Wide&Deep Learning for Recommender Systems 리뷰

소프트웨어학부 20206802 임도연

목차

Wide&Deep Learning 등장 계기

추천시스템에서의 과제는 Memorization과 Generalization을 모두 달성하는 것

- Memorization

: 과거 데이터에서 사용할 수 있는 상관관계 + item과 feature의 동시 빈발도를 학습

➜ 더 주제적이고 유저가 수행한 item과 연관성이 높은 추천을 함.

- Generalization

: 과거에 발생하지 않았던 새로운 feature 조합을 탐색

➜ Memorization 기반 추천보다 추천된 아이템의 다양성을 향상시킴.

Wide&Deep Learning 등장 계기

- cross-product feature transformations의 wide 집합을 통한 Memorization of feature interaction은 효과적임.

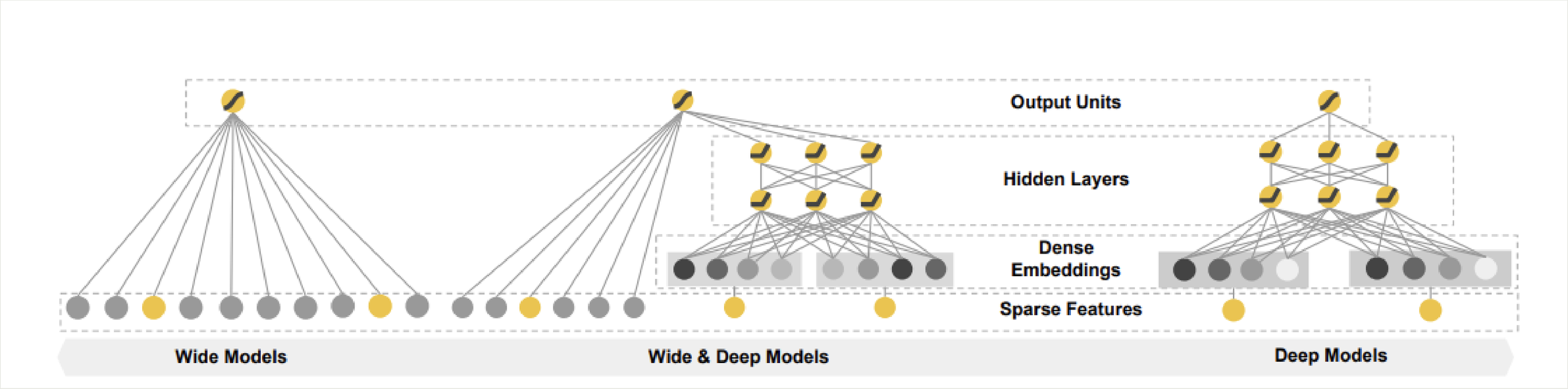
➜ 하지만, 훈련 데이터에 나타나지 않은 쿼리-아이템 feature pair의 경우 일반화할 수 없다.

- 임베딩 모델(factorization machines, deep neural networks)의 경우 feature engineering을 이용해 각 쿼리와 아이템 feature를 저차원의 dense한 임베딩 벡터로 학습한 후 전에 보지 못했던 feature pair를 일반화할 수 있음.

➜ 하지만, 쿼리-아이템 matrix가 희소하거나 high-rank일 때 관련성 없는 추천을 할 수 있다.

