영입문제 Jukebox 풀이

Jukebox 솔루션

추천기술3셀 이현성 iggy.ll

kakao

문제 소개

오토인코더를 사용한 풀이

자잘한 팁

Q&A

문제 소개

오토인코더를 사용한 풀이

자잘한 팁

Q&A

문제 소개

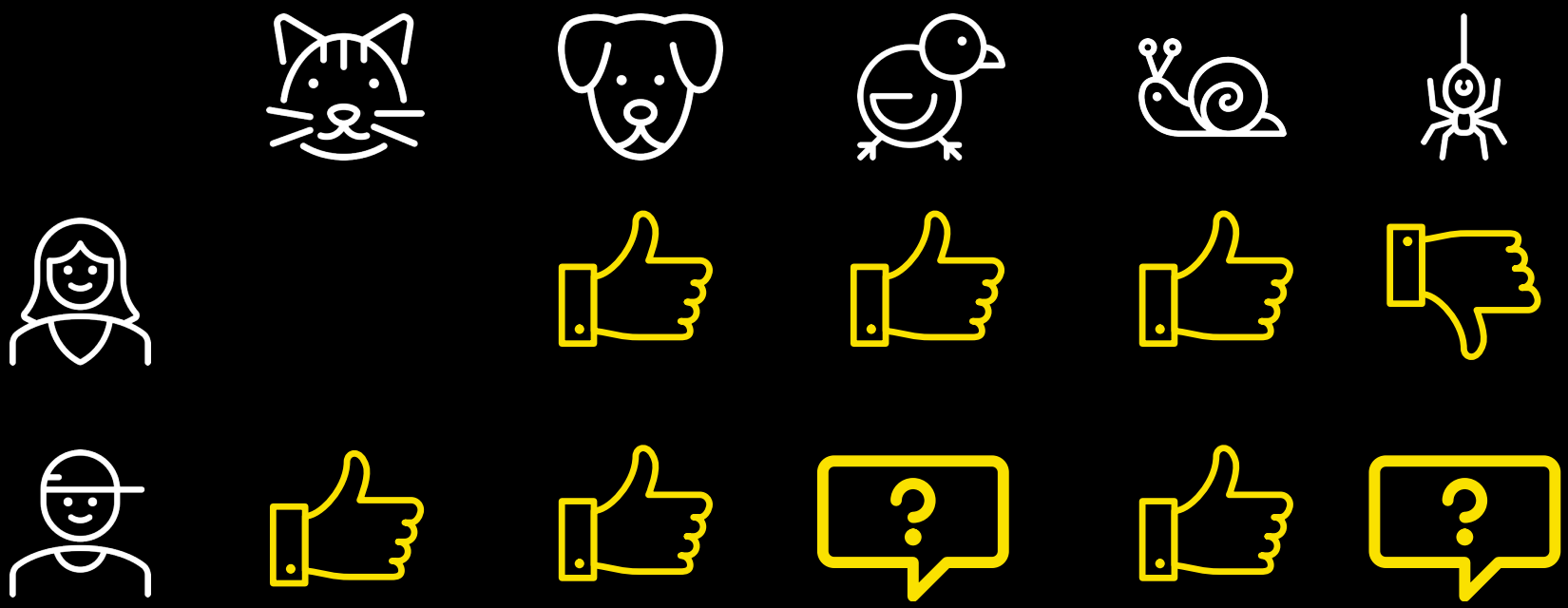
오토인코더를 사용한 풀이

자잘한 팁

Q&A

kakao

문제 소개: Collaborative Filtering



B는 병아리를 좋아할까? 거미를 좋아할까?

A와 B는 취향이 비슷하다면, B는 아마 병아리를 좋아할 거고, 거미는 싫어할 거야.

유저가 선택, 혹은 소비한 아이템으로 유저의 취향을 추측할 수 있다는 가정

문제 소개

- 추천 시스템에서 기본적으로, 그리고 여전히 가장 많이 사용되는 테크닉인 협업 필터링(Collaborative Filtering)을 사용하는 문제. (이외의 풀이 방법도 존재)

- 유저들이 들은 음악의 목록이 주어졌을 때, 유저가 듣지 않았던 음악 중 유저가 선호할 만한 100개의 아이템 추천

- Jukebox 문제는 유저가 직접적인 선호도(좋고 싫음, 혹은 평점)을 남긴 것이 아니고, 유저가 남긴 기록은 그 음악을 들은 횟수입니다.

