딥 러 닝
구 체 적 계 획 안

AI0

목 차

- Dataset 소개
- Task 소개
- 데이터 처리 계획
- Auto Encoder
- Predictor
- 요약

데이터셋 소개

데이터 특징:

- instacart는 미국의 이커머스 기업(우리나라의 경우 쿠팡 등)으로, 웹사이트와 모바일에서의 주문을 처리해 배송해주는 업체
- 해당 대회의 목적은 "Basket Analytics"로써, 주문 기록과 함께 주문된 제품들에 대한 정보를 통해 제품 간의 관계를 분석하는 기법을 의미함. → 요일, 시간대 정보 역시 "소비자의 동향 " 이라는 점에서 중요한 의미를 지님

활용 목적:

- 고객별 주문 시퀀스를 기반으로 다음 주문의 제품 목록을 예측하는 딥러닝 모델 설계

데이터 출처:

- Instacart Market Basket Analysis (링크: https://www.kaggle.com/datasets/psparks/instacart-market-basket-analysis/data?select=order_products__prior.csv)

데이터셋 디렉터리 단위 설명

1. **orders.csv**

- 가장 메인이 되는 '주문'에 대한 자료임.

2. **order_products_X.csv**

- 각 주문에서 무슨 제품을 주문했는지에 대한 자료임.

- X=train이면 개인별 주문에서 마지막 주문을 의미하며 X=prior일 시 마지막이 아닌 주문을 의미함.

→ X=test인 경우는 대회의 문제로써 직접 만들어야 함

3. **products.csv** + **departments.csv** + **aisles.csv**

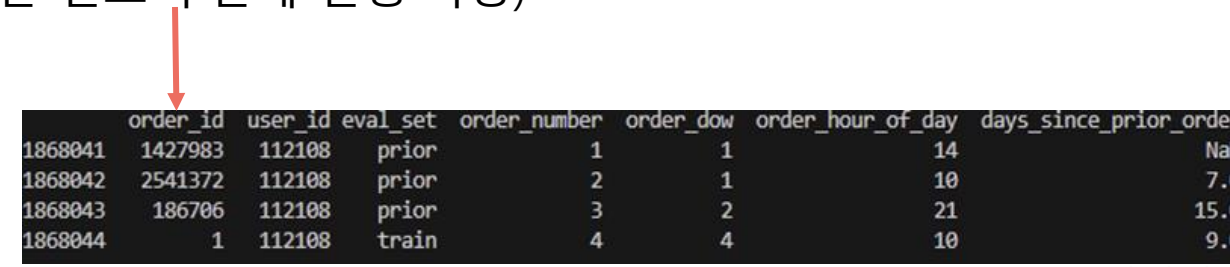
- 이들이 합쳐져서 '제품'의 특성을 수치화할 수 있음.

→ department는 제품의 대분류, aisle는 실제 제품의 진열 매대 의미

파일 단위 설명 - ORDERS.CSV

order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order
int	int	string	int	int	int	int

- **order_id**: 주문 번호(1~3421083, 주의: 1번 주문이 더 큰 번호 주문에 선행 가능)
- **user_id**: 고객 번호(1~206209)
- **eval_set**: 주문의 시퀀스적 의미(prior, train, test)



	order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_orde
1868041	1427983	112108	prior	1	1	14	Na
1868042	2541372	112108	prior	2	1	10	7.
1868043	186706	112108	prior	3	2	21	15.
1868044	1	112108	train	4	4	10	9.

→ (1) prior: 시퀀스의 마지막 주문이 아닌 주문

→ (2) train/test: 시퀀스의 마지막 주문 → test 주문 정보는 제공 안됨(**target**)

- **order_number**: 각 유저의 주문 시퀀스 순서(1~100)
- ~~**order_dow**~~: dow는 day of week의 약자로, 주문 요일(0: 일요일, 0~6)
- ~~**order_hour_of_day**~~: 주문 시각(0시부터 23시, 0~23)
- ~~**days_since_prior_order**~~: 직전 주문으로부터 경과한 날짜(Null 또는 0~30)

→ 최소는 0(당일 또 주문)이나 첫 주문의 경우 항목은 비워져 있음

파일 단위 설명 - ORDERS_PRODUCTS_X.CSV

order_id	product_id	add_to_cart_order	reordered
int	int	int	bool

- **order_id**: 주문 번호
- **product_id**: 제품 번호(1~49688)
- **add_to_cart_order**: 주문시 장바구니에 담은 순서(1~**145**)
- **reordered**: 재구매 여부(0~1)