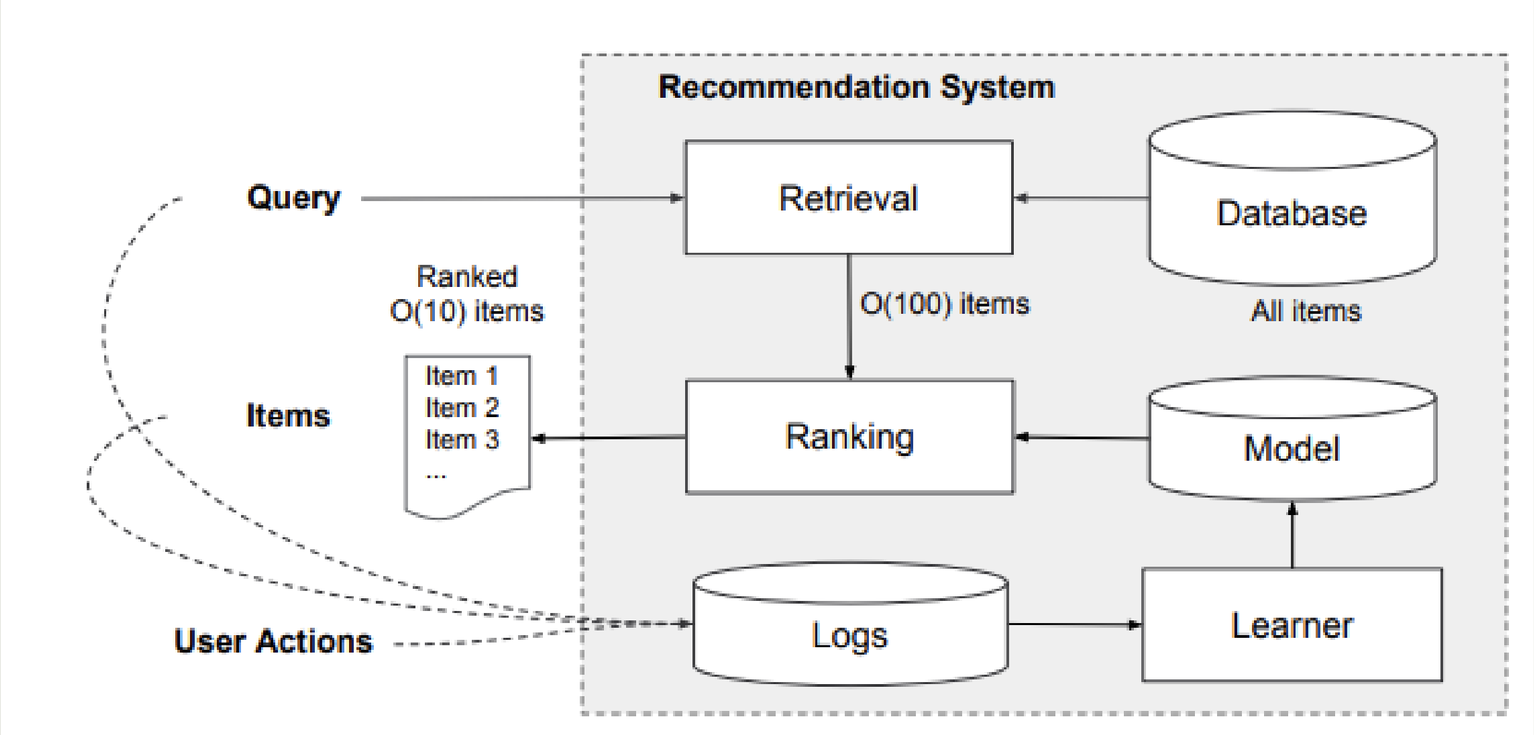
Wide&Deep Learning 등장 계기

- 앞선 경우 cross-product feature transformations는 더 적은 파라미터로 예외 규칙을 기억하는 방법을 이용
- 신경 네트워크 요소와 선형 모델 요소를 결합해 훈련함으로써 memorization과 generalization을 모두 달성할 수 있는 wide&deep learning framework가 탄생함.



추천시스템 기본 개요

- 쿼리를 받기 위한 첫번째 단계는 검색(retrieval)
- 머신러닝 모델과 사람 정의 규칙을 통해 리스트를 반환함.
- 후보군을 줄인 후 ranking 단계로 등급을 매김.