- Explicit한 점수가 주어진 것이 아니라, implicit feedback Collaborative Filtering 문제

평가 기준

평가 기준

- 제약 조건 내에 컴파일 이후 추천을 생성할 수 있는지, 코드 구현의 효율성
- 추천 결과의 정확도(NDCG@100)
- 사용한 알고리즘과 그 알고리즘에 대한 이해도

오픈소스를 사용할 수 있고, 사용 알고리즘에 대한 제약 사항이 없습니다.

- 직접 모델을 구현하는 경우 추천 알고리즘을 정확하고 효율적으로 구현했는지가 주된 채점 요소가 됩니다.
- 오픈 소스를 사용한 경우, 모델 선택 기준과, 실험 설계 및 평가가 얼마나 논리적인지가 주된 채점 요소가 됩니다.

문제 소개

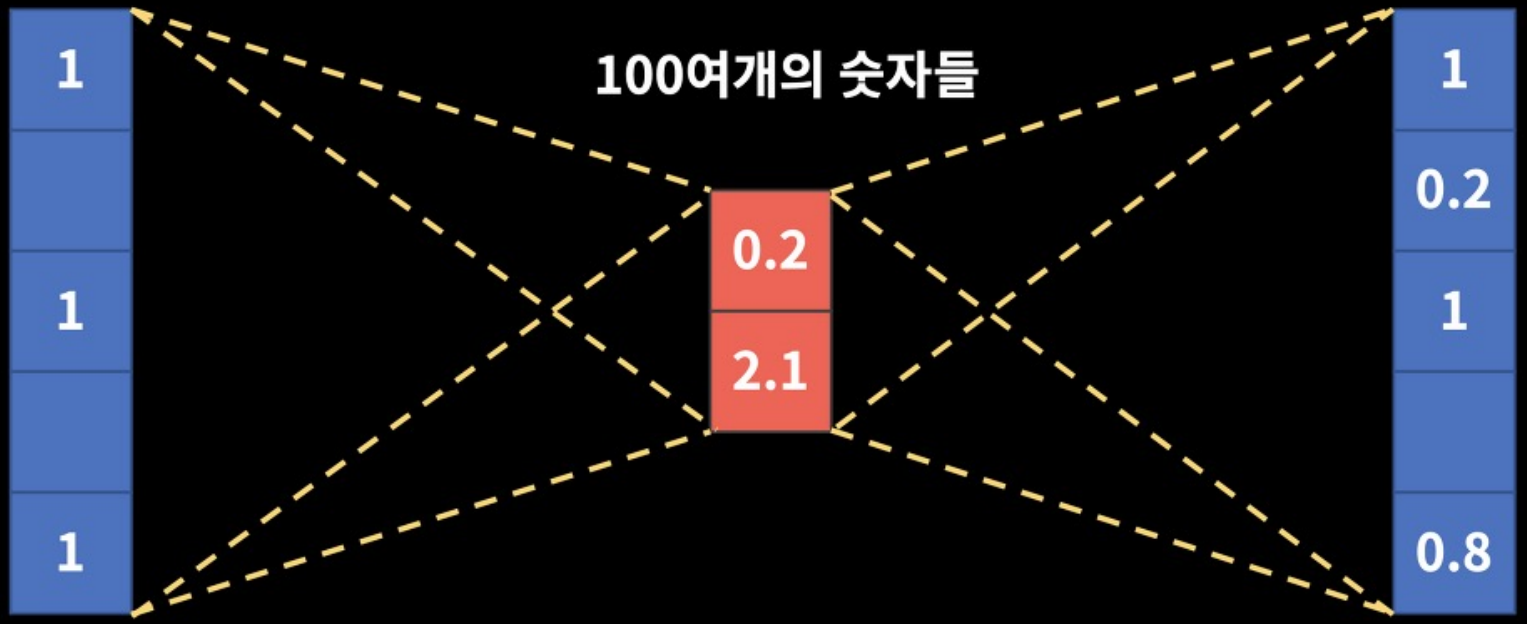
오토인코더를 사용한 풀이

자잘한 팁

Q&A

kakao

오토인코더를 이용한 유저 표현



유저가 들었던 음악 목록

이에 대한 복원

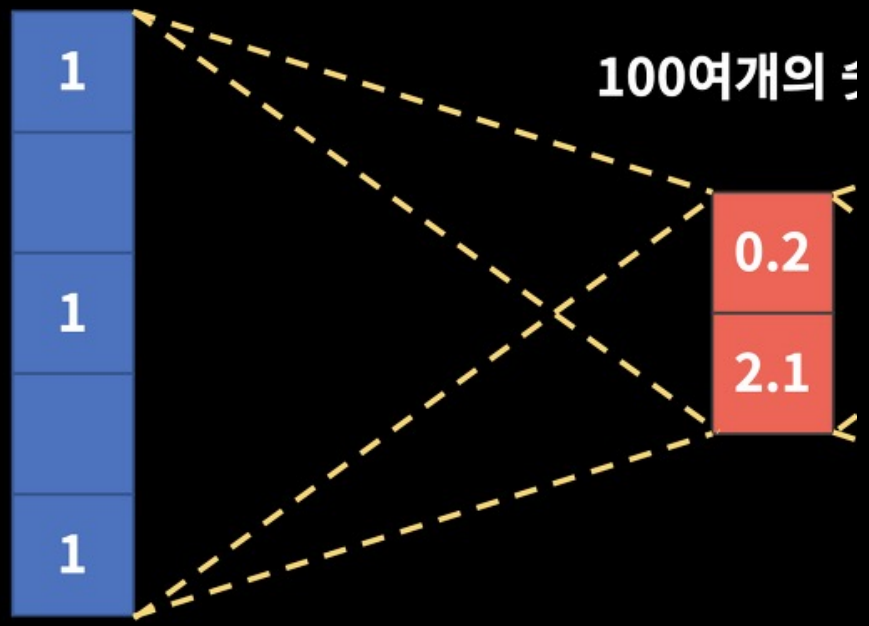
kakao

