Recommendation System

Book-Crossing 데이터를 활용한 rating 예측 알고리즘 개발

디지털 애널리틱스학과
김호헌 손영관 안재일 조예린

Index

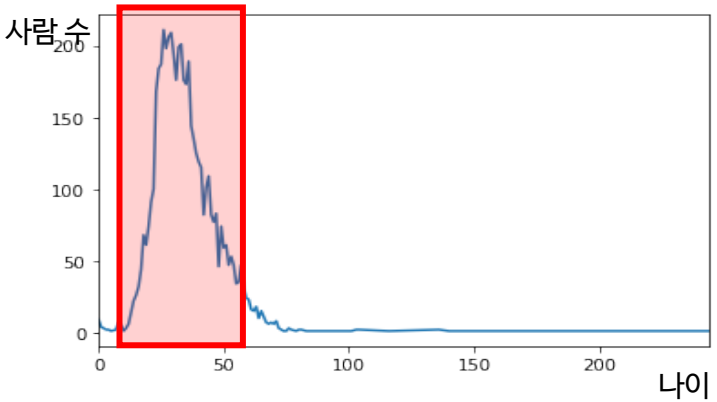
Deep Learning 시도 - 데이터 전처리

✓ Rating 관련 파생변수 생성

- 주어진 변수들을 통해 user의 rating 특성을 잘 표현하는 파생변수 추가 (**Data Contamination → 제거**)

파생변수 명	설명
Rating-Mean-Users	각 유저가 입력한 평점의 평균
Rating-Mean-Books	각 책이 받은 평점의 평균
Rating-Cnt-Books	각 책이 받은 평점의 개수
Rating-Weighted	각 책이 받은 평점의 가중 총점

✓ Age 필터링




```
def CI_range(list_, l_limit=0.01, u_limit=99.99):
    '''To calculate confidence interval

    Parameters
    ------
    list_: array, Pandas.Series, list
    l_limit: float,
        defualt: 0.001
    u_limit: float,
        default: 99.99

    Return
    ------
    tuple = (lower value, upper value)


    Example
    ------
    CI_range(book.pub)

    '''
    return np.percentile(list_, l_limit), np.percentile(list_, u_limit)
```

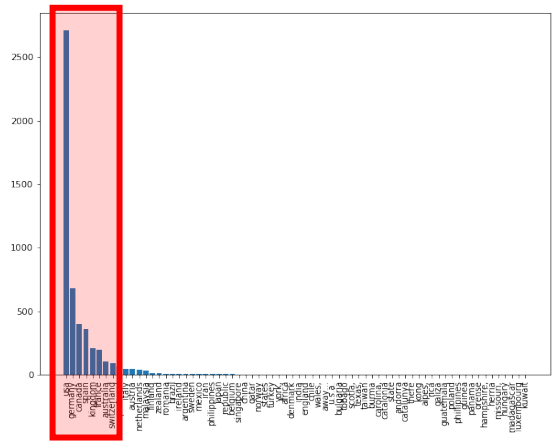
- 신뢰구간을 알 수 있는 함수를 통해,
  전체 데이터 중 5% ~ 95%에 해당하는
  데이터만 추출

- 12세 부터 57세 까지의 유저만 사용

Deep Learning 시도 - 데이터 전처리

✓ Location 나라 추출 및 그룹핑

- Location에서 유저의 나라 추출
- 전체 책의 90.53%를 상위 7개의 나라가 차지 (미국, 독일, 캐나다, 스페인, 영국, 프랑스, 호주)
- 위 7개의 나라의 관련해서 오류 데이터 전처리 진행 (‘texas,’ →’ usa’ / ‘england’ → ‘uk’)
- 그 외의 나라들은 Other로 그룹핑



