인공지능 Final Project

소프트웨어학부 2017012251 윤영훈

학습 과정 소개

STEP 0

Basic Matrix Factorization

Validation train : validation	Regularization weight_decay	learning rate lr	Number of loop epoch
X	X	0.01	20

↓

STEP 1

Matrix Factorization with Validation

Validation train : validation	Regularization weight_decay	learning rate lr	Number of loop epoch
9:1	X	0.01	20

↓

STEP 2

Matrix Factorization with Validation + Regularization

Validation train : validation	Regularization weight_decay	learning rate lr	Number of loop epoch
9:1	1e-5	0.01	20

↓

STEP 3

Matrix Factorization with Validation + Regularization + learning rate + epoch

Validation train : validation	Regularization weight_decay	learning rate lr	Number of loop epoch
9:1	1e-5	0.005	8

STEP 0 : Basic Matrix Factorization

Validation train : validation	Regularization weight_decay	learning rate lr	Number of loop epoch
X	X	0.01	20

```
Epoch : 0
train cost : 3.098190
Epoch : 1
train cost : 1.318277
Epoch : 2
train cost : 1.119654
Epoch : 3
train cost : 1.033287
Epoch : 4
train cost : 0.986350
Epoch : 5
train cost : 0.958965
Epoch : 6
train cost : 0.938478
Epoch : 7
train cost : 0.923798
Epoch : 8
train cost : 0.910650
Epoch : 9
train cost : 0.902971
Epoch : 10
train cost : 0.892057
Epoch : 11
train cost : 0.887323
Epoch : 12
train cost : 0.880996
Epoch : 13
train cost : 0.873433
Epoch : 14
train cost : 0.867596
Epoch : 15
train cost : 0.863815
Epoch : 16
train cost : 0.860175
Epoch : 17
train cost : 0.856692
Epoch : 18
train cost : 0.851183
Epoch : 19
train cost : 0.848369
```

```
4.699000835418701
3.0330612659454346
3.6023359298706055
4.194400310516357
3.1987996101379395
3.772954225540161
3.617841958999634
2.94840931892395
4.986368179321289
0.3284813165664673
```

loss는 MSELoss를 sqrt 처리하여 RMSE를 구현한 뒤 계산했다.

강의 PDF에 나와있던 Matrix Factorization code를 활용하여 Basic Model을 만들었다. train cost 자체는 잘 학습하지만 Validation data가 구분되어있지 않아 train data 외의 다른 data를 받았을 때 잘 처리할 수 있는지, over-fitting이 발생하는지 확인할 수 없다.

STEP 1 : Matrix Factorization with Validation

Validation train : validation	Regularization weight_decay	learning rate lr	Number of loop epoch
9:1	X	0.01	20

```
Epoch : 0
train cost : 3.263299
valid cost : 1.698837
Epoch : 1
train cost : 1.362929
valid cost : 1.391396
Epoch : 2
train cost : 1.120773
valid cost : 1.351762
Epoch : 3
train cost : 1.033078
valid cost : 1.334615
Epoch : 4
train cost : 0.983491
valid cost : 1.322868
Epoch : 5
train cost : 0.954739
valid cost : 1.315504
Epoch : 6
train cost : 0.932254
valid cost : 1.299390
Epoch : 7
train cost : 0.916962
valid cost : 1.311719
Epoch : 8
train cost : 0.901054
valid cost : 1.315562
Epoch : 9
train cost : 0.888512
valid cost : 1.292747
Epoch : 10
train cost : 0.881614
valid cost : 1.296141
Epoch : 11
train cost : 0.874650
valid cost : 1.277159
Epoch : 12
train cost : 0.865139
valid cost : 1.292590
Epoch : 13
train cost : 0.858602
valid cost : 1.306729
Epoch : 14
train cost : 0.854809
valid cost : 1.287402
Epoch : 15
train cost : 0.852692
valid cost : 1.283705
Epoch : 16
train cost : 0.844379
valid cost : 1.304718
Epoch : 17
train cost : 0.843992
valid cost : 1.284292
Epoch : 18
train cost : 0.842098
valid cost : 1.290491
Epoch : 19
train cost : 0.837996
valid cost : 1.286437
```

4.564311981201172
1.63540780544281
3.835482120513916
3.934006929397583
2.523132562637329
2.8479318618774414
4.095048904418945
3.50024151802063
4.0505805015563965
1.7800533771514893