오토인코더를 이용한 유저 표현 만들기

유저 a가 본 음악의 목록을 x_a라 두었을 때 (x_a의 i번째 항은 i번째 음악을 들은 횟수), 유저의 임베딩은

$e_a = W_e x_a$로 표현할 수 있습니다.

W_e는 Encoding 을 할 때 사용하는 학습 가능한 행렬입니다.



유저 표현으로 추천 만들기

W_e, W_d를 잘 만들었다면, $y_a = W_d e_a$를 계산해, y_a의 세 번째 컬럼의 값, 다섯 번째 컬럼의 값을 비교해 어떤 아이템을 더 선호할 지 확인할 수 있습니다. 일반적인 협업 필터링 추천 모델은, 이렇게 스코어를 생성한 뒤 유저가 클릭하지 않았던 아이템 중 가장 점수가 높은 아이템들을 추천합니다.

모델 학습하기

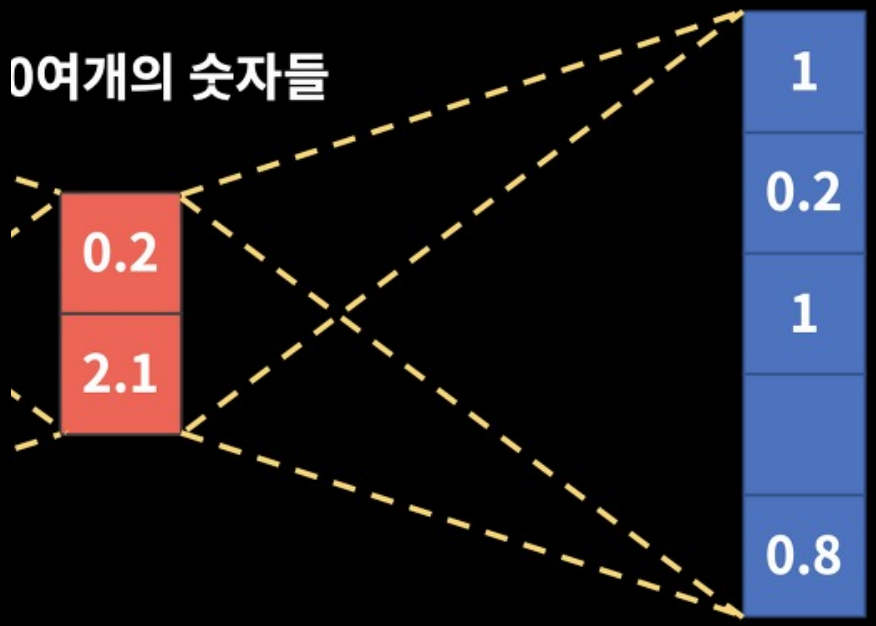
유저 표현이 잘 학습되었다면, 유저가 원래 들었던 음악을 $\mathrm{y_a} = W_d e_a$로 잘 복원할 수 있을 것입니다.

따라서 $\mathrm{x_a}$과 y_a 의 오차는 작을 것입니다.

