CSC311

Final Project

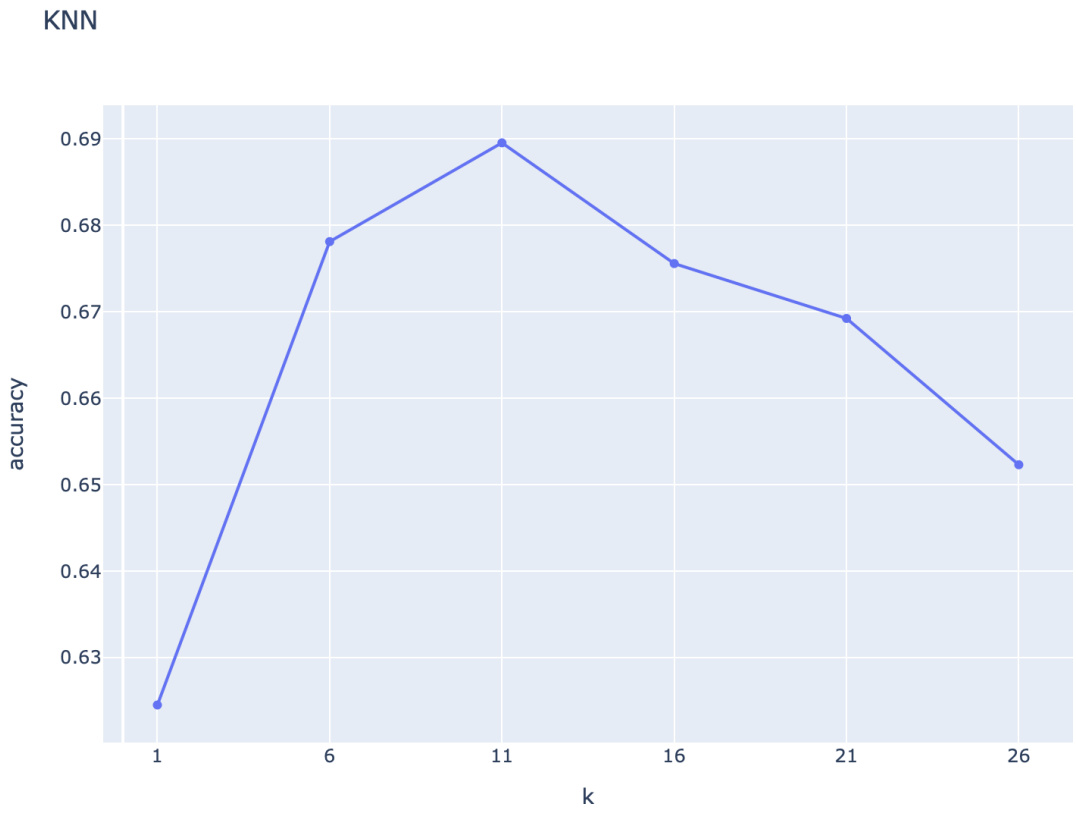
Yuning Wu, Amir Azimi, Bojian Li

November 2020

Part A

Q1 by Yuning Wu

(a) Below are the plot and outputs of user based KNN accuracy.



k	Validation accuracy
1	0.6245
5	0.6781
11	0.6895
16	0.6756
21	0.6692
26	0.6523

item [(b)] The best k in this case is 11, with validation accuracy of 0.6895; test accuracy of 0.6842

(c) The core underlying assumption for item based KNN is that if the question Q_i has the same correct and incorrect performance among students on other diagnostic questions as Q_j, then Q_i's correctness on specific students matches that of Q_j. Below are the plot and outputs of item based KNN accuracy.

k	Validation accuracy
1	0.6071
5	0.6542
11	0.6826
16	0.6860
21	0.6922
26	0.6903

The best k in this case is 21, with validation accuracy of 0.6922; test accuracy of 0.6816

(d) User based KNN is slightly better.

User based test accuracy: 0.6842

Item based test accuracy: 0.6818

(e) Potential limitations of KNN:

• The dimension of data is large, which is approximately 3 times of number of the data points. Also the quality of the data set is bad, i.e., the data set contains many missing value. Since KNN is sensitive to irrelevant features, this is probably not the ideal task to use KNN.

• The distance measure is problematic. We probably need to take the subject of a question into account, but the question metadata can not be used in this case.

• The data set is imbalanced: the number of correct questions is approximate 1.5 times the number of incorrect. This may cause KNN to make problematic predictions.

