이 생성한 추천을 제시. 이는 rich 한 cross-product feature transformations 를 이용하여 고도로 최적화된 wide-only 로지스틱 회귀 모델임.
 - 실험군 그룹의 경우, 유저의 1%가 동일한 features 집합으로 훈련된 wide&deep 모델이 생성한 추천 제시

**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%

Experiments & Results

- **Serving performance**
 - 상용 모바일 앱 스토어가 직면한 높은 수준의 트래픽에 높은 처리량과 낮은 대기 시간으로 서비스를 제공하는 것은 어려운 일임.
 - 트래픽이 최고점일 때, 추천 서버는 초당 1천만 개 이상의 앱에 점수를 매김
 - 단일 스레딩을 이용하여, 단일 배치로 모든 후보에 점수를 매기려면 31ms 가 걸림
 - 멀티스레딩을 구현하고, 각 배치를 더 작은 크기로 분할하여 table 2에 보이는 것처럼 클라이언트 측 지연 시간을 14ms (서비스 오버헤드 포함) 로 크게 줄임

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

Conclusion

- Memorization 과 Generalization 은 추천 시스템에서 모두 중요함
- Wide 선형 모델은 Cross-product feature transformations 를 사용하여 sparse feature interactions 를 효과적으로 기억(memorize) 할 수 있는 반면, deep neural networks 는 저차원 임베딩을 통해 이전에 관찰되지 않은 feature interactions 을 일반화 (generalize) 할 수 있음
- 모델의 두 타입의 강점을 결합한 Wide & Deep Learning 프레임워크
- 대규모 상업용 앱스토어인 Google Play 의 추천 시스템에서 프레임워크를 제작 및 평가함
- 온라인 실험 결과에 따르면, Wide & Deep 모델이 Wide-only 와 Deep-only 모델과 비교하여 App Acquisitions 가 크게 개선됨
- 오픈소스 제공 (TensorFlow)
 - <https://paperswithcode.com/paper/wide-deep-learning-for-recommender-systems>

Thank You