

Long-Term Project

|  |  |
| --- | --- |
| 과목명. | 데이터사이언스 |
| 담 당. | 김 상 욱 교수님 |
| 제출일. | 2021년 06월 11일 |
| 공과대학 | 컴퓨터소프트웨어학부 |
| 학 번 | 2016025969 |
| 이 름. | 정지훈 |

****

**목차.**

**1. Summary of algorithm**

**2. Detailed description of codes**

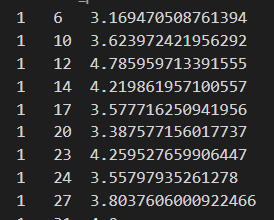
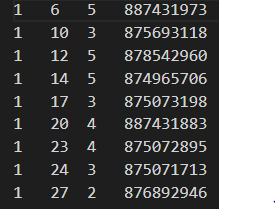
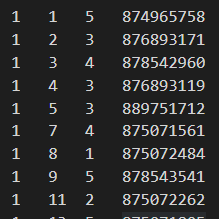
**3. Instructions for compiling source codes at TA's computer**

**4. Any other specification of implementation and testing**

**5. References**

**1. Summary of algorithm**

**- Overview**

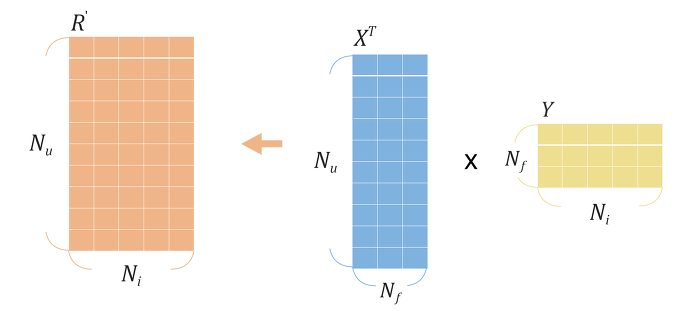


u1.test

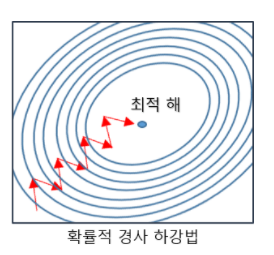
u1.prediction

u1.base

U1.base를 통해 주어진 user와 movie 그리고 평점에 대해서 Matrix Factorization 기법을 사용해 user와 movie의 Latent Factor Matrix를 얻고 두 Matrix 통해 평점을 예측합니다. 얻은 예측 값을 Neural Network Layer을 통과해 최종 값을 얻습니다. 최종 예측 값과 정답 값을 이용해 Loss를 구하고 이를 각 Layer의 weight과 latent vector를 학습합니다. 얻어진 Model은 이제 새로운 user와 movie에 대한 평점을 얻을 수 있습니다.

**- Matrix Factorization**

Matrix Factorization은 위의 그림과 같이 User와 Movie의 각각 개수를 row로 갖고 사전에 정의한 Factor의 숫자의 크기를 col으로 만드는 latent Factor Matrix를 User, Movie에 대해 각각 만듭니다. 이 두 Matrix에서 추론을 원하는 user-id, movie-id가 들어오면 해당하는 row를 각각 가져와 두 row의 Inner-product 값을 얻습니다. 하지만 User, Movie 별로 평균 등 특성이 다를 것입니다. 이 특성을 고려하기 위해 user-bias와 movie-bias에 대한 Matrix도 만들어 그 값을 예측 값에 더합니다. 이를 통해 얻은 값과 실제 평점에 대한 오차를 Loss로 설정해 Factor Matrix를 update 합니다.

