Wide & Deep Learning for Recommender Systems

디지털애널리틱스 3기 김민엽

CONTENTS

01

논문 내용요약

01. 논문내용 요약

모델 설명
(구글 플레이 스토어 앱 추천 엔진)

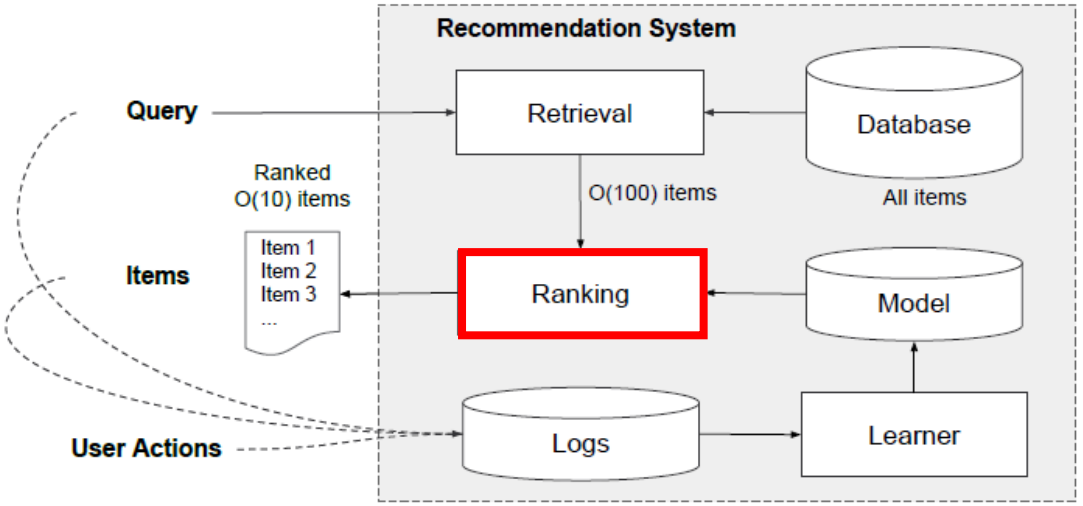


Figure 2: Overview of the recommender system.

- 사용자의 쿼리가 들어오면, 검색(retrieval) 시스템은 데이터베이스로부터 해당 쿼리에 적합한 후보 앱들을 반환
- 이어서, 랭킹 시스템을 통해 후보 앱들의 점수를 매겨 정렬
- 여기서 점수란 사용자 정보 x가 주어졌을 때, 사용자가 y 앱에 action할 확률인 P(y|x)를 구하는 것
- 본 논문에서는 랭킹 시스템에 Wide & Deep 알고리즘을 사용한 모델을 제안