Wide Component

- wide component는 일반화된 선형 모델 $y = \mathbf{w}^T\mathbf{x} + b,$

  〉 y : 예측

  〉 x : d feature들의 벡터

  〉 w : 모델 파라미터

  〉 b : 편향

- 가장 중요한 transformations는 cross-product transformation이다.

$$\phi_k(\mathbf{x}) = \prod_{i=1}^{d} x_i^{c_{ki}} \quad c_{ki} \in \{0, 1\}$$

> C ki는 boolean 변수, i번째 faature가 속해있다면 1

➜ 상호작용을 포착하고, 일반화된 선형 모델에 비선형성을 추가

Deep Component

- 범주형 feature의 경우 sparse하고 고차원적이기 때문에 저차원적이고 dense한 실수 벡터로 변환, 이를 임베딩 벡터라고 부름.
- 임베딩 벡터는 처음 무작위로 초기화된 다음 모델 훈련 동안 최종 손실 함수를 최소화하기 위해 갱신됨.
- 이 임베딩 벡터는 신경망의 hidden layer에 입력됨.

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

hidden layer 계산식

> a : 활성
> l : 계층 수
> W : l번째 계층의 모델 가중치
> b : 편향
> f : 활성 함수

Wide&Deep Model

- Wide component와 Deep component는 output 로그 odds 가중합을 하나의 로지스틱 손실 함수에 입력됨.

- 앙상블

: 개별 모델들이 서로 모르는 상태로 훈련하고 예측은 추론 시간에만 결합됨.

: 상호배제로 수행되기 때문에 각 개별 모델의 크기가 더 커야함(많은 feature or transformations)

Wide&Deep Model

- 공동 훈련

: 훈련 시간에 모든 파라미터들을 동시에 최적화함

: wide part는 적은 수의 cross-product feature transformations로

deep part의 약점을 보완

위 모델은 미니 배치 확률적 최적화를 사용하여 모델 output의 gradients를 동시에 역전파함.

Wide&Deep Model

- 이 논문에서는 FTRL 알고리즘을 사용했으며 wide part에서는 optimizer로 L1 Regularization을 deep part에서는 AdaGrad를 사용함.

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

〉 y : 바이너리 클래스 레이블

〉 σ : 시그모이드 함수

〉 $\phi(\mathbf{x})$: origin feature x의 cross-product transformations

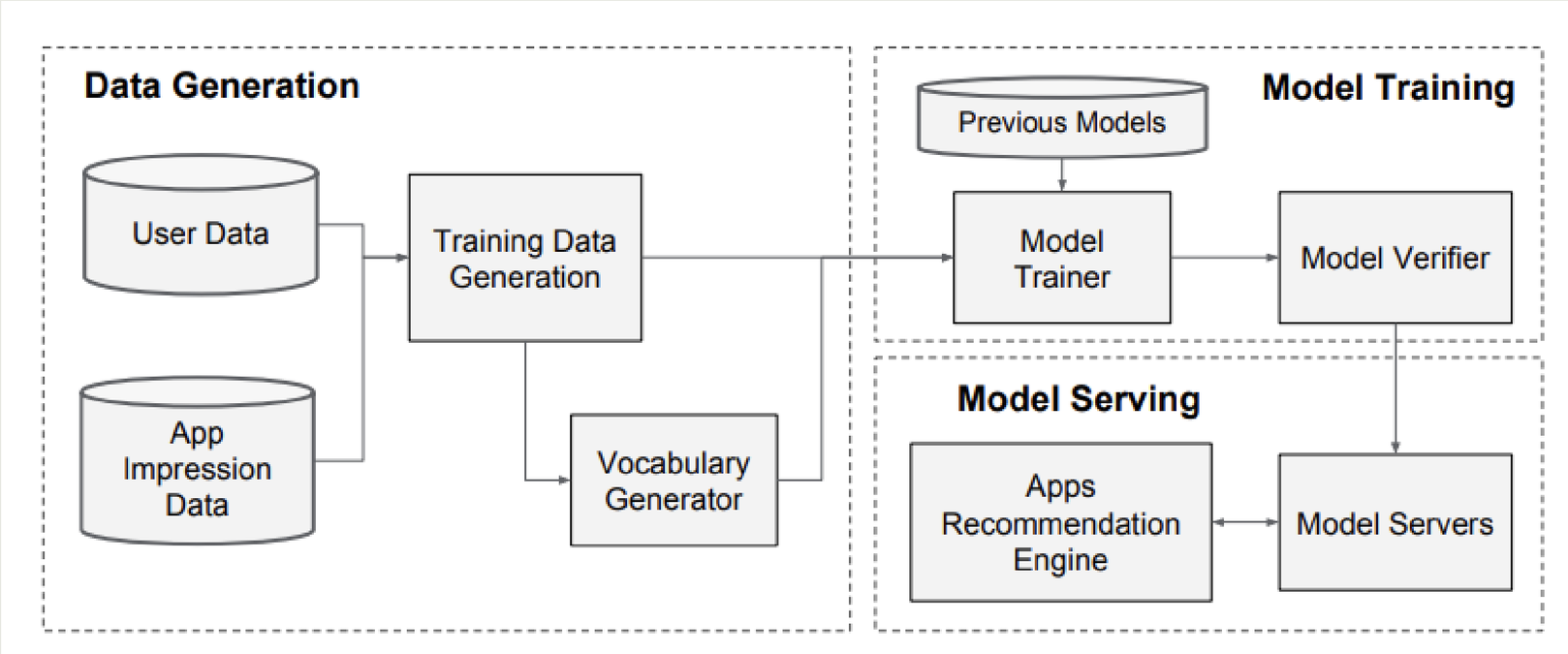
〉 $\mathbf{w}_{wide}^T$: wide 모델의 모든 가중치 벡터

