CSC311 Final Project Report

Parinita Edke and Choy Kwan Kiu

December 16, 2020

1 Part A

1.1 k-Nearest Neighbor

a. User-based collaborative filtering: Plotted accuracy on the validation data as a function of k.

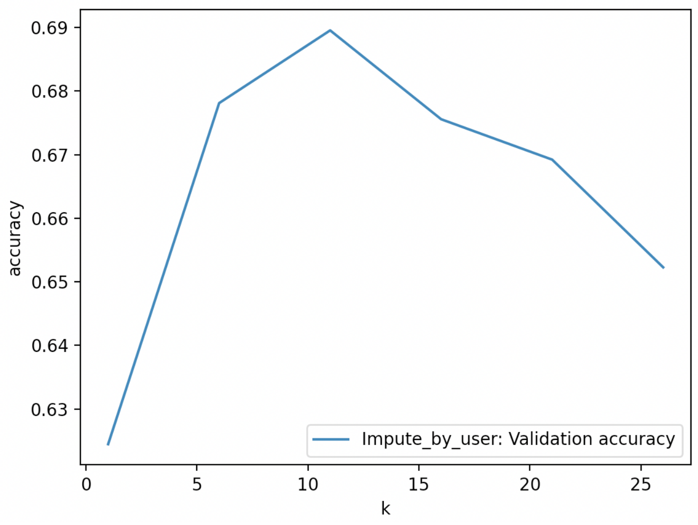


Figure 1: Validation accuracies vs k

The accuracies on the validation data for each of the k are as follows:

k-value	1	6	11	16	21	26
Accuracy	0.6244708	0.6780976	0.6895286	0.6755574	0.6692069	0.6522721

b. $k^* = 11$
Test accuracy for $k^* = 0.6412645$

c. Item-based collaborative filtering: Plotted accuracy on the validation data as a function of k.

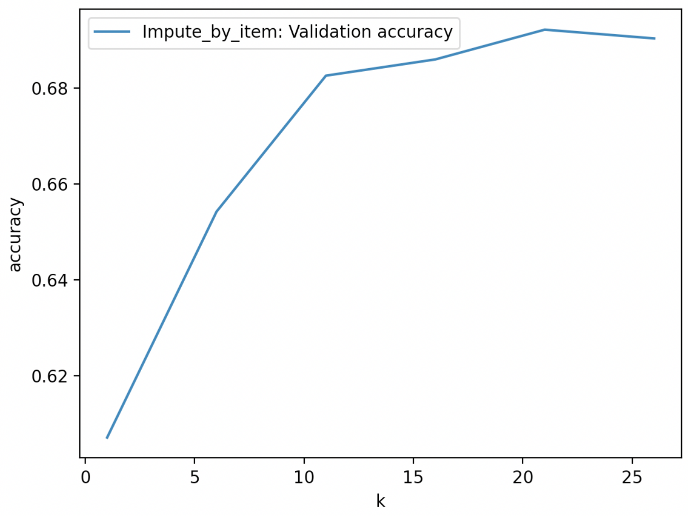


Figure 2: Validation accuracies vs k

The item-based collaborative filtering approach makes an assumption that questions with similar difficulty will be answered similarly by students. The accuracies on the validation data for each of the k are as follows:

k-value	1	6	11	16	21	26
Accuracy	0.6071126	0.6542478	0.6826136	0.6860006	0.69221	0.6903754

Item-based collaborative filtering $k^* = 21$ Test accuracy for $k^* = 0.6435224$

d. We achieved a test accuracy of 0.6412645 for the user-based collaborative filtering approach and a test accuracy of 0.6435224 for the item based collaborative filtering approach. The difference between the accuracy is 0.0022579. Since, the difference is fairly small, we can say that the test performance for the user-based and item-based collaborative filtering were quite similar, and that it would not matter which approach we choose to go for.

e. i. Just like what we've learnt in class, KNN does not work well with a high number of dimensions. In both approaches, we are dealing with high number of dimensions: 1774 in the user-based collaborative filtering approach and 542 for the item-based collaborative filtering approach.

   ii. If we are using the user-based collaborative approach, it is possible to have this bias towards a popular question (question with high number of interactions), and newly added question with 0 interactions would rarely be recommended by our system. Moreover, as the number of users and questions increases, the sparse matrix will waste a lot of memory space.