• KNN is slow with reasonably large data set

Q2 by Bojian Li

(a) The log-likelihood function is:

$$\begin{aligned} logp(C|\theta,\beta) &= log(\prod_i \prod_j C_{ij}\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)} + (1-C_{ij})(1-\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)})) \\ &= \sum_i \sum_j log(C_{ij}\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)} + (1-C_{ij})(1-\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)})) \end{aligned}$$

Note C_{ij} is 0, 1 or NaN.

$$\begin{aligned} \frac{\partial l}{\partial \theta_i} &= \frac{\partial}{\partial \theta_i}(\sum_i \sum_j log(C_{ij}\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)} + (1-C_{ij})(1-\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)}))) \\ &= \sum_j \frac{\partial}{\partial \theta_i}(C_{ij}log(\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)}) + (1-C_{ij})log(1-\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)})) \\ &= \sum_j C_{ij}\frac{exp(\theta_i)}{exp(\theta_i)+exp(\beta_j)} - (1-C_{ij})\frac{exp(\theta_i)}{exp(\theta_i)+exp(\beta_j)} \\ &= \sum_j (2C_{ij}-1)(1-\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)}) \end{aligned}$$

$$\frac{\partial l}{\partial \beta_j} = \sum_i (2C_{ij}-1)(\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)} - 1)$$

(Steps omitted for the derivative w.r.t β_j)

(b) The hyper-parameters are:

learning rate = 0.01

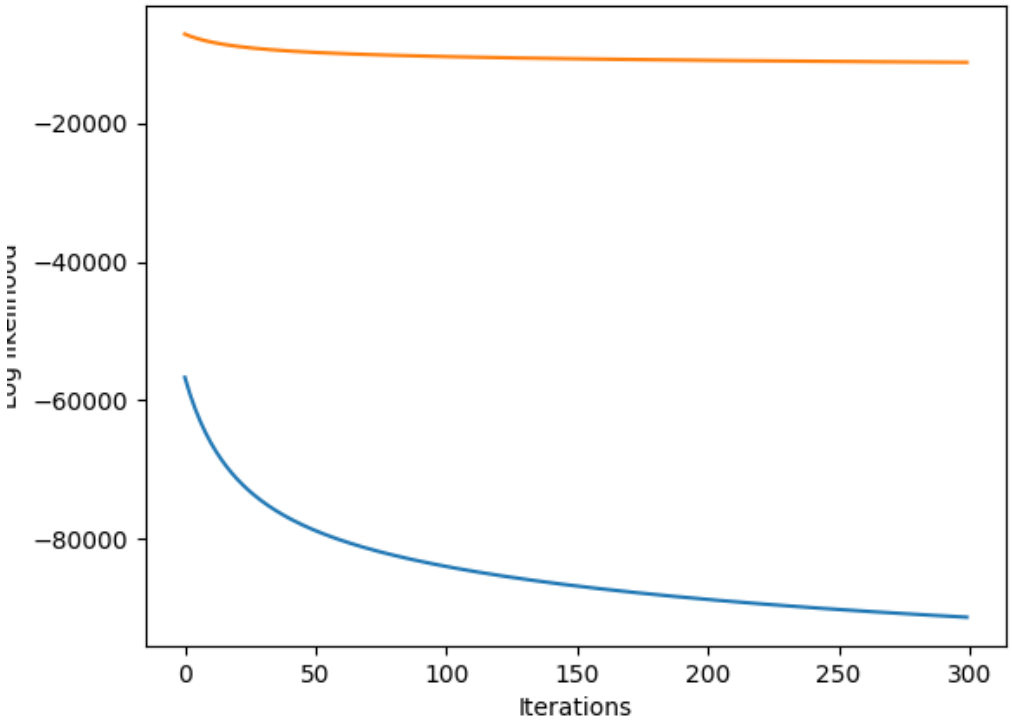
iterations = 300

As demonstrated in the graph, with 0.01 learning rate the IRT model converges at a reasonable rate and larger learning rate can lead to unwanted extreme values

from the sigmoid function.

