# FINAL Project

2019028313 / Kim Hyewon

Recommender System

User와 Item, rating정보를 통해 user에게 item을 추천해주는 system으로 Matrix Factorization, Embedding Layer, Regularization, Optimizer의 변화를 통해 model과 training방법을 개선하였습니다.

## dataset 분리

학습을 위해 주어진 train dataset을 분리하여 train, validation dataset을 구분하였습니다.

data = RecommendationDataset(f"{args.dataset}/ratings.csv", train=True)

# Data random split

train, val = torch.utils.data.random\_split(data, [data.len - size\_valdata, size\_valdata])

train\_loader = DataLoader(train, batch\_size=args.batch\_size, shuffle=True)

val\_loader = DataLoader(val, batch\_size=args.batch\_size, shuffle=False)

## origin model – Matrix Factorization

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| num | Batch size | Learning rate | Epoch# | Result |
| 1 | 16 | 0.01 | 15 |  |
| Loss: MSELoss  Optimizer: Adam  Minimum train cost : 0.739741  Minimum val cost: 1.748469 | | |
| 2 | 16 | 0.001 | 30 |  |
| Minimum train cost: 0.501020  Minimum val cost: 1.975342  learning rate감소  : cost값이 더 부드럽게 변화한다.  epoch# 증가  : lr의 감소로 조금씩 학습되지만 반복되는 epoch수를 증가시켜 충분히 학습될 수 있도록 하였습니다. | | |
| 3 | 16 | 0.001 | 20 |  |
| Minimum train cost: 0.699779  Minimum val cost: 1.746756  학습시간: 27분 | | |
| 4 | 32 | 0.001 | 20 |  |
| Minimum train cost: 0.826544  Minimum val cost: 2.279052  학습시간: 13분  batch size 증가  : 학습속도가 증가하였다. | | |
| 5 | 10 | 0.001 | 20 |  |
| Minimum train cost: 0.683769  Minimum val cost: 1.938070 | | |
| 6 | 32 | 0.001 | 50 |  |
| Minimum train cost: 0.395014  Minimum val cost: 2.021876  epoch이 반복될수록 train과 validation모두 cost가 감소하지만 두 cost값의 차이가 증가하였습니다. overfitting 발생 가능성이 있으므로 둘의 차이를 줄여야 한다. (regularization추가) | | |
| 7 | 32 | 0.001 | 20 |  |
| Minimum train cost: 13.070103  Minimum val cost: 13.012007  Regularization(weight\_decay=0.01):  Train과 Validation사의의 차이가 감소하였습니다. 그러나, min cost가 13으로 충분히 학습되지 않은 것 확인할 수 있습니다. | | |
| 8 | 32 | 0.001 | 20 |  |
| Minimum train cost: 2.341185  Minimum val cost: 2.667338  Regularization(weight\_decay=0.001):  Weight decay를 감소시켜regularization에 의한 underfitting의 발생을 억제하였습니다. 앞선 결과들보다 train cost는 증가하였지만 train과 validation의 성능이 유사해진 것을 확인할 수 있습니다. | | |
| 9 | 32 | 0.001 | 50 |  |
| Minimum train cost: 2.308742  Minimum val cost: 2.640971  Regularization(weight\_decay=0.001)  앞선 모든 확인에서 가장 성능이 좋았던 설정 8번을 epoch의 수를 증가시켜 성능을 확인하였습니다. | | |

hyperparameter를 통한 성능개선 후에는 Embedding Layer를 통해 model을 개선하였습니다.

## Embedding layer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| num | Batch size | Learning rate | Epoch# | Result |
| 1 | 32 | 0.001 | 20 |  |
| Loss: MSELoss  Optimizer : Adam  Minimum train cost: 2.481183  Minimum val cost: 2.710381  Origin model에서 가장 성능이 좋았던 hyperparameter로 embedding layer model의 성능을 확인하였습니다. | | |
| 2 | 32 | 0.001 | 20 |  |
| Minimum train cost: 0.658957  Minimum val cost: 1.1764094  Regularization(weight\_decay=0):  Weight decay를 0으로 설정하여 regularization이 적용되지 않은 학습의 결과를 확인하였습니다.  Validation과 train의 차이가 조금 증가하였으나 validation cost가 가장 작은 값까지 떨어지는 것을 확인할 수 있습니다. | | |
| 3 | 32 | 0.001 | 50 |  |
| Minimum train cost: 0.312719  Minimum val cost: 1.174236  Regularization(weight\_decay=0)  Minimum val은 떨어졌으나 시간이 지날수록 overfitting이 발생하여 train과 validation사이의 차이가 증가하고 validation cost가 증가하는 것을 확인할 수 있습니다. | | |
| 4 | 32 | 0.001 | 20 |  |
| Minimum train cost: 0.661709  Minimum val cost: 1.190939  Optimizer: AdamW  Regularization(weight\_decay=0.001)  weight decay를 사용하는 경우 Adam이 가장 좋은 Optimizer라 할 수 없기 때문에 optimizer를 AdamW로 변경하여 성능을 확인하였습니다.  (SGD, Adagrad의 사용결과는 성능이 좋지 않아 생략하였습니다.) | | |
| 5 | 32 | 0.001 | 20 |  |
| Minimum train cost: 2.419830  Minimum val cost: 2.634025  Optimizer: RMSProp  Regularization(weight\_decay=0.001) | | |

(final model 설명을 다음 페이지에 있습니다.)

## Final Model

|  |  |
| --- | --- |
| Result | Hyperparameter & Setting |
|  | Model: Embedding Layer  Epoch # : 50  Learning rate: 0.001  Batch\_size: 32  Optimizer: Adam  Loss: MSELoss  Weight decay: 0  early stopping:  validation data에서의 cost가 증가하는 양상을 보이면 학습을 종료합니다.  Minimum train cost: 0.692688  Minimum val cost: 1.128489 |

Model: Embedding Layer

* origin model과 Embedding Layer중 더 좋은 성능을 보여준(더 낮은 validation cost를 갖는) model을 사용하였습니다.

Optimizer: Adam

* SGD, Adagrad, AdamW, RMSProp을 확인해본 결과 Adam이 가장 낮은 validation cost를 얻을 수 있었습니다.

Weight decay: 0

* Embedding layer에서 weight decay를 설정하여 학습을 진행한 결과 0.1~ 0.0001의 낮은 weight decay를 설정하여도 성능의 개선이 없어 weight decay = 0 (default)로 설정하였습니다.  
  (validation cost의 감소가 줄고 train cost와의 차이에 변화가 적음)

early stopping

* weight decay가 0으로 설정되어 regularization을 하지 않으므로 early stopping을 통헤 overfitting을 해결하였습니다.
* validation cost값을 저장하여 이전 cost보다 증가하는 경우 학습을 종료한다. (위의 result를 확인하면 epoch의 초기 설정이 50이지만 early stopping에 의해 일찍 종료되었음을 확인할 수 있습니다.)