1.2 Item Response Theory

a. Let i represents the which student we are at, and j represent which question we are at.

$$
\begin{aligned}
p(C|\theta,\beta) &= \prod_{i=1}^{542}\prod_{j=1}^{1774} p\big(c_{ij}=1|\theta_i,\beta_j\big)^{c_{ij}}\Big(1-p\big(c_{ij}=1|\theta_i,\beta_j\big)\Big)^{1-c_{ij}} \\
&= \prod_{i=1}^{542}\prod_{j=1}^{1774}\left(\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)}\right)^{c_{ij}}\left(1-\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)}\right)^{1-c_{ij}} \\
log\big(p(C|\theta,\beta)\big) &= log\left(\prod_{i=1}^{542}\prod_{j=1}^{1774}\left(\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)}\right)^{c_{ij}}\left(1-\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)}\right)^{1-c_{ij}}\right) \\
&= \sum_{i=1}^{542}\sum_{j=1}^{1774} log\left(\left(\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)}\right)^{c_{ij}}\left(1-\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)}\right)^{1-c_{ij}}\right) \\
&= \sum_{i=1}^{542}\sum_{j=1}^{1774} c_{ij}\,log\left(\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)}\right)+(1-c_{ij})\,log\left(1-\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)}\right) \\
&= \sum_{i=1}^{542}\sum_{j=1}^{1774} c_{ij}(\theta_i-\beta_j)-c_{ij}\,log\big(1+exp(\theta_i-\beta_j)\big)-(1-c_{ij})\,log\big(1+exp(\theta_i-\beta_j)\big) \\
&= \sum_{i=1}^{542}\sum_{j=1}^{1774} c_{ij}(\theta_i-\beta_j)-log\big(1+exp(\theta_i-\beta_j)\big) \\
\frac{\partial log\big(p(C|\theta,\beta)\big)}{\partial\theta_i} &= \sum_{j=1}^{1774} c_{ij}-\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)} \\
\frac{\partial log\big(p(C|\theta,\beta)\big)}{\partial\beta_j} &= \sum_{i=1}^{542} -\left(c_{ij}-\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)}\right)
\end{aligned}
$$

b. We have tried out our model with different learning rates and number of iterations. We tested the following learning rates of 0.001, 0.005, 0.01, 0.05, 0.1, 0.5. We also tested the following number of iterations of 10, 50, 100, 250, 500, 1000. After training our model with different learning rate values and number of iterations, we found that our models gave the best validation accuracy score when the learning rate is 0.01 and when the iteration is 10. The plot below (Figure 3) shows the training and validation curve that our model produces with our chosen hyper-parameters.

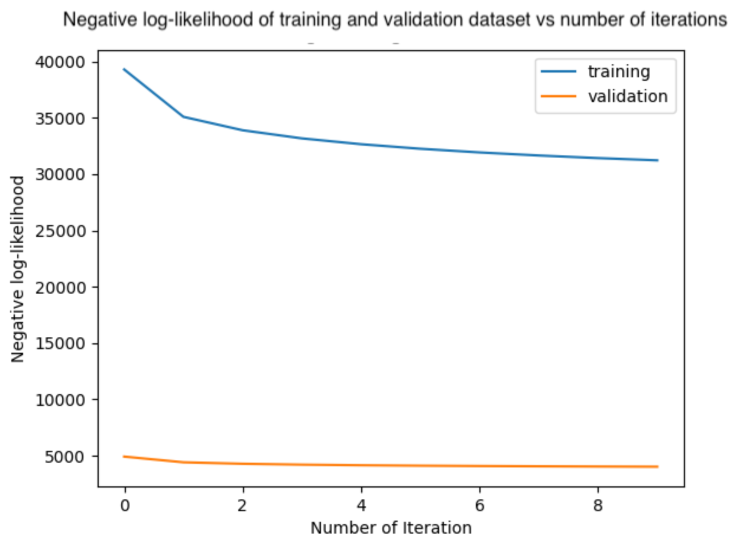


Figure 3: Training and validation log-likelihoods as a function of iteration

c. With the chosen hyper-parameters, the final validation and testing accuracies achieved were:

   (a) Validation Accuracy: 0.70716907

   (b) Final Testing Accuracy: 0.69968953

d. The shape of the curve in the plot below (Figure 4) resembles a sigmoidal curve, where the $p(c_{ij})$ increases with θ for each question; this means that as the students' ability increases, the probability of answering the questions we picked will also increase. Moreover, note that the curves for the different questions do not intersect, and that they have the same sigmoidal shape. The different questions simply differ in difficulty (i.e. their location parameter). The plot shows that the easier questions are the curves stacked on top (i.e. blue, yellow, red, green, purple), as they have a higher $p(c_{ij})$ for each value of theta.