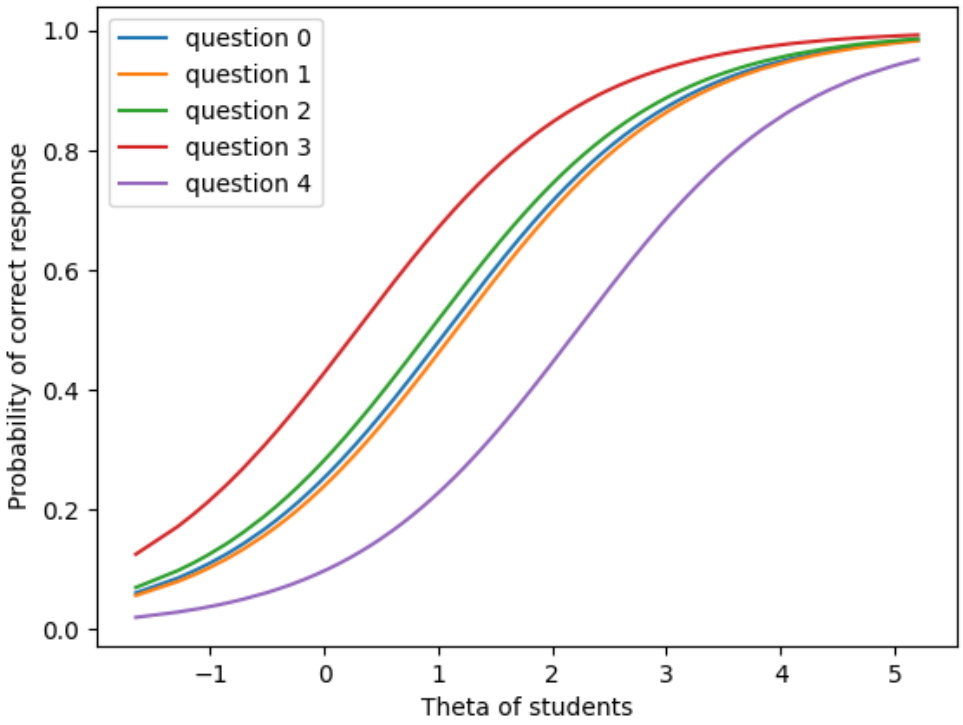
blue - training set

orange - validation set



(c) The final validation accuracy is 0.7064634490544736 and the final test accuracy is 0.7092859158904883.

(d) The graph represents an Item Characteristic Curve which are used to describe the relationship between the ability, defined on an ability scale, and each item.



Q3 by Amir Azimi

(a) The k-value with the best validation score was a k-value of 5.

The validation accuracy for a k-value of 5 was: 0.6590460

The test accuracy for a k-value of 5 was: 0.66356195

(b) Note that for SVD we replace the missing values with the mean of the columns of the data. This means that missing values are replaced by the mean for the question. Thus one limitation of this approach is it doesn't account for individual student performance.

For example, assume there was a student that answered every question they attempted correctly. Assume there was also a question that slightly more than half

of the students answered incorrectly. In this case, replacing the missing values by the mean of the question would result in claiming that this student would have answered this question incorrectly as well. However, given that this student answered every question they attempted correctly, it is likely that they would also have answered this question correctly had they attempted it, however the way we replace the values does not account for this.

(c) The code will be provided in separately.

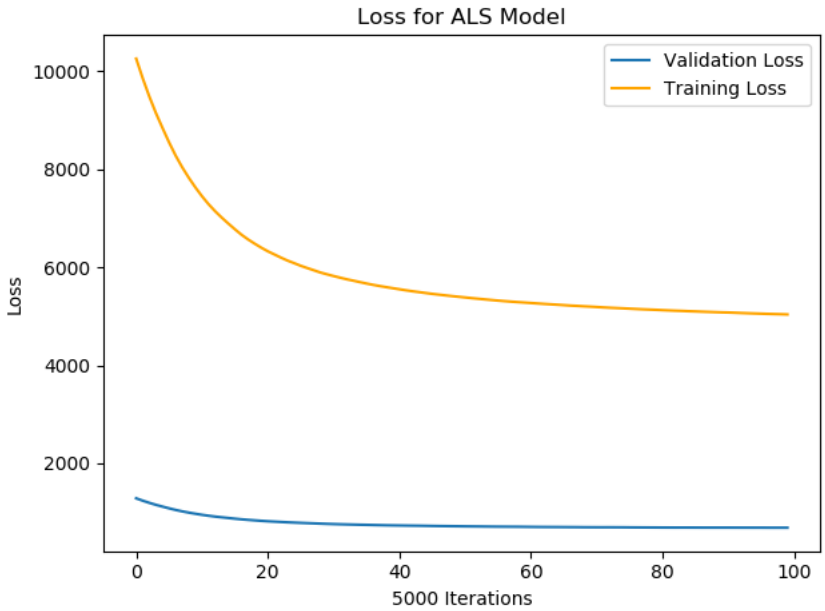
(d) The hyper-parameters that performed best were:

A learning rate of 0.01.