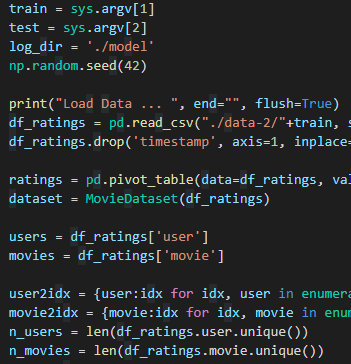
**- SGD**

SGD(Stochastic Gradient Descent)는 기존에 있던 Gradient Descent에서 모든 Data에 대한 Loss를 이용하는 것이 아닌 확률적으로 선택된 data의 Loss만을 이용합니다. 이를 통해 더 빠른 학습이 가능하며 기존 방법에 있었던 Local Minimum에 빠지는 단점을 해결합니다.

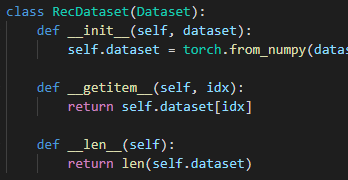
**- Result Test**

|  |  |
| --- | --- |
| Data | RMSE |
| U1 | 0.9419 |
| U2 | 0.9298 |
| U3 | 0.9279 |
| U4 | 0.9251 |
| U5 | 0.9309 |

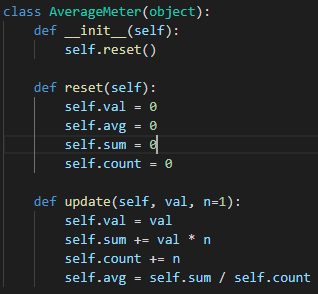
**2. Detailed description of codes**

**- Data Read**

compile에서 base 파일과 그에 대응하는 test 파일의 이름을 인자로 받습니다. 이후 base file에서 학습에 필요하지 않은 timestamp를 제거하고 user와 movie를 각각 row, column으로 해 rating을 value로 갖는 ratings pivot table을 만듭니다. 이후 user와 movie 번호가 중간에 비어 있을 수 있으므로 각각 비어 있지 않은 값으로 matching 하는 Dictionary를 만듭니다.

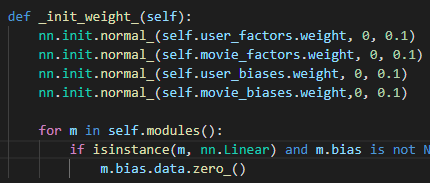
**- RecDataset**

추후 Model을 학습할 때 Dataloader에서 data를 얻어 올 때 dataset class를 통해 얻습니다. Dataset에 u1.base 값이 담겨 있는 상태에서 index가 주어지면 해당 값을 얻게 됩니다.

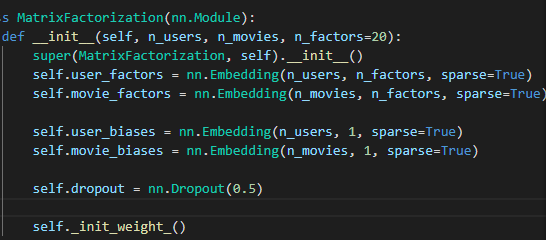
**- AverageMeter**

Model에서 얻은 예측 평점과 실제 점수인 Ground Truth를 가지고 MSE Loss를 구합니다. 이후 batch마다의 loss를 기록해 한 Epoch 안에서의 Loss 평균을 구하게 됩니다. 이 값은 추후에 Model이 학습되는지 그리고 Valid에서 Best Model 저장의 근거가 됩니다.

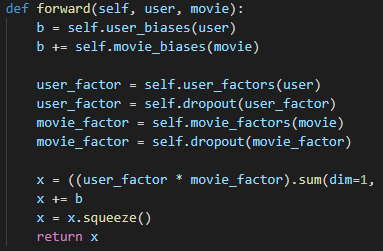
**- Init-weight**

만들어지는 latent Factor matrix와 user, movie 각 bias에 대해 값을 초기화 시킵니다.

**- Model Init**



Matrix Factorization의 user, movie 각각 latent factor matrix와 bias matrix를 선언하는 역할을 합니다. User, movie 각각 latent factor matrix에서 index에 해당하는 row를 얻을 때 dropout을 적용하기 위해 선언했습니다.