train : validation
9 : 1

```
Epoch : 0
train cost : 3.330713
valid cost : 1.811105
Epoch : 1
train cost : 1.398394
valid cost : 1.459933
Epoch : 2
train cost : 1.129369
valid cost : 1.389429
Epoch : 3
train cost : 1.031137
valid cost : 1.372632
Epoch : 4
train cost : 0.980316
valid cost : 1.347667
Epoch : 5
train cost : 0.949634
valid cost : 1.342988
Epoch : 6
train cost : 0.924814
valid cost : 1.335999
Epoch : 7
train cost : 0.910686
valid cost : 1.326009
Epoch : 8
train cost : 0.895633
valid cost : 1.334323
Epoch : 9
train cost : 0.886734
valid cost : 1.342649
Epoch : 10
train cost : 0.876680
valid cost : 1.326429
Epoch : 11
train cost : 0.868145
valid cost : 1.334942
Epoch : 12
train cost : 0.864203
valid cost : 1.320764
Epoch : 13
train cost : 0.857008
valid cost : 1.330456
Epoch : 14
train cost : 0.851488
valid cost : 1.329864
Epoch : 15
train cost : 0.845769
valid cost : 1.327301
Epoch : 16
train cost : 0.840023
valid cost : 1.319100
Epoch : 17
train cost : 0.837176
valid cost : 1.333120
Epoch : 18
train cost : 0.836755
valid cost : 1.327615
Epoch : 19
train cost : 0.829568
valid cost : 1.317078
```

3.566685199737549
1.5816470384597778
4.92038631439209
2.558687210083008
2.8821775913238525
1.5340135097503662
4.128353118896484
2.825493335723877
5.219149112701416
2.7422924041748047

train : validation
8 : 2

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

train_data.data_pd, valid_data.data_pd = train_test_split(train_data.data_pd, test_size=0.1, shuffle=True, random_state=34)

train_data.items = torch.LongTensor(train_data.data_pd['itemId'])
train_data.users = torch.LongTensor(train_data.data_pd['userId'])
train_data.ratings = torch.FloatTensor(train_data.data_pd['rating'])

valid_data.items = torch.LongTensor(valid_data.data_pd['itemId'].values)
valid_data.users = torch.LongTensor(valid_data.data_pd['userId'].values)
valid_data.ratings = torch.LongTensor(valid_data.data_pd['rating'].values)


train_loader = DataLoader(train_data, batch_size=args.batch_size, shuffle=True)
valid_loader = DataLoader(valid_data, batch_size=args.batch_size, shuffle=True)
```

sklearn의 train_test_split를 이용하여 train data 중 일부를 validation data로 전환하여 validation test를 진행했다. 이때 train data와 validation data의 비율을 각각 9:1, 8:2로 설정하여 비교해본 결과 9:1로 설정했을 때 valid cost와 test data의 예상값이 더 좋게 return 되었으므로 test_size를 0.1로 설정했다. (8:2의 test data 예상값 중 9번째 값이 5를 초과 → 잘못된 예상값)

STEP 2 : Matrix Factorization with Validation + Regularization

Validation train : validation	Regularization weight_decay	learning rate lr	Number of loop epoch
9:1	1e-5	0.01	20

```
Epoch : 0
train cost : 2.409789
valid cost : 1.344694
Epoch : 1
train cost : 1.195123
valid cost : 1.265437
Epoch : 2
train cost : 1.103887
valid cost : 1.260099
Epoch : 3
train cost : 1.068003
valid cost : 1.258165
Epoch : 4
train cost : 1.046686
valid cost : 1.255831
Epoch : 5
train cost : 1.032443
valid cost : 1.245132
Epoch : 6
train cost : 1.016283
valid cost : 1.273765
Epoch : 7
train cost : 1.010129
valid cost : 1.260536
Epoch : 8
train cost : 0.997423
valid cost : 1.228813
Epoch : 9
train cost : 0.985676
valid cost : 1.243588
Epoch : 10
train cost : 0.976395
valid cost : 1.230224
Epoch : 11
train cost : 0.964188
valid cost : 1.225408
Epoch : 12
train cost : 0.960058
valid cost : 1.226400
Epoch : 13
train cost : 0.952534
valid cost : 1.221592
Epoch : 14
train cost : 0.947618
valid cost : 1.217781
Epoch : 15
train cost : 0.948239
valid cost : 1.216970
Epoch : 16
train cost : 0.943674
valid cost : 1.215554
Epoch : 17
train cost : 0.942773
valid cost : 1.212333
Epoch : 18
train cost : 0.944797
valid cost : 1.222410
Epoch : 19
train cost : 0.942270
valid cost : 1.210212
```

```
4.071491718292236
3.1007673740386963
4.259515762329102
3.9661059379577637
3.429273843765259
1.686652660369873
3.146862268447876
1.9349384307861328
4.215606212615967
3.6525871753692627
```

weight_decay
1e-5

```
Epoch : 0
train cost : 2.375932
valid cost : 1.452916
Epoch : 1
train cost : 1.427343
valid cost : 1.419497
Epoch : 2
train cost : 1.386546
valid cost : 1.395610
Epoch : 3
train cost : 1.350070
valid cost : 1.372207
Epoch : 4
train cost : 1.323791
valid cost : 1.351545
Epoch : 5
train cost : 1.299466
valid cost : 1.329535
Epoch : 6
train cost : 1.284776
valid cost : 1.329338
Epoch : 7
train cost : 1.277490
valid cost : 1.317409
Epoch : 8
train cost : 1.266754
valid cost : 1.317896
Epoch : 9
train cost : 1.259939
valid cost : 1.310292
Epoch : 10
train cost : 1.254209
valid cost : 1.307194
Epoch : 11
train cost : 1.253899
valid cost : 1.291321
Epoch : 12
train cost : 1.250303
valid cost : 1.301611
Epoch : 13
train cost : 1.248840
valid cost : 1.299093
Epoch : 14
train cost : 1.248939
valid cost : 1.293964
Epoch : 15
train cost : 1.252823
valid cost : 1.298704
Epoch : 16
train cost : 1.251984
valid cost : 1.297372
Epoch : 17
train cost : 1.252723
valid cost : 1.296829
Epoch : 18
train cost : 1.251070
valid cost : 1.295597
Epoch : 19
train cost : 1.250326
valid cost : 1.291702
```