500 000 iterations.

The k* value with the highest validation accuracy was a k*-value of 30.

(e) The required plot for a k*-value of 30:



Final training accuracy for k*: 0.7398214789726221

Final validation accuracy for k*: 0.705193338978267

Final test accuracy for k*: 0.7036409822184589

(f) To train the model as a binary classification problem, I would change the loss function to a logistic cross-entropy loss. Thus the loss function would be of the form: $\mathcal{L}_{CE=} - C_{nm}\log(y) - (1 - C_{nm})\log(1 - y)$, where y would be our model's prediction. Then, like ALS, we would perform stochastic gradient descent to minimize the loss.

Q4 by Yuning Wu

In this question, we select three base models. We performed bootstrapping to train the models. The number of bootstrap and size of each bootstrap are the same for all the three models, where number of bootstrap $n = 32$, size of each bootstrap $N' = N$ where N is the size of the original data-set. Note that we performed bootstrapping to each model separately, so each model is expected to train on different data sets. Final prediction from each model is represented by a sparse matrix, which is the average of the prediction from each bootstrap. Then, we take the average prediction from the three model to get the final matrix, then use the given function sparse_evaluate() to get validation and test accuracy.

We tried numbers of combination of models and different settings of hyper-parameters. Here we will present two of them.

(i) For this one, we leave the hyper-parameter settings exactly same as the models we identified as the best from previous sections.

Model	Hyper-parameters
Matrix factorization by ALS	$k = 2$, $\alpha = 0.01$, # iterations = 500000
Matrix factorization by ALS	$k = 2$, $\alpha = 0.01$, # iterations = 500000
IRT	$\alpha = 0.01$, # iterations = 500000

The final validation accuracy is 0.707734

The test accuracy is 0.7077028

(ii) We then tried models with hyper-parameters settings different from previous questions. This time we ensemble three same models, where all three models are Matrix factorization by ALS, with $k = 20$, $\alpha = 0.03$, # iterations = 500000.

The final validation accuracy is 0.712250

The test accuracy is 0.7075924

So we conclude that bootstrap bagging and ensemble does not really improve accuracy, if we compare the results to the IRT model we identified in Q2, which has a better test accuracy. Perhaps we did not use the best hyper-parameter setting. Also, we know bagging is used to reduce variance and overfitting. We consider that overfitting is probably not a big problem in this context. Also, perhaps different models generate similar predictions, thus ensemble need not to improve accuracy.

Part B

Q1 by Amir Azimi

We chose to perform our modification on the ALS model in question 3(i).

One of the limitations of this ALS model is the fact that the model only captures the interactions that occur between a student's response and the question the answered (Koren et al., 2009). Thus the model is trained upon the assumption that a student's answer being correct or incorrect is directly reliant on how students interacted with other questions, i.e whether they got either questions right, or whether a question was answered correctly by other students.

However, there are other factors that play a role in determining whether a student

can answer a question correctly:

First is how "difficult" a question tends to be relative to other questions, which we call the question bias and denote b_m.

Second is how well a student tends to perform on questions relative to how other students performed , which we call the student bias and denote b_n.

The intuition behind this modification is that we expect some questions to be harder or easier than others, and some students to perform better or worse than other students.

In order to capture these new biases, we introduce the three new terms μ, b_n, b_m to our model, where μ is the global average of our data. Therefore we assume that the correctness of a student's answer, r_{nm}, is given by the equation:

$$r_{nm} = \mu + b_n + b_m + u_n^\top z_m$$

Thus our loss function becomes:

$$\min_{\boldsymbol{U},\boldsymbol{Z},\boldsymbol{B}} \frac{1}{2} \sum_{(n,m)\in O} (C_{nm} - \mu - b_n - b_m - \boldsymbol{u}_n^\top \boldsymbol{z}_m)^2 + \lambda(b_n^2 + b_m^2) \text{ (Hug, 2015)},$$

where the $\lambda(b_n^2 + b_m^2)$ is intended to regularize the bias terms.