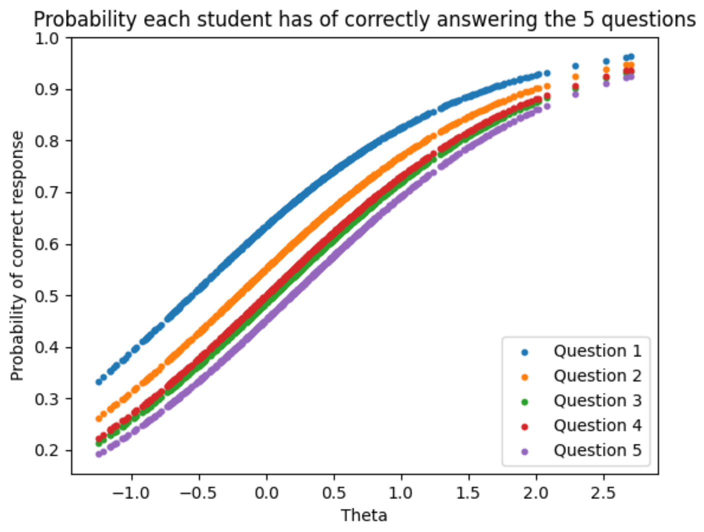


Figure 4: Probability of the correct response $p(c_{ij})$ as a function of θ given a question j

1.3 Matrix Factorization

a. We tested the *svd_reconstruct*() function with the following 9 values of k: 10, 15, 20, 25, 30, 35, 40, 45, and 50. We found that we achieved the highest validation accuracy when $k = 25$. The validation accuracy and the test accuracy when $k = 25$ was 0.6594694 and 0.6556590 respectively.

b. Since SVD is only defined for a complete matrix, we have to fill in the missing values before actually applying the matrix decomposition. There can be different ways to fill in missing values, and using the mean to fill that in (the starter code uses this method) is just one of them. Using different ways to fill in the missing values can yield different results. In our case, if we use the mean to fill in the missing values, it might lead to an underestimation of our standard errors and might cause significant changes to our co-variance matrix.

d. We tried the following combinations of hyper-parameters before we picked the best ones:

(a) $k \in \{10, 15, 20, 25, 30, 35, 40, 45, 50\}$

(b) learning rates $\in \{0.001, 0.005, 0.01, 0.05, 0.1\}$

(c) $num\,iterations \in \{80000, 128000, 150000\}$

We concluded that the hyper-parameters that gave us the best results were: $k = 40$, $lr = 0.05$ and $num\_iterations = 128000$. The validation accuracy we obtained was 0.6998. The validation accuracy we obtained for $num\,iteration = 150000$ was 0.7012 and it was higher than the validation accuracy we achieved with $num\_iterations = 128000$. However, considering the run-time of our algorithm and the small difference between the two validation accuracies, we chose $num\_iterations = 128000$ instead.

e. With our chosen $k^* = 40$, the plot below shows the training and validation squared-error-losses change as a function of iteration.

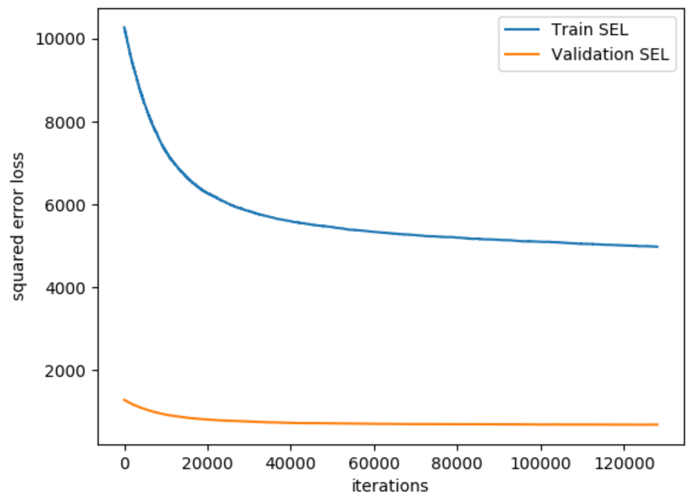


Figure 5: Training and validation squared-error-losses change as a function of iteration.

