3. 모델 설계 및 구현

- 하이퍼파라미터 튜닝

학습에서 조정할 수 있는 하이퍼파라미터로는 크게 3가지(1. 모델 관련 2. 학습 관련 3. 손실함수 관련)로 볼 수 있다.

모델 관련 하이퍼파라미터에는 CNN out size, CNN Kernel size, CNN Pool size, GRU hidden dimension, GRU Layers, Dropout, FC hidden dimension이 있다. CNN은 우리 모델에서 GRU의 입력을 하기 위해 (145(최대 제품 수), 64(제품 임베딩 벡터))의 차원을 축소하는 역할을 하는데 CNN out size는 해당 차원(145, 64)을 CNN에 넣고 차원 축소를 진행한 후 출력되는 차원이다. 낮으면 표현력이 충분하지 않고, 높으면 복잡하고 계산 량이 늘어난다. CNN Kernel size는 1개의 주문내에서 제품간 관계를 몇 개씩 묶어서 볼 것인지 결정한다. 낮으면 문맥이 부족하고, 높으면 지역성이 흐려진다. CNN Pool size는 CNN 출력 시퀀스를 얼마만큼 묶어서 요약할지 결정하는 크기로서 낮으면 요약을 시키지 못해 연산 량이 증가하지만 높으면 정보가 손실될 수 있다. GRU hidden diemension은 GRU가 CNN에서 요약된 각각의 주문을 시계열로 받아서 처리할 때 각 시점의 은닉 상태 크기를 의미한다. 낮으면 빠르지만 표현력이 낮아지고, 높으면 과적합 위험, 연산 량이 많아진다. GRU Layers는 1일때는 CNN 출력 주문 시퀀스를 처리한 뒤 FC로 바로 넘기지만, 2 이상일때는 이전 레이어의 은닉 상태 시퀀스를 다시 처리한다. 즉 시계열 패턴을 인식시키는데 도움을 준다. Dropout은 과적합 위험을 낮추는 하이퍼파라미터로서 CNN에는 사용되지 않고, GRU layers가 2 이상일때, FC에서 사용된다. FC hidden dimension은 최종 예측 전 FC의 중간 차원 수로서 모델이 GRU의 출력을 받아서 한 번 더 비선형 변환할 때 출력되는 차원이다. 작으면 표현력이 부족하고 높으면 계산량과 과적합이 높아진다.

학습 관련 하이퍼파라미터에는 Learning Rate, Batch Size, Epochs, Early Stopping Patience가 있다. 학습률, 배치사이즈, 에폭 수는 생략하겠다. Early Stopping Patience는 성능이 개선되지 않을 때 학습을 조기 종료하는 매커니즘이다. 단위는 epoch이고 일정 epoch 동안 validation 성능이 더 좋아지지 않으면 멈추게 한다. 시간 절약과 과적합 방지에 도움이 된다.

손실함수는 BCEWithLogitsLoss, MarginRankingLoss, BPRLoss를 활용하였다. 하이퍼파라미터로는 Pos Weight, Ranking Margin, Ranking Weight, Ranking Sampling Ratio, Threshold가 있다. Pos Weight는 BCE에 활용되는 인자인데 모델 출력이 49688(전체 제품 수)개의 Logits이기 때문에 정답의 비율(최대 145)이 매우 적은 불균형 데이터이다. 따라서 모델이 정답을 맞출 때 가중치를 주는 역할을 한다. 낮으면 대부분을 부정으로 예측하고 높으면 오답도 긍정으로 예측하므로 Precision이 낮아진다. Ranking Margin은 Margin에 쓰이는 인자로 정답과 오답 간 logits 차이를 얼마나 크게 벌려야 하는지 정하는 값이다. 작으면 구분 효과가 약하고 높으면 학습이 불안정해진다. Ranking Weight는 BCE와 Margin/BPR을 혼합할 때의 비율이다. 계산 수식은 total\_loss = bce\_loss + ranking\_weight \* ranking\_loss(BPR/Margin)이고 1이면 ranking과 BCE과 동등하게 반영되고, 1 이상이면 ranking이 주도하게 된다. Ranking Sampling Ratio는 BPR, Margin에서 정답과 오답 비교를 위해 몇 개 오답을 샘플링할지 비율이다. 출력할 때 전체 제품(49688개)에서 오답이 대부분이기 때문에 일부만 비교하기 위해 사용한다. 낮으면 빠르지만 성능이 낮고 높으면 느리지만 성능이 좋아진다.