Thus the update rules using the stochastic method in question 3(i) are:

$$u_n \leftarrow u_n + \alpha(C_{nm} - \mu - b_n - b_m - \boldsymbol{u}_n^\top \boldsymbol{z}_m) z_m$$

$$z_m \leftarrow z_m + \alpha(C_{nm} - \mu - b_n - b_m - \boldsymbol{u}_n^\top \boldsymbol{z}_m) u_n$$

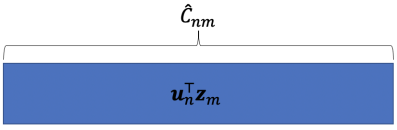
$$b_n \leftarrow b_n + \alpha(C_{nm} - \mu - b_n - b_m - \boldsymbol{u}_n^\top \boldsymbol{z}_m - \lambda b_n)$$

$$b_m \leftarrow b_m + \alpha(C_{nm} - \mu - b_n - b_m - \boldsymbol{u}_n^\top \boldsymbol{z}_m - \lambda b_m) \text{ (Hug, 2015)}$$

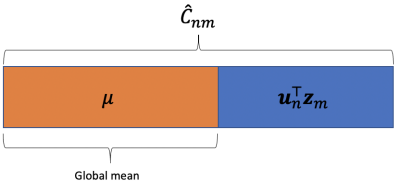
We expect that this modification should help the model better capture the relationships between questions and students which would help by both reducing the loss, and by allowing the model to generalize with higher accuracy. We also expect that this modification will allow the model to converge faster than it would without the bias terms.

Q2 by Yuning Wu

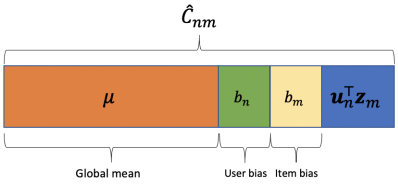
$$\min_{U,Z} \frac{1}{2} \sum_{(n,m)\in O} (C_{nm} - \boldsymbol{u}_n^\top \boldsymbol{z}_m)^2$$



$$\min_{U,Z} \frac{1}{2} \sum_{(n,m)\in O} (C_{nm} - \mu - \boldsymbol{u}_n^\top \boldsymbol{z}_m)^2$$



$$\min_{U,Z,B} \frac{1}{2} \sum_{(n,m)\in O} (C_{nm} - \mu - b_n - b_m - \boldsymbol{u}_n^\top \boldsymbol{z}_m)^2$$



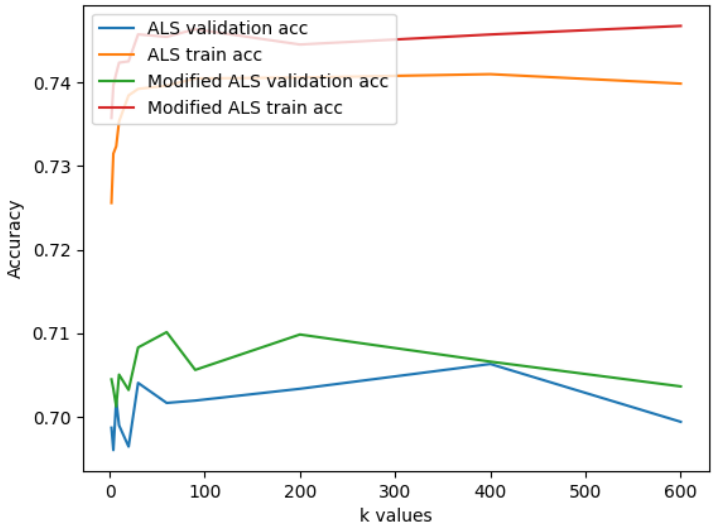
Regularized loss:

$$\min_{U,Z,B} \frac{1}{2} \sum_{(n,m)\in O} (C_{nm} - \mu - b_n - b_m - \boldsymbol{u}_n^\top \boldsymbol{z}_m)^2 + \lambda(b_n^2 + b_m^2)$$