파일 단위 설명

- PRODUCTS.CSV + DEPARTMENTS.CSV + AISLES.CSV

product_id	aisle_id	department_id	product_name	department	aisle
int	int	int	string	string	string

- **product_id**: 제품 번호(1~49688)
- **aisle_id**: 매대 번호(1~134) → 6번의 경우 'other'이기에 구분성이 떨어진다고 볼 수도 있으나, 그 외의 주의해야 할 항목이 없으므로 유지하는 것이 바람직할 듯 함.
- **department_id**: 제품군 번호(1~21) → 2번의 경우 'other', 21번의 경우 'missing'이므로 이들을 **동일 값으로 취급**하는 방안도 가능할 것으로 보임
- **product_name**, **department, aisle**: 문자열로 된 실제 항목명

TASK 소개

각 고객의 과거 주문 내역을 바탕으로 다음 주문(제품 목록)을 예측하는 다중 레벨 분류 문제

입력:고객의 과거 주문 데이터

출력:다음 주문에서의 각 제품 구매 여부

데 이 터 처 리 계 획

전처리 목표: 고객별 주문 이력을 시간 순서대로 정렬하고, 이를 모델 학습에 맞는 구조로 바꾸는 것

- 고객 ID(user_id) 기준으로 주문들을 주문 순서(order_number)대로 정렬
- 각 주문은 하나의 주문 번호(order_id)를 가지며, 여기에 포함된 제품 목록을 오토인코더를 통해

임베딩 벡터(Embedding Vector)로 요약함

데이터 활용

- 예측 모델에는 여러 개의 주문 EV를 입력하고, 그 다음 주문의 EV의 제품별 재구매율을 예측하도록 학습
- 사용자 별 구할 수 있는 마지막 주문(train / test 직전의 prior)을 target으로 두고 직전까지의 데이터를 이용

AUTO ENCODER

목적: 주문 하나를 잘 요약한 임베딩 벡터로 변환

1. **오토 인코더 입력 전**

- 주문마다 제품 번호 문장 만들기(ex.24852 13176 21137...)

2. **오토 인코더 입력**

- 입력: 주문 하나(제품 하나)의 TF-IDF 희소 벡터

3. **오토 인코더 구조**

- 주문을 TF-IDF 벡터로 변환(->order_tfidf)
- Decoder 마지막 층 Linear(2048->p, bias=False) ->제품 별 임베딩 필요시

  행 - latent -> 제품 가중치 벡터 -> 제품 임베딩

- EV_order 추출 - Encoder 출력 z전체를 그대로 저장
- EV_product 추출

선택사항: 제품 별 임베딩 추출

Decoder weight를 간접 제품 임베딩으로 사용 가능

4. 오토 인코더 출력

- EV_order와 EV_product를 합친 주문 하나의 정보가 담긴 EV_combo

AUTO ENCODER

1. **오토 인코더 사용 이유**

- 한 유저의 주문 내역에는 수많은 제품이 들어가 있고,

  제품 개수(product_id 49688개 존재)도 매우 많음.

- 제품별로 벡터(임베딩)를 바로 만들려면 계산량이 너무 많고 희소성도 커서 효율이 떨어짐.

2. **오토 인코더 구조 상세 설명**

- TF-IDF 행렬 : TF - 그 주문에서 제품 p가 차지하는 비율

  IDF - 모든 주문 중 제품 p가 얼마나 드물게 나타는지 비율

-> 특징적인 제품이 동시에 존재하는 주문들이 가깝다고 판단 가능(주문간 유사성)

- 구조: Encoder(EV로 변환) / Decoder(EV의미 해석)
- 목적: 의미 압축 → 복원

특징

- 단순한 one-hot보다 제품 의미가 잘 반영됨
- 제품 간 연관 관계 기반의 dense한 의미 벡터
- 일종의 룩업 테이블로 볼 수 있음
- 과적합 유도

성능 평가

- 훈련 시 발생한 loss들의 log를 통해 파악
- 주문 유사도 – cosine similarity
- 제품 유사도 – cosine similarity (제품 벡터를 구한다면)
- T-SNE/UMAP