-모델 훈련 및 성능 평가

본 모델은 제품 재구매 예측을 위해 CNN + GRU 기반 시퀀스 모델을 사용하며, 전체 학습 파이프라인은 크게 데이터 로딩 → 모델 정의 → 학습 → 검증 → 테스트 순으로 구성된다. 아래에 각 단계를 상세히 기술한다.

1. 데이터 구성 및 분할  
   OrderSequenceDataset 클래스를 통해 데이터를 구성하며, 한 유저의 과거 5개의 주문(products\_x)과 그 이후 주문의 정답 제품 목록(target\_y)을 활용한다. 각 제품은 사전 학습된 임베딩 벡터(64차원)를 통해 표현된다. 전체 샘플 수를 기준으로 학습, 검증, 테스트 데이터를 6:2:2 비율로 분할하며, custom\_collate\_fn을 통해 배치 처리 시 텐서로 묶는다.
2. 모델 정의  
   config 설정에 따라 CNNPredictorModel 또는 AttentionPredictorModel 중 하나를 선택하여 사용한다. 기본 구조에서는 CNN이 한 주문 내 제품 간 관계를 요약하고, GRU가 주문 시퀀스를 처리한 뒤, FC 계층을 통해 전체 제품에 대한 재구매 확률을 예측한다. 출력 차원은 전체 제품 수에 해당한다.
3. 손실 함수 구성  
   손실 함수는 BCE, BCE + BPR, BCE + Margin으로 설정되며, compute\_loss 함수 내에서 자동으로 분기 처리된다. 다중 라벨 예측 문제 특성상 정답(1)의 비율이 극히 적으므로, BCE 손실 함수에는 pos\_weight 인자를 적용하여 불균형을 보정한다.
4. 학습 과정  
   에폭 단위로 학습을 반복하며, 각 배치마다 입력 시퀀스를 모델에 전달하여 logit을 예측하고 손실을 계산한 뒤 역전파 및 옵티마이저를 통한 파라미터 갱신이 이루어진다.
5. 검증 및 Early Stopping  
   각 에폭 종료 후 validation set을 활용하여 성능을 평가한다. 사용되는 평가지표는 F1-micro, Precision@10, Recall@10이며, validation 기준 F1-micro가 개선될 경우 해당 모델 가중치를 저장한다. config.early\_stopping\_patience 설정에 따라, 성능 향상이 일정 횟수 이상 발생하지 않으면 학습을 조기 종료한다.
6. 테스트 평가 및 시각화  
   학습이 완료된 후 가장 성능이 좋았던 시점의 모델을 불러와 test set에 대해 최종 평가를 수행한다. 평가 결과는 F1-micro, Precision@K, Recall@K로 기록되며, 학습 로그는 그래프로 시각화되어 .png 파일로 저장된다.

총괄적으로 전체 학습 파이프라인은 train\_model() 함수 하나로 통합되어 있으며, 다양한 손실 함수 실험과 하이퍼파라미터 조정이 config.py에서 파라미터만 수정함으로써 유연하게 실험할 수 있도록 설계되었다.

-성능 비교 및 모델 개선 노력

본 연구에서는 제안한 CNN-GRU 기반 딥러닝 모델을 다양한 구조 및 손실 함수 조합으로 실험하였으며, 이를 XGBoost 기반 베이스라인 모델과 비교하였다. 모든 실험은 동일한 데이터셋과 평가 지표(F1-micro, Precision@10, Recall@10)를 기준으로 수행되었다.

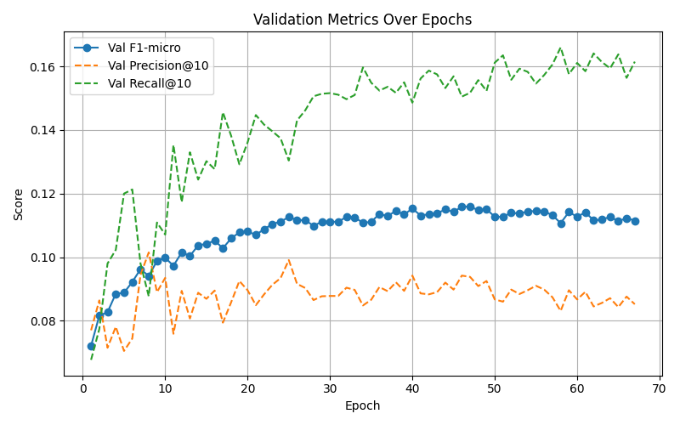
먼저 XGBoost는 입력 데이터를 (304,000명 × 128차원)으로 요약한 뒤, 300개의 제품 후보군에 대해 One-vs-Rest 구조로 이진 분류기를 학습하였다. 클래스 불균형에 대응하기 위해 클래스별 scale\_pos\_weight를 적용하였으며, 평가 결과 F1-micro는 약 0.0119, Precision과 Recall은 각각 0.01 수준으로 딥러닝 모델 대비 매우 낮은 성능을 보였다.

