Final Project

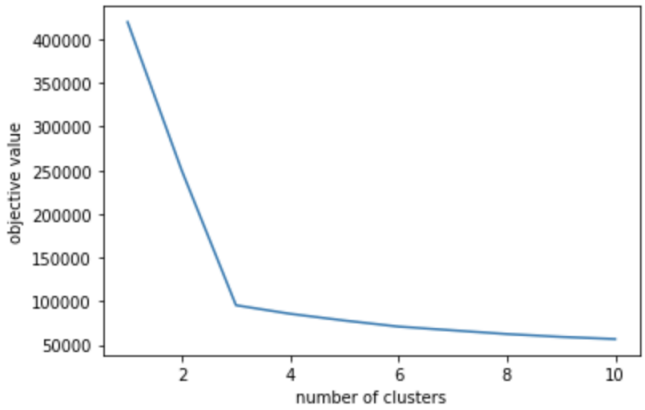
Wei Lu

December 18, 2019

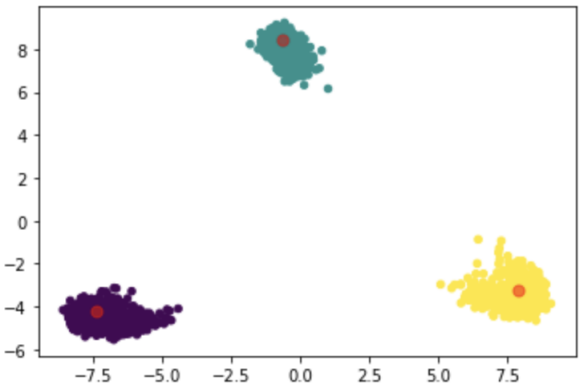
1 Method: matrix factorization and regression

1.1 Simplify the dataset

We want to predict users' ratings based on their surfing history and rating records. So we first divide the users into groups according to their preferences. We use users' surfing history to do clustering.



From the plot of the objective function, it's natural to select the number of clusters to be equal to 3 by the "elbow" method. To see this more clearly, we plot the clusters after using PCA to reduce the dimension of surfing history into 2. It's clear to see the 3 clusters from the plot below.



We assume users in the same cluster have similar preferences and give similar ratings to products. So in the following sections, we build models and make prediction separately for each cluster.

1.2 Prepare training and validation data

For each cluster, we first select users with indices bigger than 2999. These users have no rating records and we will predict their features from surfing history. For the remaining users, we randomly select 80% from them as training data and 20% as validation. We treat these 20% users as users without rating record and predict their ratings from their surfing history.

1.3 Build matrices of user features and product features

Let R be rating matrix where R_{ij} represents the rating of i-th user on the j-th product. We assume R can be factorized as

$$R = UP$$

where each row of U represents the feature of a user and each column P represents the feature of a product. The number of columns of U, denoted by k, represents the latent dimension of feature space and is a hyper-parameter. We choose MSE as our objective function. Using the training data from the last step and stochastic gradient descent we can find U and P. Then for users with rating records, we can use this factorization to predict missing values in R.

1.4 Compute user features of users without rating record

For the last 1500 users who have no rating record, we don't have their user features yet. But we can predict their features from their surfing history through linear regression. Assume user features U can be factorized as

$$U = HW$$

where H is surfing history. We assume there is a linear relation between U and H given by a matrix W. We use the user features obtained in the last step and the corresponding surfing history to find W by using `sklearn.linear_model.Ridge`. You can see the code in the jupyter notebook.

1.5 Train the model

We define the function `initialize(n_users, n_products, k)` to initialize user features and product features and function `train_sgd(cluster, model, epochs, train_ratio=0.8)` to train our model on the 3 clusters respectively. We choose $k = 10$ and train for 10 epochs. The training and validation loss is as follows.

```
42 sgd_model = initialize(3000, 100, k)
43 train_sgd(cluster3, sgd_model, 10)
```

```
Loss after epoch #1 is: train/5.265553749941821 --- val/5.110322883873916
Loss after epoch #2 is: train/0.678848826258615 --- val/0.7222135468606791
Loss after epoch #3 is: train/0.4794642588110919 --- val/0.6234218961686513
Loss after epoch #4 is: train/0.46681414015452555 --- val/0.6188082736360139
Loss after epoch #5 is: train/0.4592088607948625 --- val/0.6126935751679262
Loss after epoch #6 is: train/0.4504818557602185 --- val/0.6039239301220153
Loss after epoch #7 is: train/0.43900049179841744 --- val/0.5917016664939155
Loss after epoch #8 is: train/0.4238801751454668 --- val/0.5755503113089551
Loss after epoch #9 is: train/0.40490383119631607 --- val/0.5557695050937422
Loss after epoch #10 is: train/0.3826908275506378 --- val/0.5336535802413506
```

```
1 sgd_model = initialize(3000, 100, k)
2 train_sgd(cluster1, sgd_model, 10)
```

```
Loss after epoch #1 is: train/5.924896088561804 --- val/7.201570117655508
Loss after epoch #2 is: train/0.8404702399870361 --- val/7.007508593247162
Loss after epoch #3 is: train/0.457064593950112 --- val/6.991006623460792
Loss after epoch #4 is: train/0.443099358565436 --- val/6.992072402572778
Loss after epoch #5 is: train/0.4367715607604293 --- val/6.992106482653852
Loss after epoch #6 is: train/0.4302371576780109 --- val/6.991922714146128
Loss after epoch #7 is: train/0.4219102869750084 --- val/6.99164804039267
Loss after epoch #8 is: train/0.41092624157308605 --- val/6.991290731526413
Loss after epoch #9 is: train/0.3967445445474269 --- val/6.990851532050046
Loss after epoch #10 is: train/0.3792341837248756 --- val/6.990345650678639
```

```
1 sgd_model = initialize(3000, 100, k)
2 train_sgd(cluster2, sgd_model, 10)
```

```
Loss after epoch #1 is: train/5.661019511307787 --- val/7.535029187463097
Loss after epoch #2 is: train/0.7574056679368498 --- val/7.524572220362792
Loss after epoch #3 is: train/0.44955259438913264 --- val/7.521890802711111
Loss after epoch #4 is: train/0.4350203534764868 --- val/7.521867823717141
Loss after epoch #5 is: train/0.42820961012543424 --- val/7.521768221433296
Loss after epoch #6 is: train/0.42141438798879827 --- val/7.521710146519897
Loss after epoch #7 is: train/0.41328609484182616 --- val/7.521659353879663
Loss after epoch #8 is: train/0.4033755511464576 --- val/7.521577403432585
Loss after epoch #9 is: train/0.3916541967269166 --- val/7.521436403032617
Loss after epoch #10 is: train/0.37837512750833013 --- val/7.52121895781533
```

As we can see, the training result for cluster3 is very satisfactory (with validation loss 0.53), while the results for other 2 clusters are poor (with validation loss around 7). It implies that there is a clear linear relation between surfing history and user ratings for users in cluster3, but it's not the case for users in cluster1 and cluster2.