PREDICTOR

목적: 고객의 과거 주문 벡터 시퀀스를 보고 다음 주문의 EV 벡터를 예측

1. CNN **입력 구성**

- 하나의 주문 = (145(주문 제품 수), 64(제품 벡터))EV_combo
- CNN을 통해 하나의 주문을 요약 벡터(예: 128차원)로 압축
- 여러 주문 시퀀스(ex.N=4)를 (B, N, 128) 형태로 GRU에 입력

2. CNN **구조**

- Conv1D (64 → 128): 제품 EV(64차원)를 필터링해 지역 패턴 추출
- Kernel size = 3: 연속된 3개 제품 간의 관계를 파악
- ReLU: 비선형 활성화 함수로 표현력 향상
- AdaptiveMaxPool1d: 제품 수(145개)에 관계없이 1개의 요약 벡터로 압축

3.CNN **출력**: 주문 1개당 128차원 EV_order

4. GRU **입력 구성**

- N개의 고객 과거 주문 시퀀스(EV_order)
- EV_order는 CNN으로 변환된 128차원 EV

5. GRU **구조**

- GRU(Gated Recurrent Unit) Layer: 시계열 데이터를 입력받아 고객의 과거 주문 패턴을 기억 및 요약
- Fully Connected(Dense) Layer : GRU의 최종 시퀀스를 최종 예측 차원(제품 수)으로 변환

6.GRU 출력 : 다음 주문에서 주문할 제품들의 목록을 담은 EV_order_pred

PREDICTOR

CNN 사용 이유

- 오토 인코더로 변환된 하나의 주문 = 64차원
  EV_product x product_id(최대 145개)→ (64, 145)
- 하지만 64 * 145 * N (주문 수)를 GRU 입력에 넣기에는 많음
- CNN은 이미지(예: 256×256×3)와 같은 대용량 다차원 배열을 효과적으로 처리하는 구조적 강점
- 실제로 제품 임베딩 및 주문 정보를 결합한 텐서 형태의 데이터는 CNN의 컨볼루션 레이어를 통해 중요한 feature를 추출함

GRU 사용 이유

- Task가 시계열과 큰 관련이 있어서 RNN 중 GRU와 LSTM을 고려함
- LSTM보다 시계열 처리 능력은 낮지만 계산을 단순화한 구조
- 각 주문을 이미 임베딩 벡터로 변환한 후이기 때문에 시퀀스 길이가 짧을 땐 LSTM의 장기 기억력이 도움이 되지 않음
- LSTM은 과적합되기 쉽고 튜닝이 힘듬

-> GRU 학습 후 성능이 부족하다고 느끼면 LSTM 사용

PREDICTOR 성능 평가

Baseline ML : XGBoost

선정 이유

- XGBoost는 Gradient Boosting 알고리즘의 효율적인 구현으로, 다양한 예측 문제에서 우수한 성능
- 재구매 예측과 같은 이진 분류 문제에 적합함

입력 (Features):

- 고객의 과거 구매 이력
- 제품 카테고리 정보
- 구매 간격 등 시계열 특성
- 고객 및 제품의 메타데이터

출력 (Target):

- 다음 주문에서 주문할 제품들

기대 역할

- 딥러닝 모델(CNN, GRU 등)과의 성능 비교를 위한 기준선 제공
- 모델의 성능 향상 여부를 판단할 수 있는 참조점 역할

정답(Label) : eval_set이 train/test직전의 prior인 마지막 주문 데이터에 담긴 제품들

평가지표

- F1-score(macro/micro): 다중 레이블 분류의 대표적 지표
- Precision@K, Recall@K: 추천 품목 중 실제 정답 적중률
- AUC-ROC: 예측 확률의 구분력
- Log Loss: 확률 예측의 신뢰도

비교방식

- XGBoost와 평가지표들을 수치화해서 성능 우위 평가

요 약

- Instacart 주문 데이터를 기반으로 한 데이터셋 전처리
- 오토인코더에 주문(제품 정보) 입력

-> 그 주문이 담긴 145(주문 당 최대 제품 수) x 64차원 EV_product(제품 하나의 정보) 출력

- CNN에 145 * EV_product 입력 -> 주문 한 번의 정보를 담은 EV_order 출력
- RGU에 EV_order * N(prior 주문 개수) 입력

-> 다음 주문을 예측한 EV_order_pred 출력

- EV_order_pred를 오토인코더로 변환한 정답(train 주문)과 비교