〉 $\mathbf{w}_{deep}^T$: 최종 활성에 적용된 가중치들

앱 추천 파이프라인 구현

앱 추천 파이프라인 구현

- 데이터 생성
- 모델 훈련
- 모델 서빙



Data Generation

- period 내, 유저와 앱 impression 데이터가 훈련 데이터 생성을 위해 사용됨
- app acquisition : 만약 앱이 설치되었다면, 1 아니면 0으로 값을 설정
- 연속적인 실수 feature x를 누적 분포 함수에 매핑 해 [0,1]로 정규화함.
- 정규화된 값은 i번째 분위 수에서 $\frac{i-1}{n_q-1}$ 이 됨.

Data Training

- 훈련하는 동안 input 계층은 훈련 데이터와 어휘를 입력으로 받으며 sparse하고 dense한 feature를 생성

- wide component : impression 앱과 유저가 설치한 앱의 cross-product transformations으로 구성
- deep component : 32차원의 임베딩 벡터는 각 카테고리 feature에 대해 학습

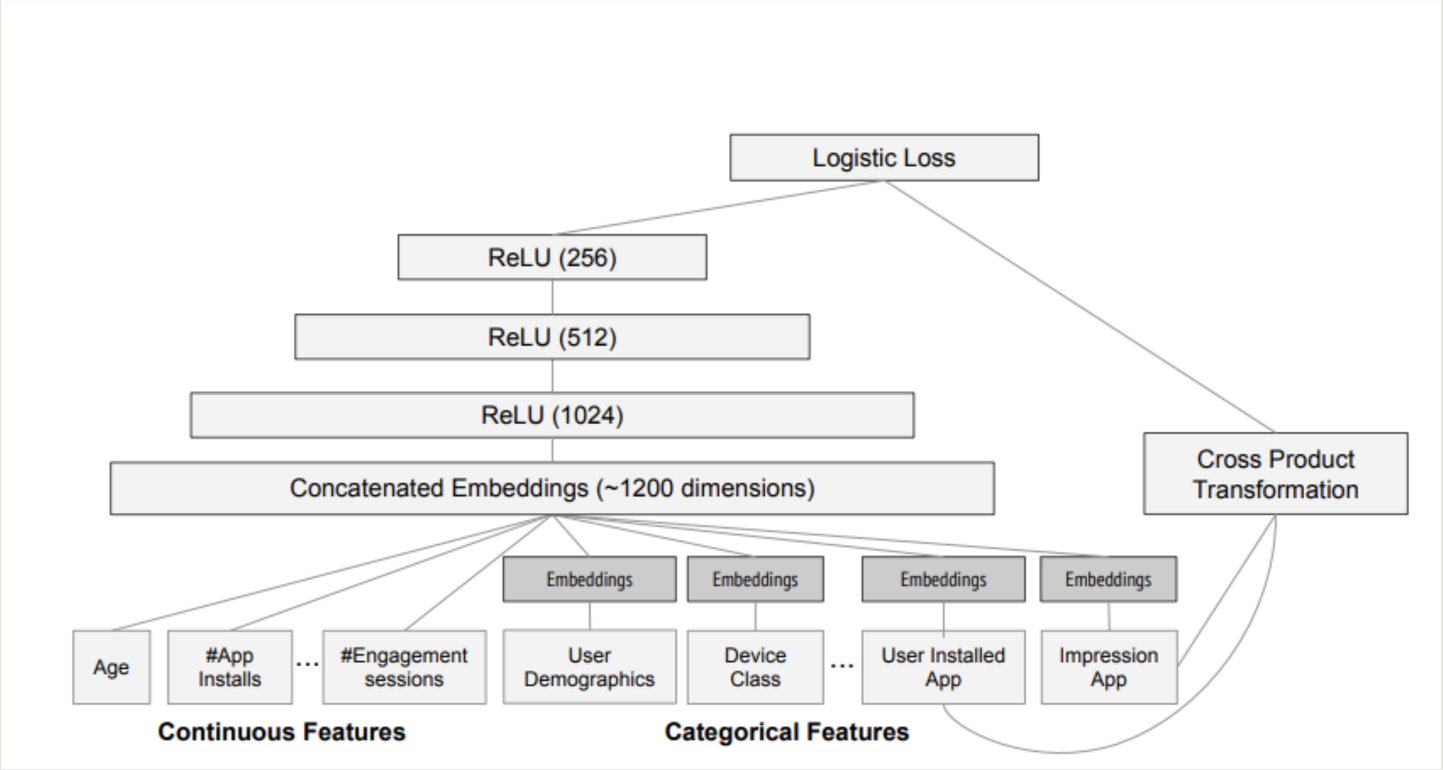
➜ dense한 feature를 이용한 모든 임베딩을 이어 붙여 dense한 벡터를 내놓음

Model Training

- 하지만, 새로운 훈련 데이터 집합이 도착할 때마다 모델을 다시 traning해야함.

➜ 비용이 많이 들고 시간이 많이 지연되는 문제가 생김