The final validation and test accuracies we achieved were:

(a) Validation Accuracy: 0.6971493

(b) Final Test Accuracy: 0.703641

f. In scenarios like this, we need a threshold r. With z being the result of our model, we then compare z and r. If z is less than r, then the result will be classified as one class, whereas if z is more than or equal to r, then the result will be classified to be another class.

1.4 1.4 Ensemble

1. We used 3 IRT models in our ensemble. Each model had the same hyper-parameters; we set the $learning\_rate = 0.01$ and $number\_of\_iterations = 500$. The final validation and test accuracies of our bagged model were:

   (a) Final Validation Accuracy: 0.7073102

   (b) Final Testing Accuracy: 0.7030765

   We kept our hyper-parameters to be the same in all 3 models as they gave us the best accuracy for each individual model. Hence, when we take the average of the best accuracies for each model, it ensures that our bagged model will have the best accuracy too.

   We did not observe an increase in accuracy using our bagged model but instead, we observed that we started to get more consistent and stable results. Since we average the predictions of our 3 IRT models, any fluctuations that arise due to the differences in the re-sampled data set are evened out and we achieve a more consistent set of results.

2 Part B

1. For Part B, we decided that we would extend the IRT model that we implemented in Part A so that we can further optimize and improve our accuracy. In part A, we implemented the 1-parameter IRT model. The training, validation and testing accuracies for the 1-parameter IRT model were 0.7276, 0.7072, and 0.6997 respectively. The training accuracy obtained from the 1-parameter IRT model is on the lower end, hence suggesting that our model is likely underfitting. In the 1-parameter IRT model, we based our prediction of whether a student will get an question correct simply on the student's ability and the question's difficulty. However, there may also be other confounding variables that are going to affect the prediction of whether a student will get a particular question correct; this might be the reason why there is a high bias in our model. An example of such a variable could be the subject topics that the question tests a student on. The more advanced a subject topic is, the more the question would discriminate between students with varying levels of ability. Therefore, in this part, we are going to use the question metadata and the 2-parameter IRT model to account for the subject topics that each question tests the students on.

   Part 1: How do we treat our new data?
   The csv file *question_meta.csv* consists of 2 columns: a question id and a list of subject topics that the question tests a student on. Since it is hard to use a list of subjects in our 2-parameter IRT model, we decided that we would divide questions into different levels of discriminability. Based on Khan Academy [1] and *subject_meta.csv*, we categorized each subject topic into 3 groups based on the grade the topic is first taught to students. We placed the subject topics taught in grades 1 - 5 into Group 0, the subject topics taught in grades 6 - 10 into Group 1, and the subject topics taught in grade 11 and beyond into the Group 2.

   After assigning groups to all the 388 subject topics, we set our third parameter k for question q to be the following:

$$k = \frac{\Sigma\,\text{group number of subject topics of question}\,q}{2 * (\text{no of subject topics of question}\,q)}$$

   Part 2: Our Model
   Let u_i, q_i, c_i, k_q, represent the indices of user, question, correctness and the discriminability of the question of the i^{th} entry in the dictionary, (i.e: $u_i = \text{dict}[''\mathit{user\_id}''][i]$; $q_i = \text{dict}[''\mathit{question\_id}''][i]$; $c_i = \text{dict}[''\mathit{is\_correct}''][i]$ etc). Let N be the number of entries we have in the training data.
   For the 2-parameter IRT model, the probability that question j is correctly answered by student i is formulated as:

$$p(c_{ij} = 1|\theta_i\beta_j) = sigmoid(k_j * (\theta_i - \beta_j))$$

   Similar to the IRT, we also want to calculate the $P(C|\theta, \beta)$.

$$\begin{aligned} P(C|\theta,\beta) &= \prod_{i=1}^{542}\prod_{j=1}^{1774} P(c_{ij}=1|\theta_i,\beta_j)^{c_{ij}}(1-P(c_{ij}=1|\theta_i,\beta_j)^{1-c_{ij}} \\ &= \prod_{i=1}^{542}\prod_{j=1}^{1774}\left(\frac{exp\big(k_j(\theta_i-\beta_j)\big)}{1+exp\big(k_j(\theta_i-\beta_j)\big)}\right)^{c_{ij}}\left(1-\frac{exp\big(k_j(\theta_i-\beta_j)\big)}{1+exp\big(k_j(\theta_i-\beta_j)\big)}\right)^{1-c_{ij}} \end{aligned}$$