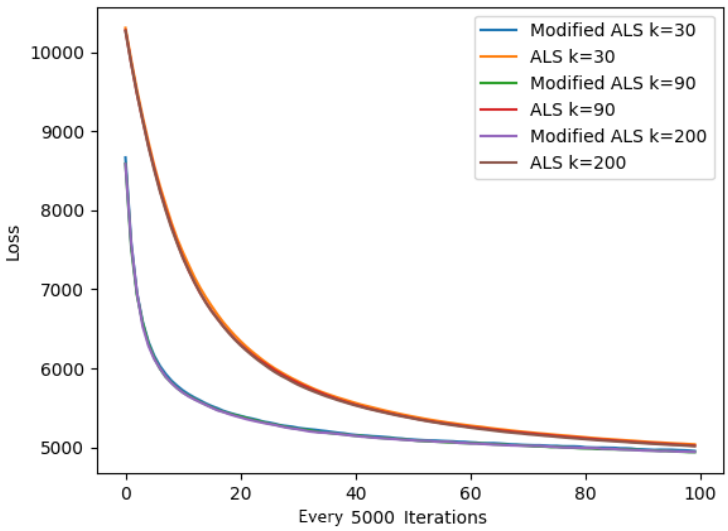
Q3 by Bojian Li

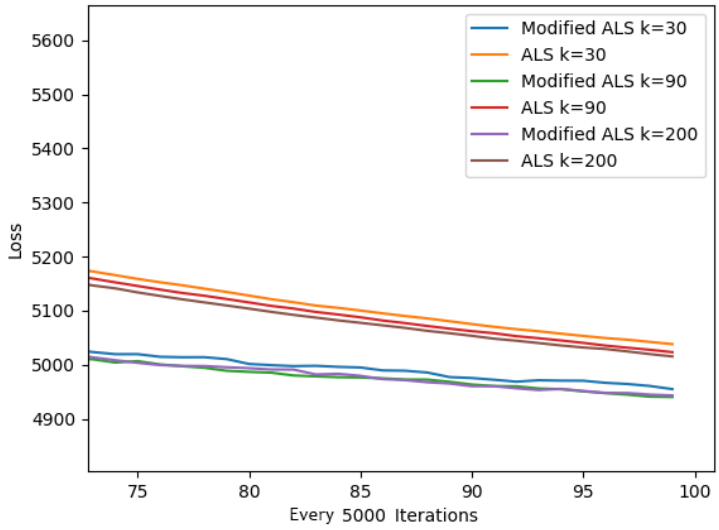
As explained in Q1 with additional bias term (i.e. more complexity)to the matrix factorization model we expect it to perform better for less underfitting problem. To demonstrate that by solely adding the bias term without regularization, we could get better accuracy for training set which represents reduced underfitting, we plotted the

average accuracy over multiple runs for different K-values with and without the bias term(fixed iteration and learning rate). It's shown that despite the test accuracy goes down as larger k values cause overfitting, the modified model consistently performs better than the original model.

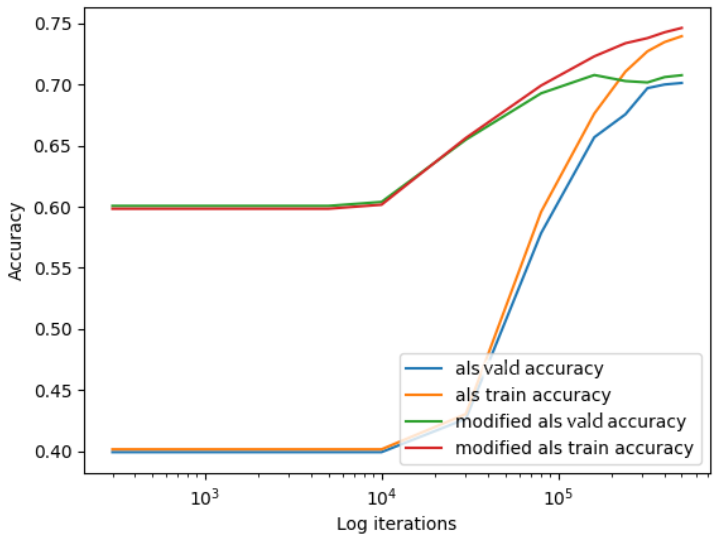


The modified ALS model is also expected to conduct a faster convergence rate due to reduced loss. To test this, we plotted the following graph with multiple k values for the entropy as a function of iterations.