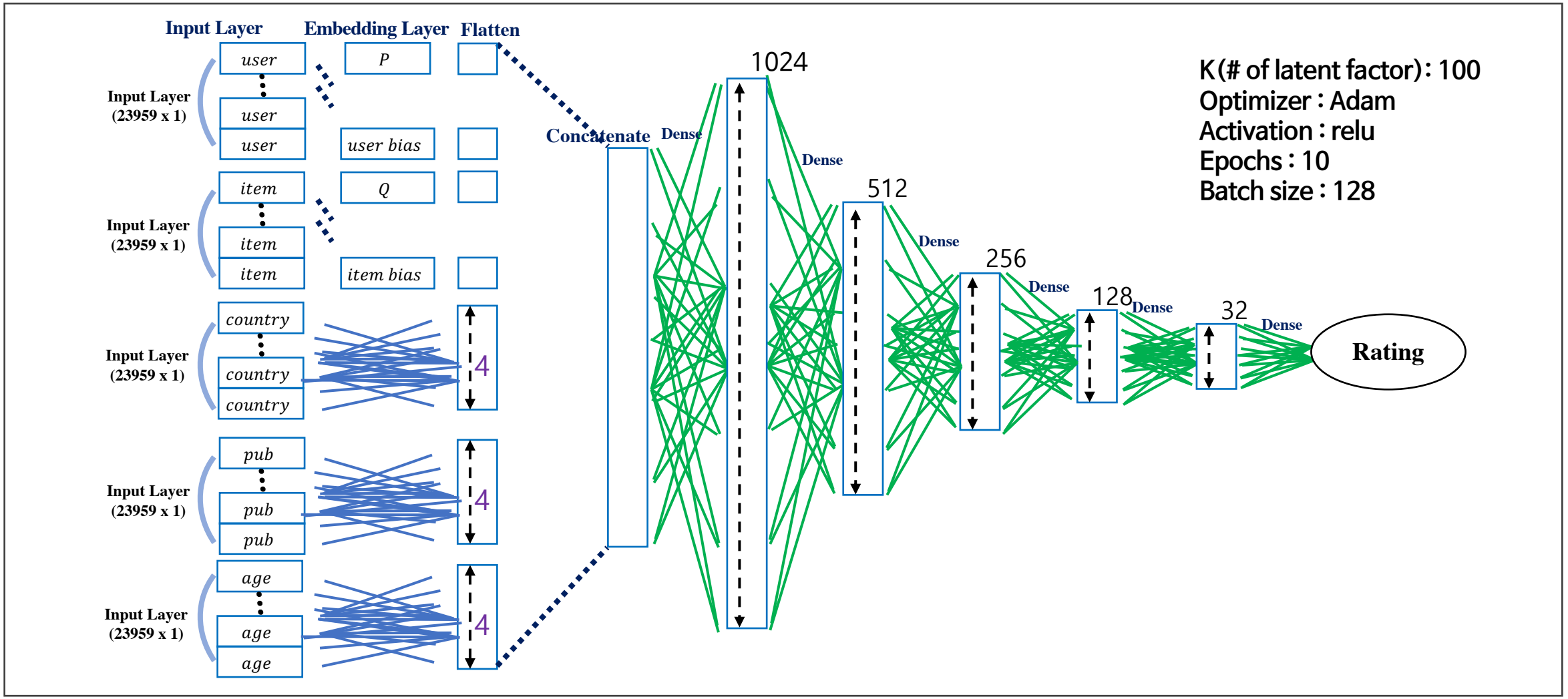
✓ Publisher 전처리

- 동일 출판사이지만 ‘Publishing’, ‘Incorporated’ 같은 단어들이 ‘Pub.’, ‘Inc.’약어로 되어 다르게 인식되는 출판사명 통일
- 총 6,593개에서 5,771개까지 정리