그렇다면, 우리는 $\mathrm{x_a} - y_a$를 최소화시키도록 W_e, W_d를 데이터에 맞게 수정할 수 있을 것이고,

$$\mathrm{Loss} = (\mathrm{x_a} - y_a)^2$$

이는 다음 목적함수를 최소화하도록 W_e, W_d를 변경하는 것과 같습니다.



kakao

오토인코더를 조금 다르게 만들기

방금 만든 오토인코더는 여러 가지 방식으로 확장할 수 있습니다. 모델 학습 과정에서 노이즈를 추가해주는 방법이 일반적이지만, 여기서는 모델을 더 단순하게 만드는 방법을 사용해봅시다.

$$\mathrm{y_a} = W_d e_a = W_d(W_e x_a) = (W_d W_e)x_a$$

오토인코더의 인코딩/디코딩 과정은 행렬 두개로 표현되는데, 이 행렬들 두개로 둘 필요 없이, **하나의 행렬** $H = (W_d W_e)$**으로 표현할 수 있습니다.** 이렇게 하면 모델 하이퍼패러미터 중 하나였던 latent dimension을 없애버릴 수 있다는 장점이 있습니다. 이럴 경우 업데이트할 목적 함수는

$$\mathrm{Loss} = (\mathrm{x_a} - Hx_a)^2$$

가 됩니다. 단, 예외 처리해주어야 할 경우가 생기는데, $(\mathrm{x_a} - Hx_a)^2 = x_a^T(I-H)^T(I-H)x_a$으로 두었을 때, $I - E$가 되는 경우, 모델은 유의미한 추천을 생성해주지 못합니다.

구현

목적 함수를 개별 유저 a가 아니라, 모든 유저에게 더해 준다면 다음과 같은 값을 얻을 수 있습니다.

$$\sum_u (x_u - Ex_u)^2 = [x_1 \dots | x_n]^T (I - H)^T (I - H) [x_1 | \dots | x_n]$$

Let $X = [x_1 | \dots | x_n]$

$$X^T (I - H)^T (I - H) X = (X - HX)^T (X - HX) = |X - HX|_2^F$$

Constrained by $diag(E) = 0$

위 최적화 문제는 여러 방법으로 해결할 수 있습니다. 여기서는 Batched Gradient Descent를 사용합시다.

kakao

모델 정의

```
X = jnp.array(X.todense()) #n_user * n_items의 행렬
P = jnp.array(np.random.normal(0, 0.001, size=(8190, 8190)))
P.at[jnp.diag_indices(Y.shape[0])].set(0) # constraint

def loss(X, P):
    Y = X @ P
    ret = (((X - Y) **2).sum() / X.shape[0])
    ret = ret + 0.01 * (P * P).sum() # regularization
    return ret
```

학습

```
lr = 0.05
for i in range(100):
    batch = np.random.choice(X.shape[0], 512)
    l = loss(jX[batch], P)
    print(l)
    r = grad(loss, argnums=1)(jX[batch], P)
    P -= lr * r
    P.at[jnp.diag_indices(P.shape[0])].set(0) # constraint
    lr *= 0.99 # weight decay
```