01. 논문내용 요약

Wide & Deep Model 전체 구조

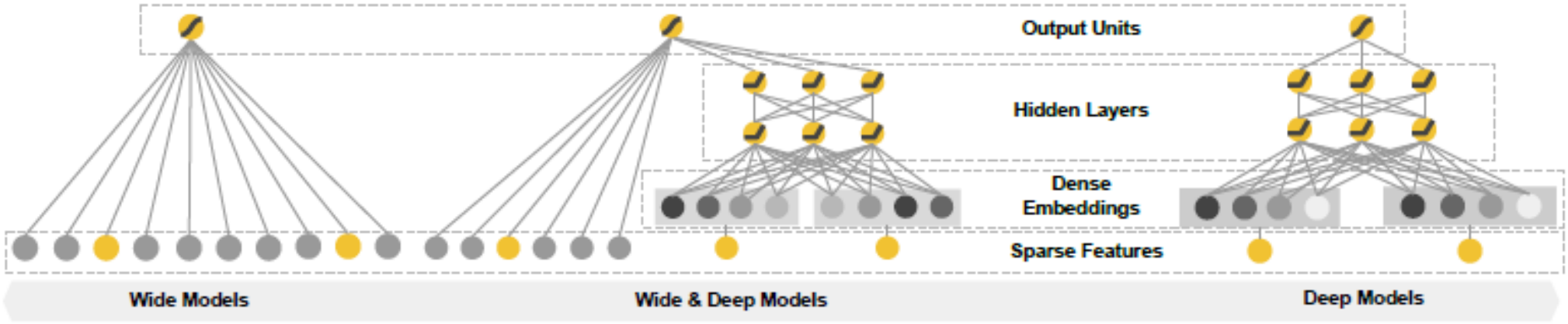
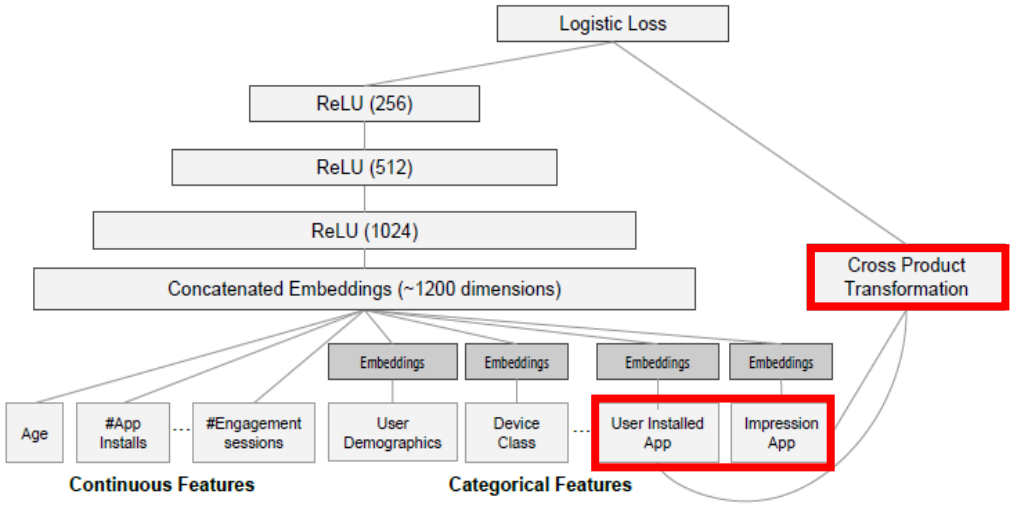


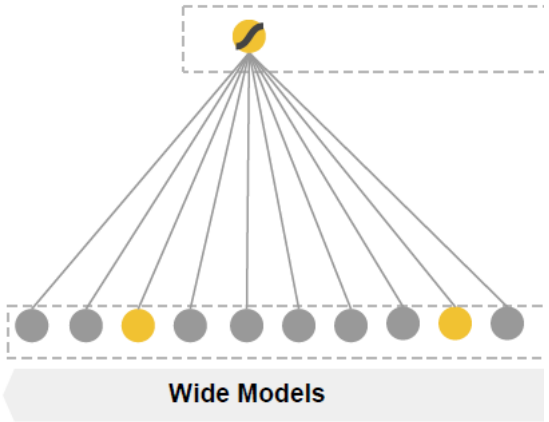
Figure 1: The spectrum of Wide & Deep models.

- 본 논문에서는 선형 모델과 신경망을 결합해서 학습함에 따라 암기(Memorization)와 일반화(Generalization)를 한 모델에서 달성해서 각각의 이점을 결합