✓ Year of Publication 전처리

- 이상치인 1378년, 1900년, 2006년 이상 제거

Deep Learning 시도 - 모델 아키텍처



Deep Learning 시도

```
result = model.fit(
  x=[X_train["User-ID2"].values, X_train["ISBN2"].values,
    train_year_pub, train_country, train_age, train_pub],
  y=y_train.values - mu,
  epochs=epochs,
  batch_size=12,
  validation_data=(
    [X_test["User-ID2"].values, X_test["ISBN2"].values,
    test_year_pub, test_country, test_age, test_pub],
    y_test.values - mu
  )
)
```

```
Train on 23959 samples, validate on 10269 samples
Epoch 1/10
23959/23959 [==============================] - 274s 11ms/sample - loss: 447.8514 - mean_squared_error: 42269140.0000 - RMSE: 44
0.3481 - val_loss: 11.4171 - val_mean_squared_error: 3.3791 - val_RMSE: 1.8022
Epoch 2/10
23959/23959 [==============================] - 266s 11ms/sample - loss: 12.2839 - mean_squared_error: 2670.2334 - RMSE: 2.9612
- val_loss: 11.2285 - val_mean_squared_error: 3.3373 - val_RMSE: 1.7863
Epoch 3/10
23959/23959 [==============================] - 266s 11ms/sample - loss: 10.7615 - mean_squared_error: 3.4087 - RMSE: 1.8055 - v
al_loss: 10.2177 - val_mean_squared_error: 3.3681 - val_RMSE: 1.7904
Epoch 4/10
23959/23959 [==============================] - 273s 11ms/sample - loss: 9.6286 - mean_squared_error: 3.4091 - RMSE: 1.8043 - va
l_loss: 8.9566 - val_mean_squared_error: 3.3923 - val_RMSE: 1.7979
Epoch 5/10
23959/23959 [==============================] - 270s 11ms/sample - loss: 22.4557 - mean_squared_error: 385296.9688 - RMSE: 15.69
38 - val_loss: 8.1762 - val_mean_squared_error: 3.3358 - val_RMSE: 1.7822
Epoch 6/10
23959/23959 [==============================] - 278s 12ms/sample - loss: 14.4261 - mean_squared_error: 49909.7891 - RMSE: 7.9205
- val_loss: 8.0231 - val_mean_squared_error: 3.3341 - val_RMSE: 1.7901
Epoch 7/10
23959/23959 [==============================] - 288s 12ms/sample - loss: 125.4629 - mean_squared_error: 25450100.0000 - RMSE: 11
8.1165 - val_loss: 8.7248 - val_mean_squared_error: 3.3255 - val_RMSE: 1.7855
```

→ 약 1.79 높은 RMSE로 낮은 성능

→ sparse한 데이터의 한계

→ 다른 추천 알고리즘 탐색

PMF(Probabilistic Matrix Factorization)란?

- Matrix factorization 의 핵심: R ≈ U·V
- R matrix에서 파생된 U, V matrix을 Gaussian distribution을 이용하여 추정하는 모델

$$\mathcal{N}(x|\mu,\sigma^2) \uparrow U_i \quad \begin{bmatrix} U_{11} & \cdots & U_{1D} \\ \vdots & \ddots & \vdots \\ U_{N1} & \cdots & U_{ND} \end{bmatrix}$$

$U = D \times N$

(Latent user matrix)

$$\mathcal{N}(x|\mu,\sigma^2) \uparrow V_i \quad \begin{bmatrix} V_{11} & \cdots & V_{1M} \\ \vdots & \ddots & \vdots \\ V_{D1} & \cdots & V_{DM} \end{bmatrix}$$