**- Forward**

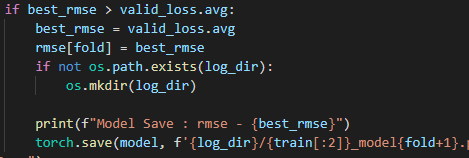
Model에 구하고자 하는 평점에 대한 user, movie index가 들어오면 각 matrix에서 factor를 얻어와 Inner-product를 통해 값을 얻습니다. 결과 값에 사전에 정의한 User, Movie bias를 더해 최종 값으로 추론합니다. Train에서 학습할 때 loss는 미분의 chain rule에 따라 backpropagation을 통해 각 latent factor vector와 bias vector를 학습합니다. 기존에 학습을 진행할 때 valid loss는 고정이고 train loss만 떨어지는 overfitting 현상이 일어나 dropout을 진행 feature를 불러올 때 값을 덜 가져오도록 설정했습니다.

**- Train Hyper Parameter**

|  |  |
| --- | --- |
| Type | Value |
| KFold | 5 |
| Epochs | 30 |
| Train : Valid | 0.95 : 0.05 |
| Batch Size | 32 |
| Optimizer | SGD |
| Learning rate | 0.1 |
| Loss | MSE Loss |

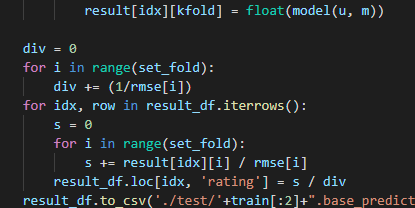
Generalization을 위해 Ensemble을 적용하기 위해 Dataset에 KFold를 적용했습니다. 다만 data가 부족해 충분한 학습이 안 될 것을 우려해 data split은 train : valid = 0.95 : 0.05로 나누었고 K-Fold 20번의 반복은 시간이 오래 걸리기에 5번의 fold가 끝나면 Train을 끝내 얻어낸 5개의 model에 대해서만 ensemble을 적용했습니다. 이후 각 Fold에서 충분한 학습을 위해 Epochs는 50으로 설정했습니다. Mini-batch를 위해 batch size는 32로 설정하였고 latent Factor의 수는 50, Loss는 RMSE를 위한 MSELoss 그리고 optimizer는 SGD에 learning rate는 0.1로 설정하였습니다.

**- Best Model**

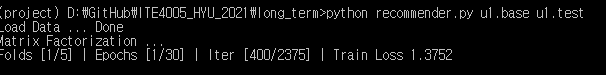


학습을 하는 과정에서 개발자는 어떠한 Model이 최적의 model인지 판단을 할 수 없습니다. 따라서 Validation set을 이용해 MSE loss를 구하고 이를 이용해 각 epoch마다 best MSE를 기록, 해당 model을 fold와 train파일 기준으로 저장하도록 만들었습니다.

**- Ensemble**

5개의 model을 통해 얻어진 값을 각 model의 MSE를 통해 Soft Voting Ensemble을 했습니다. MSE는 낮을수록 좋은 값이기에 MSE의 역수를 이용해서 구현했습니다.

**3. Instructions for compiling source codes at TA's computer**



파일 경로와 이름 설정을 위해 base 파일과 test파일은 해당 소스 파일이 위치한 곳 data-2 폴더 안에 있어야 하며 best model은 model 폴더 안에 base파일과 fold 정보를 가지고 저장이 되게 됩니다. 다만 이때 base파일은 u0~u9의 이름으로 해야 합니다. 그렇게 학습이 끝나고 prediction은 존재하는 test folder 안에 저장이 되게 됩니다.

**4. Any other specification of implementation and testing**

|  |  |
| --- | --- |
| Library | Version |
| Python | 3.8.10 |
| Numpy | 1.19.2 |
| Pandas | 1.2.4 |
| Pytorch | 1.4.0 |
| Scikit-learn | 0.24.2 |

Anaconda 가상 환경에서 실행했습니다.

**5. References**

- Matrix Factorization 그림 : <https://yeomko.tistory.com/5>

- SGD 그림 : https://twinw.tistory.com/247