CNN-GRU 기반 딥러닝 모델에서는 먼저 최적 Pos Weight와 Threshold를 찾기 위해 Grid Search를 진행하였다.

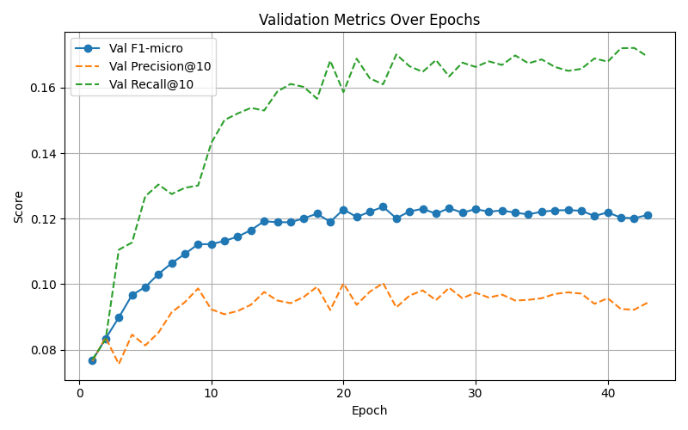
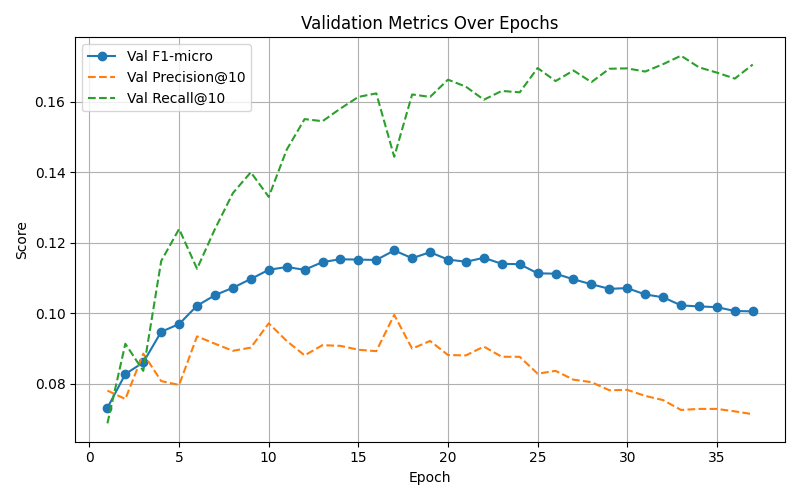
텍스트, 메뉴, 스크린샷, 모노크롬이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

이것을 기반으로 Pos Weight=10, Threshold=0.4로 고정한 채 모델 하이퍼파라미터를 조절하여 CNN\_1, CNN\_2, CNN\_3, CNN\_4, CNN\_5 등의 실험을 진행하였다.



CNN\_1 CNN\_2



CNN\_3 CNN\_4

텍스트, 라인, 그래프, 도표이(가) 표시된 사진

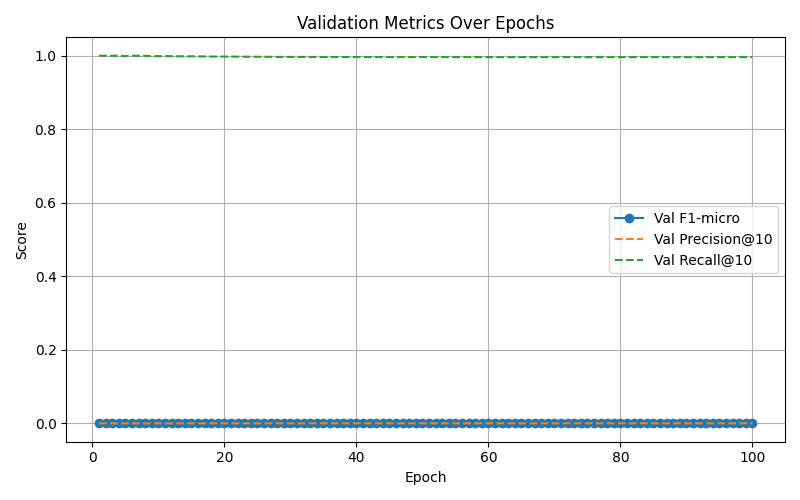
AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