01. 논문내용 요약

Wide Component



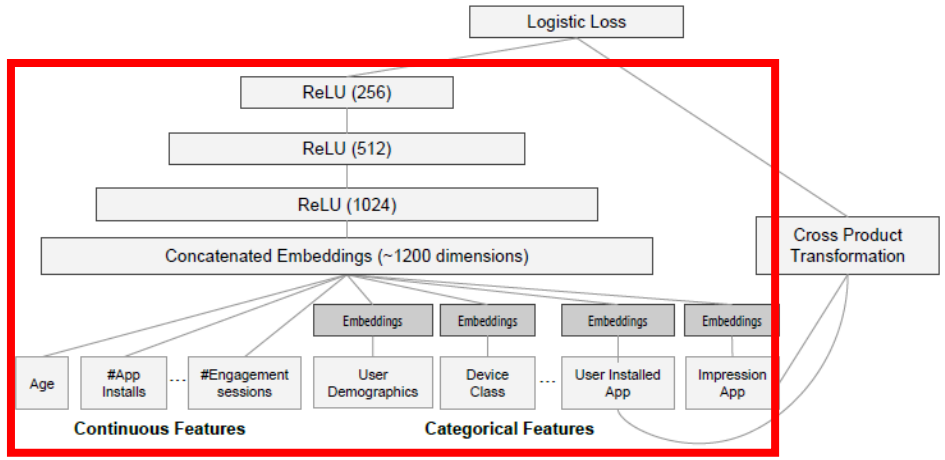


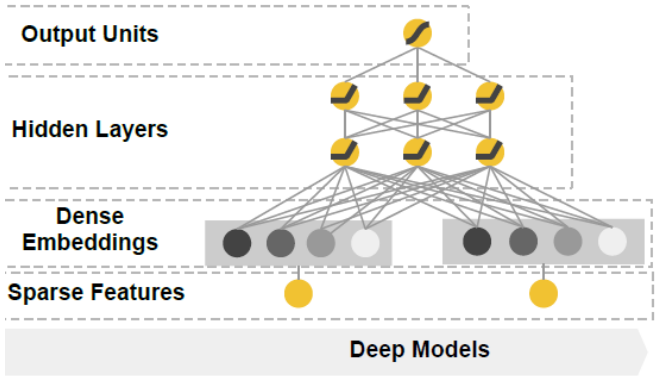
$$y = w^T x + b$$

- **암기**(Memorization)는 동시에 발생하는 **아이템** 또는 **피처**를 학습하고 사용 가능한 과거 데이터로부터 상관관계를 추출하는 작업
- 암기에 기반한 추천은 **사용자가 실행한 행동과 관련된 아이템과 직접적으로 관련**
- Wide 모델에서 **선형 모델**을 사용하며, 동시 출현 빈도를 표현하는 **cross-product**를 통해 적은 파라미터로도 모든 피처의 조합을 기억할 수 있어 **wide** 하고 **sparse**한 정보 **기억**에 효과적
- Wide 모델의 단점:
  - 학습데이터에 나타나지 않은 쿼리-아이템 쌍은 학습하지 못함
  - 뻔한 추천을 함
  - 특성 간의 Cross-product transformation이라는 무거운 피처엔지니어링이 필요함

