Arda Cankat Bati

A53285400

CSE 285 Assignment 1 Report

Task1 Read Prediction

For this task, I first included the data from the rating predictions task to my overall data, making 210K total datapoints. I shuffled the data, and split the train and validation sets as 200K train samples, 10K validation set. Then I doubled the size of each set by sampling books each user hasn’t read. (so that in the end it was 400K train, 20K validation set size). I created feature mappings for each data points (user, book) pair as follows: 0-bias term, 1- maximum Jaccard book similarity, 2- average Jaccard book similarity 3- maximum Jaccard user similarity 4- maximum Jaccard user similarity 5- maximum cosine book similarity, 6- average cosine book similarity, 7- book in popular books set (the set from HW3, with threshold ratio of 0.64).

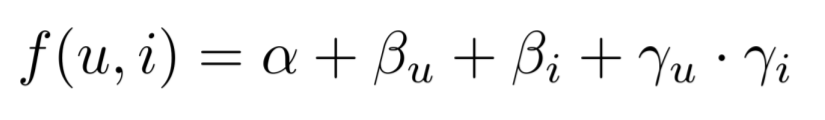
Then, I put the train set through a logistic regressor to get the best feature weights. I used the validation set to decide on the best regularization parameter C (the one that gave the highest validation accuracy and BER). Finally, for the actual test predictions, for each user, I sorted the predictions according to their confidence score. For each user’s read books, I predicted 1 for the first half of the books (with higher confidence scores) and 0 for the lower half. This final prediction is what I uploaded to Kaggle.

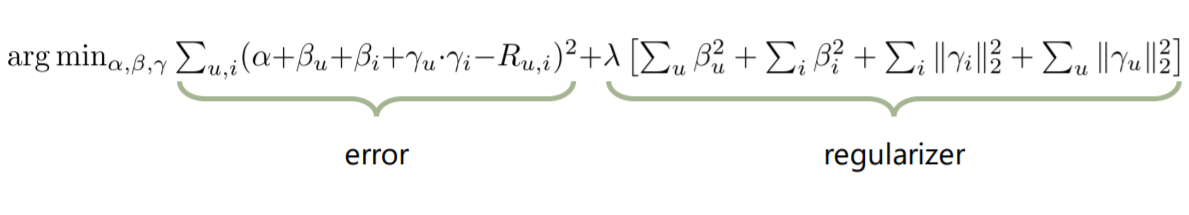
To calculate the Jaccard similarities I used what we already implemented in class. The only difference I made is that while training, I excluded the Jaccard similarity of a pair if the two pairs were the same. I did this to not overfit the trainset. I didn’t to this while testing, because it would be a disadvantage. To calculate the cosine similarities, I set each rating to 1 if it is greater than or equal to the current user’s average, -1 otherwise. I tried working with Pearson similarities as well but didn’t find them very successful. Therefore, I excluded it in the final model. I realized there are around 8K 0 scores, but you can’t actually give 0 score on Goodreads, so they probably mean something else.

Finally, this is my idea on how I could improve: I think in my implementation I had to push C too low, which means higher regularization. I think this was caused by my train and validation sets not being big enough. I could have generated more negative test points to make them bigger. To keep the resulting model balanced, I could increase the weights of positive examples (as there would be fewer of them compared to the negatives).

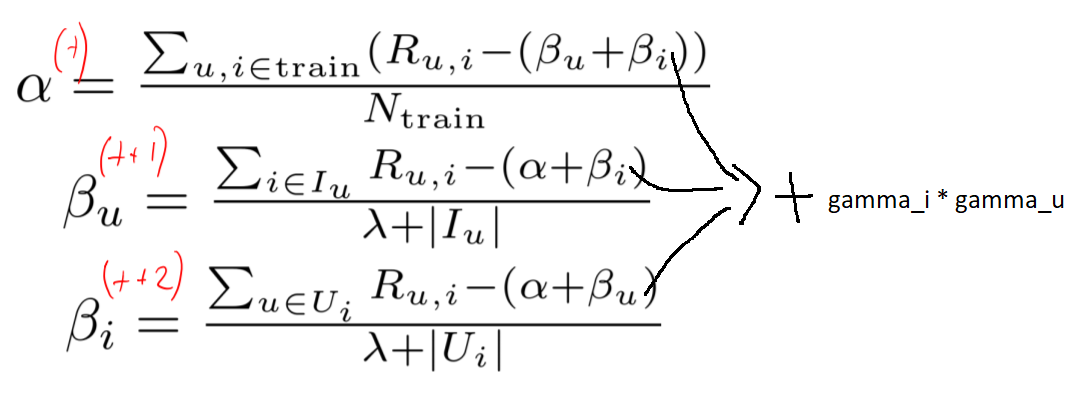
Task 2 Rating Prediction

For this task I used a latent factor model as outlined in lecture 8 slides, with the following formula:

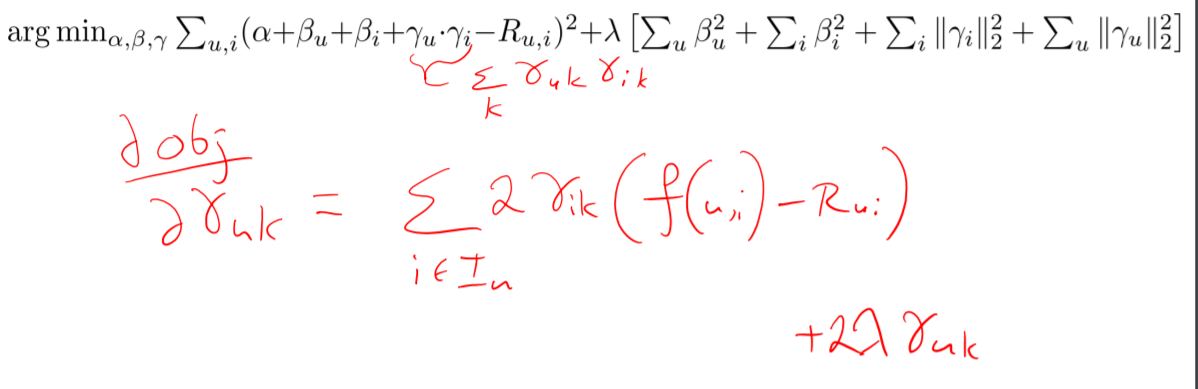


From the data given to us, I used 195K as train and 5K as validation points to calculate the MSE on. I sed a loss function as the one below:

The lambda value I used was lambda = 2.8. I chose the size of the latent factor vectors as k = 4, although I believe higher k’s are possible to use. For the updates I had a loop, in each iteration all values alpha, beta\_u, beta\_i, gamma\_u, gamma\_i got updated **once,** in that order. I used coordinate descent as outlined in lecture 8 for alpha, beta\_u and beta\_i updates. I also included gamma\_i and gamma\_u in the coordinate update equation, as shown below.



For the gamma\_i and gamma\_u I used the gradient descent (simple solution as outlined in lecture 8 slides). The update equation derived in class is as follows (it is for gamma\_u, gamma\_i is similar):

I didn’t use any gradient or coordinate descent libraries, since I wanted to do the implementation on my own. For the starting values, I set alpha = 0, beta\_u = beta\_i = average\_score. For gamma\_u and gamma\_i, I set each element of the vectors to a random value between –0.25 to +0.25. To keep track of how the algorithm is converging and possibly overfitting, I calculated the current MSE on the validation set at each iteration. The MSE starts decreasing as the gradient descent begins, but after a certain point it overfits the trainset and validtion MSE starts increasing. At that stage I stopped the model. I also had an alternative model trained which just had alpha\_simple, beta\_u\_simple, beta\_i\_simple, derived with coordinate descent, no latent factor vectors. I always tried to get better MSE than this simple model with my latent factor model (not so easy).

Finally, after finding the right alpha, beta\_u, beta\_i, gamma\_u, gamma\_i values, I predicted the ratings with the **f(u, i)** function I outlined in the beginning of this section. As the final step, I realized some predictions go out of the boundaries for ratings which is 0 <= rating <= 5. Since this is not possible for Goodreads, in such cases I reverted back to the simple model’s prediction using just: alpha\_simple, beta\_u\_simple, beta\_i\_simple. If that value was not in the correct range as well, (which almost never happened), I predicted the closest value to it, either 0 for f(u, i) < 0, or 5 for predictions f(u, i) > 5.

Overall, I found this task to be much more straightforward than Task 1.