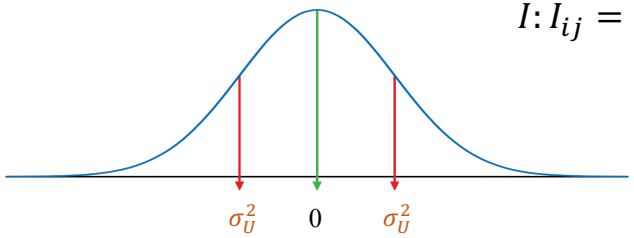
$V^T = M \times D$

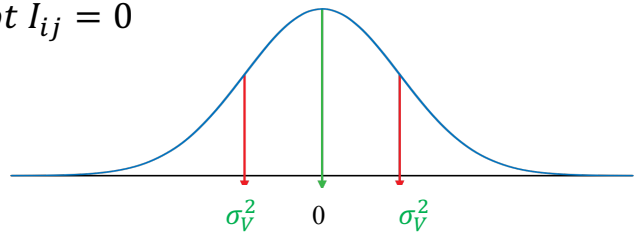
(Latent book matrix)

$$p(U|\sigma_U^2) = \prod_{i=1}^{N} \mathcal{N}(U_i|0,\sigma_U^2 I) \qquad p(V|\sigma_V^2) = \prod_{i=1}^{N} \mathcal{N}(V_i|0,\sigma_V^2 I)$$

$where\ \sigma_U^2 : variance\ of\ U\ matrix,\ \sigma_V^2 :\ variance\ of\ V\ matrix$

$I : I_{ij} = 1\ if\ entry\ was\ observed, if\ not\ I_{ij} = 0$





PMF(Probabilistic Matrix Factorization)란?

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|U_i^T V_j,\sigma^2)\right]^{I_{ij}}$$

해석: Matrix factorization에서 파생될 수 있는 U, V에 대해서, 각각의 행 벡터 U, V 두 곱인 Rij 의 확률

$$p(U|\sigma_U^2) = \prod_{i=1}^{N}\mathcal{N}(U_i|0,\sigma_U^2 I) \qquad p(V|\sigma_V^2) = \prod_{i=1}^{N}\mathcal{N}(V_i|0,\sigma_V^2 I)$$

- posterior probabilistic: $p(U,V\,|R,\sigma^2,\sigma_U^2,\sigma_V^2)$
- log of the posterior probabilistic : $\ln p(U,V|R,\sigma^2,\sigma_U^2,\sigma_V^2)$

$$= -\frac{1}{2\sigma^2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\left(R_{ij}-U_i^T V_i\right)^2 - \frac{1}{2\sigma_u^2}\sum_{i=1}^{N} U_i^T U_i - \frac{1}{2\sigma_v^2}\sum_{j=1}^{N} V_i^T V_i - \frac{1}{2}\left(\left(\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\right)\ln\sigma + ND\ln\sigma_u^2 + MD\ln\sigma_V^2\right) + c$$

$$\underset{R\in\mathbb{R}^{NXM},\sigma^2,\sigma_U^2,\sigma_V^2\in\mathbb{R}}{\operatorname{argmax}} \ln p(U,V|R,\sigma^2,\sigma_U^2,\sigma_V^2)$$

$$\Leftrightarrow \underset{R\in\mathbb{R}^{NXM},\sigma^2,\sigma_U^2,\sigma_V^2\in\mathbb{R}}{\min} -\ln p(U,V|R,\sigma^2,\sigma_U^2,\sigma_V^2) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\left(R_{ij}-U_i^T V_i\right)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N}\|U_i\|^2{}_{fro} + \frac{\lambda_v}{2}\sum_{i=1}^{N}\|V_j\|^2{}_{fro}$$

PMF 핵심 코드 - Objective function

$$\underset{R\in\mathbb{R}^{NXM},\sigma^2,\sigma_U^2,\sigma_V^2\in\mathbb{R}}{\operatorname{argmax}} \ln p(U,V|R,\sigma^2,\sigma_U^2,\sigma_V^2)$$

$$\Leftrightarrow -\ln p(U,V|R,\sigma^2,\sigma_U^2,\sigma_V^2)$$

$$= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\left(R_{ij}-U_i^T V_i\right)^2 + \underbrace{\frac{\lambda_U}{2}\sum_{i=1}^{N}\|U_i\|^2{}_{fro} + \frac{\lambda_v}{2}\sum_{i=1}^{N}\|V_j\|^2{}_{fro}}_{Column\ vectors\ U_i,V_j}$$