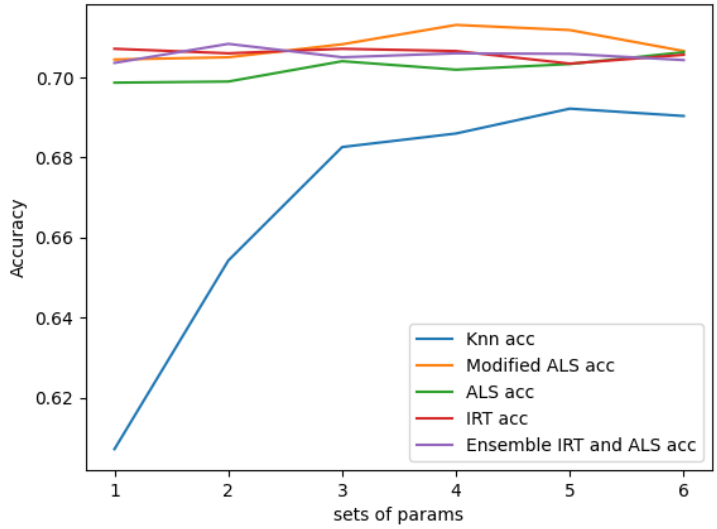


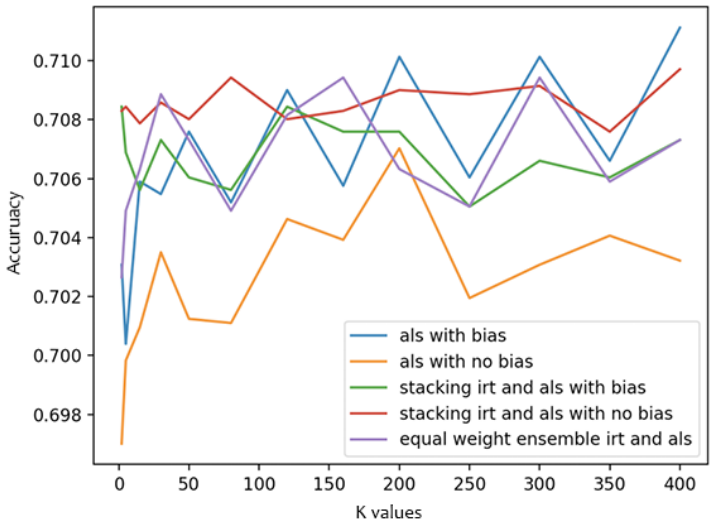


To demonstrate it more straightforward, we also tested the performance of the original and the modified model as a function of the iterations with a fixed k.



Also, while we chose ALS to perform our modification hoping to predict the answers with a higher accuracy, noticing that the ALS model was not the best performing model among the baseline models we tested in part A). We got IRT model and ensemble with both about 0.71 accuracy rate and the ALS matrix factorization obtained a 0.69 accuracy rate for the test set. However, the ALS model had the highest training accuracy overall so we could expect the test accuracy to be closer to the validation accuracy with regularization. Comparison of the final results with base models over different sets of parameters(except for the two ALS model where they run over same parameters and the results are average over multiple runs) including the ones with best results are plotted in the following graph together with a detailed comparison between different modifications of ALS over same k values.





Q4 by Yuning Wu

It is known that matrix factorization based algorithms are usually prone to overfitting. With the introduction of regularization term, overfitting is reduced. However, in such a cold-start context, i.e., the data matrix is very sparse, our model is not guaranteed to correct modelling a new user. For example, the sparse matrix global mean μ could have

large bias compare to the true mean of the population. On the hand, because the loss function need not to be convex, we could still get trapped in a local extremum; with the usage of stochastic gradient decent, this problem could be partially reduced. Note that the in such way the algorithm involves some randomness, one may need to train the model several times to get a satisfied result, and it is not guaranteed to be optimal. In contrast, the SVD algorithm should always give an optimal and stable result, but it generally leads to a worse accuracy.

Overall, our modified alternating least square algorithm with bias and regularization is competitive. Although the algorithm is not the most stable one, it outperforms KNN and SVD in this context, and has a considerable improvement compare to the original ALS algorithm in terms of validation accuracy. Some possible extension could be made in the future are

- Bayesian/Probabilistic Matrix Factorization: Assume an conditional Gaussian distribution on observed sample, and Gaussian priors on Q and P, the user and item latent feature vectors. This method also has many variation and extension, which could be an interesting topic to investigate in the future. The limitation of this approach is obvious, which is it makes strong assumption on data.
- Using cross-validation to train the model, it may or may not improve prediction

References

(1) Hug, N. (2015). Matrix Factorization-based algorithms. https://surprise .readthedocs.io/en/stable/matrix_factorization.html.

(2) Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. https://datajobs.com/data-science-repo/Recommender-Systems-%5BNetflix%5D.pdf.