CNN\_5

CNN2는 F1-micro 0.1229, Precision@10 0.1067, Recall@10 0.1601로 가장 우수한 성능을 기록하였다. 이는 dropout 적용 및 학습률 조절을 통해 모델이 더 안정적으로 학습되었기 때문이다.

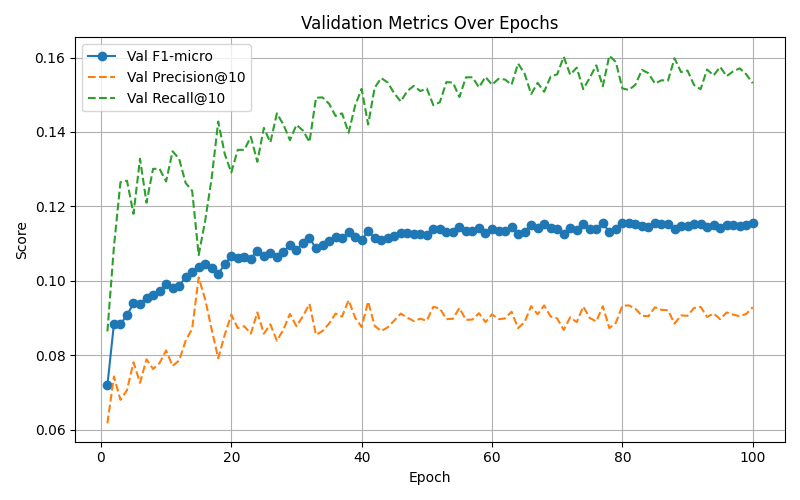
손실 함수에 대해서는 기본 손실 함수인 BCEWithLogitsLoss 외에도 BPRLoss, MarginRankingLoss를 적용하거나 BCE와 혼합하여 성능 비교를 수행하였다. BPR 단독, Margin 단독 사용 시에는 학습이 불안정하거나 F1이 0에 가까워 실질적인 학습이 이루어지지 않았으며, 혼합 손실에서는 BCE + Margin 구조가 정밀도가 일부 향상되었으나 추천 시스템 관점에서 중요한 Recall이 감소하는 문제가 있었다. 그리고 BCE + BPR 구조는 단순히 성능이 낮았다.

텍스트, 스크린샷, 라인, 그래프이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

BPR 단독 사용 Margin 단독 사용

텍스트, 라인, 그래프, 도표이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다. BCE + Margin BCE + BPR

결과적으로 BCE 손실 단독 구조가 가장 안정적인 학습과 균형 잡힌 성능을 보였다.

CNN 대신 Self-Attention 구조를 사용하여 주문 내 제품 간 관계를 모델링하는 시도도 이루어졌으나, 학습 안정성과 성능 모두에서 CNN 기반 구조에 비해 떨어졌다. Attention 기반 모델은 F1-micro 0.1013, Precision@10 0.1245, Recall@10 0.0854로 성능이 낮았으며, 일부 실험에서는 손실값이 NaN으로 발산하여 학습이 정상적으로 진행되지 않았다.

요약하자면 XGBoost는 baseline으로서 의미는 있으나 딥러닝 모델에 비해 성능이 크게 낮았고, CNN-GRU 구조 중에서는 CNN\_5(CNN out size=256, CNN Kernel size=4, CNN Pool size=8, GRU hidden dimension=256, GRU Layers=1, Dropout=0.2, FC hidden dimension=256, Learning Rate=0.001, Batch Size=32, Epochs=100, Early Stopping Patience=20, Pos Weight=15, Threshold=0.4)가 가장 높은 성능과 안정성을 달성하였다. 손실 함수 중에서는 BCE 단독 사용이 가장 적절하며, Self-Attention 구조는 실험 결과상 불안정하므로 최종 구조에서는 제외되었다.