코드화

```
def loss(self):
    # the loss function of the model
    loss = np.sum(self.I*(self.R-np.dot(self.U, self.V.T))**2)
        + self.lambda_alpha*np.sum(np.square(self.U))
        + self.lambda_beta*np.sum(np.square(self.V))

    return loss
```

PMF 최종 결과

```
pmf = PMF(R=R,
          lambda_alpha = 0.001,
          lambda_beta = 0.01,
          latent_size = 30,
          momuntum = 0.07,
          lr=0.00015,
          iters=100)


U, V, train_loss, valid_rmse = pmf.train(train_data=train_new, vali_data=test_new, verbose=True)
```

```
traning iteration: 0 ,loss: 2204262.277798, vali_rmse: 1.452397
traning iteration: 1 ,loss: 2202565.545713, vali_rmse: 1.451847
traning iteration: 2 ,loss: 2200831.234297, vali_rmse: 1.451293
traning iteration: 3 ,loss: 2199064.862626, vali_rmse: 1.450738
traning iteration: 4 ,loss: 2197265.475287, vali_rmse: 1.450183
traning iteration: 5 ,loss: 2195431.659552, vali_rmse: 1.449628
traning iteration: 6 ,loss: 2193561.967610, vali_rmse: 1.449075
traning iteration: 7 ,loss: 2191654.947394, vali_rmse: 1.448523
traning iteration: 8 ,loss: 2189709.146257, vali_rmse: 1.447975
traning iteration: 9 ,loss: 2187723.113005, vali_rmse: 1.447432
traning iteration: 10 ,loss: 2185695.400070, vali_rmse: 1.446895
traning iteration: 11 ,loss: 2183624.565955, vali_rmse: 1.446367
traning iteration: 12 ,loss: 2181509.177967, vali_rmse: 1.445849
traning iteration: 13 ,loss: 2179347.815238, vali_rmse: 1.445345
traning iteration: 14 ,loss: 2177139.072015, vali_rmse: 1.444858
traning iteration: 15 ,loss: 2174881.561244, vali_rmse: 1.444390
traning iteration: 16 ,loss: 2172573.918407, vali_rmse: 1.443945
traning iteration: 17 ,loss: 2170214.805617, vali_rmse: 1.443528
traning iteration: 18 ,loss: 2167802.915941, vali_rmse: 1.443143
traning iteration: 19 ,loss: 2165336.977939, vali_rmse: 1.442795
traning iteration: 20 ,loss: 2162815.760378, vali_rmse: 1.442489
traning iteration: 21 ,loss: 2160238.077086, vali_rmse: 1.442231
traning iteration: 22 ,loss: 2157602.791912, vali_rmse: 1.442028
traning iteration: 23 ,loss: 2154908.823742, vali_rmse: 1.441886
traning iteration: 24 ,loss: 2152155.151509, vali_rmse: 1.441813
traning iteration: 25 ,loss: 2149340.819147, vali_rmse: 1.441816
convergence at iterations: 25, patience:1
traning iteration: 26 ,loss: 2146464.940414, vali_rmse: 1.441904
convergence at iterations: 26, patience:2
traning iteration: 27 ,loss: 2143526.703530, vali_rmse: 1.442086
convergence at iterations: 27, patience:3
```

최종 RMSE : 1.4418

→ PMF결과를 Deep learning input으로 활용해보았으나,
Deep Learning은 여전히 낮은 성능을 보여 제외

Lesson Learned

- ⭐ Deep Learning 〉 Maching Learning?
- ⭐ 많은 input features 〉 적은 input features?
- ⭐ Unseen data를 예측한다는 가정 하의 Feature Engineering
- ⭐ 데이터의 Sparse한 특징을 잘 다루는 것이 Key

감사합니다