01. 논문내용 요약

Deep Component



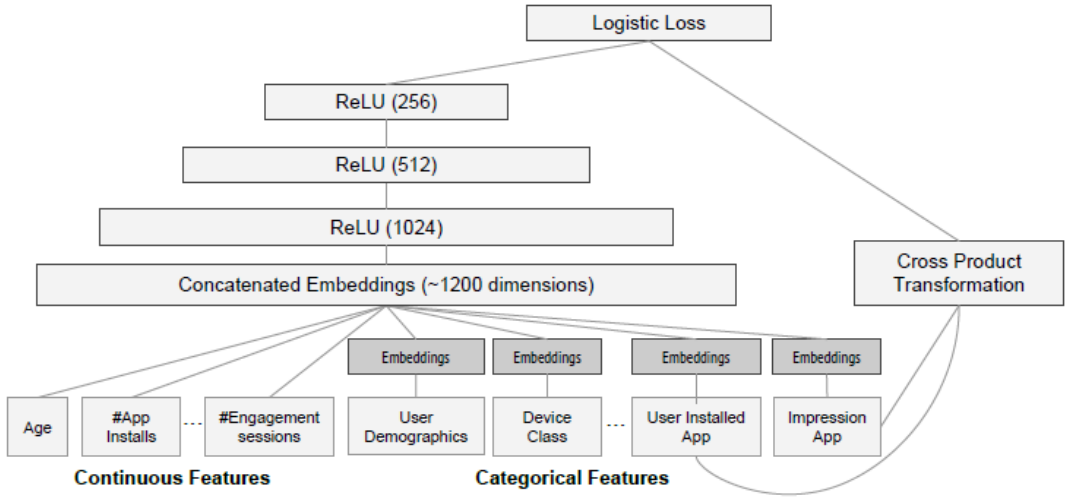


$$a^{(l+1)} = f(W^{(l)}a^{(l)} + b^{(l)})$$

- 일반화는 과거에 전혀 혹은 드물게 발생한 **새로운 변수들의 조합**을 탐구하여 추천의 **다양성**을 향상시킴
- **Embedding based** 모델인 **DNN**(Deep Neural Network)을 사용함
- 저차원 Embedding 벡터의 학습을 통해 **이전에 나타나지 않은 변수들에 대해서도 연관성을 학습**시킬 수 있음
- Cross-product가 필요하지 않으므로, **피처엔지니어링에 적은 노력**이 필요함
- 범주형 변수 각각은 저차원 임베딩 벡터로 변환 되어 임베딩 벡터 a가 input으로 사용되고, hidden layer로 feed 되어짐
- Deep 모델의 단점:
  - **sparse**하고 **high-rank**인 경우에는 쿼리-아이템 행렬에 대해 저차원 표현으로 학습하는 것이 어려움
  - 즉, 실제로 존재할 수 없거나 희소한 관계에 대해서도 **지나친 일반화**를 하여 **관련이 적은 추천**이 이루어질 수 있음

01. 논문내용 요약

Joint Training of Wide & Deep Model



- Wide 와 Deep의 구성요소는 예측치로서 각각의 output log odds의 가중치 합으로 결합되고, 그것은 joint training을 위해 하나의 공통의 로지스틱 함수로 feed 됨
- Joint training은 여러 개의 모델을 각각 따로 학습 후에 결합하는 앙상블과 달리, 두 모델을 동시에 학습하며 각 모델의 약점을 보완함
- 논문에서는, Wide 모델에는 FTRL (Follow the regularized leader) 알고리즘을 사용하고, Deep 모델에는 Adagrad를 옵티마이저로 사용함
- 아래 식은 wide and deep 모델의 prediction으로, P(y = 1|x)는 특정 앱을 설치할 확률
  - 각 모델에서 나온 결과를 더해 sigmoid 함수를 통과시킨 결과가 최종 output

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

01. 논문내용 요약

Experiment Results

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%

- 실험은 3주간의 실시간 온라인 A/B 테스트를 진행되었음
- Wide-only, Deep-only, Wide & Deep 3가지 모델을 각각 전체의 1%에 해당하는 사용자에게 랜덤으로 적용하였음
- 실험 결과
  - Wide & Deep 모델이 Wide-only 모델에 비해서 3.9% 높은 앱 설치율을 보임
  - Wide & Deep 모델이 Deep-only 모델에 비해서 1.0% 높은 앱 설치율을 보임

02

비평 (논문의 장,단점과 개선점)

02. 비평

장점

- 기존 Wide 방법과 Deep 방법의 단점을 커버하고 각 방법의 장점을 합쳐서 성능을 개선시킨 것
- 실제로 널리 사용되고 있는 구글 플레이에 적용해서 철저하게 검증을 하고 좋은 성능을 보여주어서, 연구의 실효성을 입증한 점
- 구글플레이 스토어 앱의 전반적인 추천엔진을 그림으로 설명해줘서 어떤 식으로 추천이 이루어지는지 보여주는 점이 좋았음
- 텐서플로우 공식 홈페이지에 오픈소스화 해서 실제로 사용해 볼 수 있음

02. 비평

단점

- 논문에서는 단순히 wide 모델과 deep 모델의 output을 더한 후 sigmoid를 통해 output을 냄

- Wide 구성요소에 cross-product feature를 만들기 위해서는 여전히 피처 엔지니어링에 수고가 많이 들어감

03

추가 연구 아이디어

03. 추가 연구 아이디어

단점을 개선하기 위한 추가 연구 아이디어

- Wide 모델과 Deep 모델의 output 간의 비율을 최적화 시켜서 좀 더 좋은 성능을 이끌어낼 수 있을 것 같음

- Wide 에 사용되는 모델을 선형 모델이 아닌 FM 또는 다른 모델을 사용해 볼 수 있을 것 같음 → 피처엔지니어링에 들어가는 노력을 크게 줄일 수 있지 않을까?

  - FM(Factorization Machine) 장점: 구현의 용이성과 도메인 특성에 크게 의존하지 않는 피처엔지니어링

Thank you