```
4.683922290802002
3.5322494506835938
4.424973964691162
2.1982638835906982
3.2783045768737793
1.5668359948883606
4.843100070953369
1.9277609586715698
3.5485880374908447
2.7376210689544678
```

weight_decay
1e-4

```
optimizer = torch.optim.Adam(model.parameters(), lr=0.005, weight_decay = 1e-5)
criterion = nn.MSELoss()
```

Regularization을 위해 Adam optimizer에 weight_decay를 추가했다. 이때 1e-5, 1e-4를 비교했고, 1e-5에서 괄목할만한 valid cost의 감소 (기존 1.3 대비 약 -0.1) 가 발생하였으므로 weight_decay를 1e-5로 설정했다.

STEP 3 : Matrix Factorization with Validation + Regularization + learning rate + epoch

Validation train : validation	Regularization weight_decay	learning rate lr	Number of loop epoch
9:1	1e-5	0.005	8

```
Epoch : 0
train cost : 3.380319
valid cost : 1.876640
Epoch : 1
train cost : 1.348295
valid cost : 1.261962
Epoch : 2
train cost : 1.035751
valid cost : 1.205619
Epoch : 3
train cost : 0.954042
valid cost : 1.203078
Epoch : 4
train cost : 0.918134
valid cost : 1.209182
Epoch : 5
train cost : 0.888818
valid cost : 1.212139
Epoch : 6
train cost : 0.875698
valid cost : 1.216182
Epoch : 7
train cost : 0.860323
valid cost : 1.207463
Epoch : 8
train cost : 0.853390
valid cost : 1.218601
Epoch : 9
train cost : 0.843744
valid cost : 1.224148
Epoch : 10
train cost : 0.837480
valid cost : 1.236767
Epoch : 11
train cost : 0.830451
valid cost : 1.220189
Epoch : 12
train cost : 0.823638
valid cost : 1.227840
Epoch : 13
train cost : 0.821502
valid cost : 1.224391
Epoch : 14
train cost : 0.815278
valid cost : 1.226325
Epoch : 15
train cost : 0.814565
valid cost : 1.214402
Epoch : 16
train cost : 0.810181
valid cost : 1.228877
Epoch : 17
train cost : 0.806842
valid cost : 1.224271
Epoch : 18
train cost : 0.804362
valid cost : 1.227092
Epoch : 19
train cost : 0.801477
valid cost : 1.228496
```

→

```
Epoch : 0
train cost : 3.450574
valid cost : 1.922857
Epoch : 1
train cost : 1.370532
valid cost : 1.267416
Epoch : 2
train cost : 1.046684
valid cost : 1.217208
Epoch : 3
train cost : 0.959364
valid cost : 1.211563
Epoch : 4
train cost : 0.919326
valid cost : 1.216504
Epoch : 5
train cost : 0.893639
valid cost : 1.204656
Epoch : 6
train cost : 0.880324
valid cost : 1.210023
Epoch : 7
train cost : 0.868040
valid cost : 1.211083
```

```
4.539312839508057
1.728360891342163
4.526708126068115
3.341174840927124
3.0922765731811523
1.6287461519241333
3.501698970794677
2.3605692386627197
3.968982696533203
2.2797932624816895
```

<Fig 1>

기존 lr=0.01 대비 learning rate를 0.005로 낮춘 뒤 학습을 진행했을 때 대략 epoch= 2, 3, 4, 7에서 여태껏 비교했던 valid cost 중 가장 작은 값이 나왔다. 이 중 train data를 최대한 많이 학습한 epoch = 7을 선택해 hyper parameter tuning을 완료했다. 최종적으로 python run.py를 통해 result.txt에 저장된 값이 Fig 1임을 확인할 수 있다.

피드백

초기 계획했던 변경점은 Regularization / Validation / Hyper parameter tuning / Model structure change 총 4가지였다. 이 중 Model structure change의 경우 Embedding layer를 적용하려고 시도했으나 지속적인 ERROR 발생 및 과도한 학습 시간 (train.py 1회 실행 당 약 30분 소모)로 인해 결국 기본적인 Matrix factorization model을 그대로 사용하게 되었다. Model structure change를 성공적으로 적용했을 경우 valid cost를 더욱 줄일 수 있었을 것으로 예상된다. 그 외 Regularization, Validation, Hyper parameter tuning의 경우에는 valid cost가 눈에 띄게 감소하는 것을 확인할 수 있어서 성공적이었다고 생각한다.