➜ warm-starting 방식을 이용해 이전 선형 모델의 가중치와 임베딩을 가지고 새로운 모델을 초기화 함.



Model Serving

- 앱 검색 시스템과 각 앱에 점수를 매긴 유저 feature들로부터 앱 후보 집합을 받음.
- 이 후보들을 점수를 통해 등급을 매긴 후 유저에게 앱을 보여줌.
- 모든 후보 앱에 점수를 매기는 방식이 아닌 더 작은 배치를 병렬로 실행하여

병렬 멀티스레딩 방식을 이용해 성능을 최적화했음

결론

wide-only 로지스틱 회귀 모델을 이용한 대조군 그룹과 wide&deep 모델을 이용한 실험군 집단에게 온라인 실험을 진행

➜ 그 결과 Acquisition Gain에서 두드러진 향상을 보임

➜ 또한 Offline AUC의 경우도 높은 것을 확인할 수 있음(온라인의 경우 데이셋의 레이블이 고정된 오프라인과 달라 generalization과 memorization을 혼합해 새로운 추천을 생성할 수 있음)

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

결론

모바일 앱 스토어의 경우 높은 수준의 트래픽에 높은 처리량과 낮은 대기 시간으로 서비스를 제공하는 것이 어려움

➜ 멀티스레딩을 구현해 각 배치를 더 작은 크기로 분할하여 지연 시간을 많이 줄임.

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

감사합니다