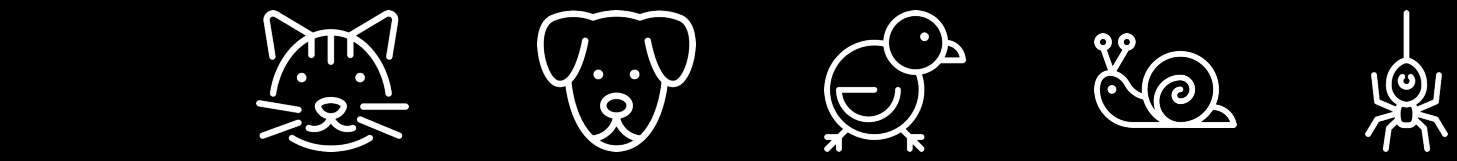
문제 소개

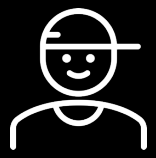
오토인코더를 사용한 풀이

자잘한 팁

Q&A

가장 간단한 모델 체크

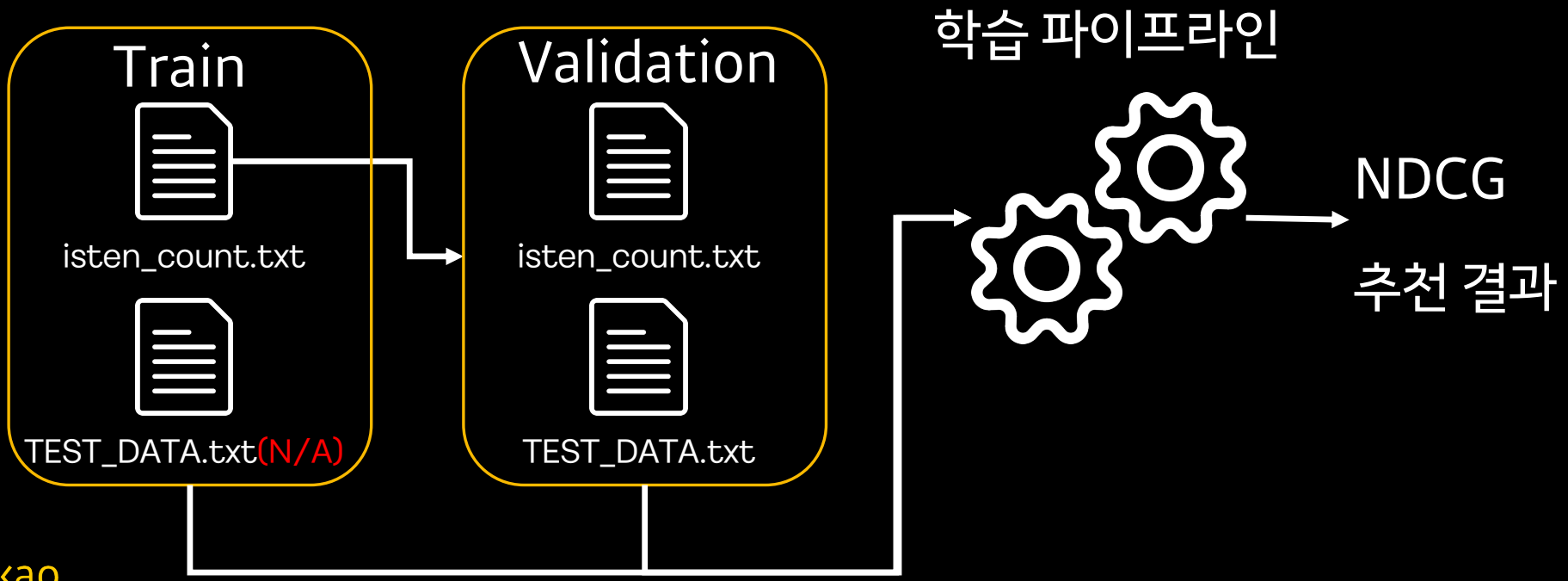


	0.8	1.0	0.9	1.0	0.1
	1.0	0.9	0.9	0.9	0.1

모델을 학습한 이후 추천을 생성했을 때, 유저가 이미 본 아이템에 대한 스코어가 낮으면, **무언가 잘못되어 있다고 확신해도 좋습니다.**

Validation과 evaluation 포맷을 동일하게 정의하기

주어진 데이터를 train/validation으로 분리할 때, 디버깅과 테스트를 용이하게 하기 위해서, validation 데이터의 입력은 listen_count.txt와 같은 형태로, validation test 데이터는 jukebox.pdf에 설명되어 있는 데이터와 같은 형태로 정리하는 것이 좋습니다.



유저 아이디, 아이템 아이디는 모델 외부에서 관리

특히 유저와 아이템의 아이디가 정수인 경우 이를 혼동하기 쉽습니다.

예를 들면, 아이템 아이디가 1부터 시작하는데, numpy array 인덱스는 0부터 시작합니다. 추천 결과를 아이템 아이디로 변환하는 과정에서 실수로 numpy 인덱스를 결과로 리턴하는 경우가 있을 수 있고, 가장 발생하기 쉬우며 치명적인 실수 중 하나입니다.

```
with open(user_ids_fname, 'r') as f:
    target_user_ids = f.read().strip().split()
tr = pd.read_csv(listen_count_fname, sep=' ', header=None, dtype=str)
tr.columns = ['uid', 'sid', 'cnt']
tr['cnt'] = tr['cnt']
uid2idx = {_id: i for (i, _id) in enumerate(tr.uid.unique())}
sid2idx = {_id: i for (i, _id) in enumerate(tr.sid.unique())}
idx2uid = {i: _id for (_id, i) in uid2idx.items()}
idx2sid = {i: _id for (_id, i) in sid2idx.items()}
tr['uidx'] = tr.uid.apply(lambda x: uid2idx[x])
tr['sidx'] = tr.sid.apply(lambda x: sid2idx[x])
```

유저/아이템 id를 스트링 등 다른 표현을 사용하고, 모델은 정수만을 처리하는 방법이 가장 단순하고 쉽습니다.

Q&A

kakao

감사합니다!

kakao