   We obtain the following derivations after a series of computation:

$$\frac{\partial log\, p(C|\theta,\beta)}{\partial\theta_i} = \sum_{j=1}^{1774} \frac{k_j * c_{ij} - k_j(exp(k_j(\theta_i-\beta_j)))(1-c_{ij})}{1+exp(k_j(\theta_i-\beta_j))}$$

$$\frac{\partial log\, p(C|\theta,\beta)}{\partial\beta_j} = \sum_{i=1}^{542} \frac{-k_j * c_{ij} + k_j(exp(k_j(\theta_i-\beta_j)))(1-c_{ij})}{1+exp(k_j(\theta_i-\beta_j))}$$

$$\frac{\partial log\, p(C|\theta,\beta)}{\partial k_j} = \sum_{i=1}^{542}(\theta_i - \beta_j) * \left(c_{ij} - \frac{exp(k_j(\theta_i-\beta_j))}{1+exp(k_j(\theta_i-\beta_j))}\right)$$

Algorithm 1 Updating θ and β and k

```
0: Initialize $\theta$ and $\beta$ to be zeros
0: for each j in iterations do
0:     for each i in N do
0:         # where $k_{q_i}$ is the subject rank of the question at the $i^{th}$ entry
0:         $\theta_i \leftarrow \theta_i + lr * \frac{k_{q_i} * c_i - (k_{q_i} * exp(k*(\theta_i-\beta_i)) * (1-c_i))}{1+exp(k_{q_i} * (\theta_i-\beta_i))}$
0:         $\beta_i \leftarrow \beta_i + lr * -(\frac{k_{q_i} * c_i - (k_{q_i} * exp(k_{q_i} * (\theta_i-\beta_i)) * (1-c_i))}{1+exp(k_{q_i} * (\theta_i-\beta_i))})$
0:         $k_{q_i} \leftarrow k_{q_i} + lr * (\frac{c_i * (\theta_i-\beta_i) - (1-c_i) * (exp(k_{q_i} * (\theta_i-\beta_i)) * \theta_i - exp(k_i * (\theta_i-\beta_i)) * \beta_i)}{1+exp(k_{q_i} * (\theta_i-\beta_i))})$
0:     end for
0: end for=0
```

After completing our 2-parameter IRT model, we created an ensemble by averaging predictions from 10 2-parameter IRT models. Then we evaluated the average predictions with the validation data set to get the validation accuracy.

2. The two figures below show a scatter plot of the probability each student has of correctly answering 5 questions and a diagram illustrating the overarching idea of how our model works.

   The scatter plot we have below (Figure 6) differs a lot from that of question 2. We can see that for this plot below, the curves for the 5 questions all have different slopes and intersect with each other; in Figure 4 in question 2, the curves for the 5 questions have the same slopes and do not intersect at all. This intersection of curves is caused by the discrimination parameter k that was introduced in the 2-parameter IRT model; the discriminability of a question represents how crucial it is to understand the topic the question is testing the student on in order to get the correct answer. The parameter k influences the slope of the curve. Figure 7 shows how we set our discrimination parameter, the flow of the updates, and the ensemble of 10 2-parameter IRT models.

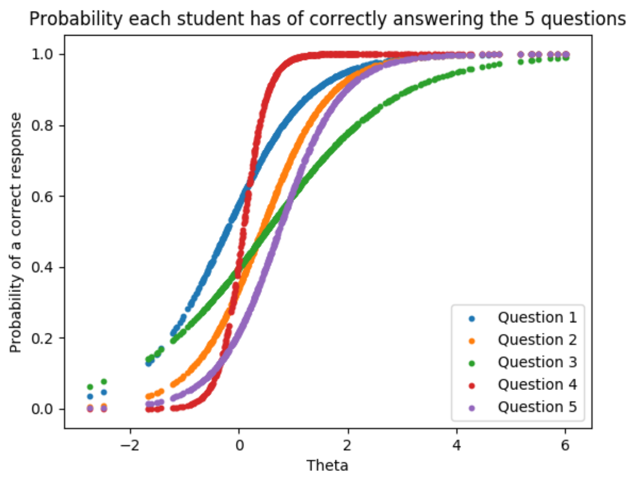


Figure 6: Probability distribution of our model

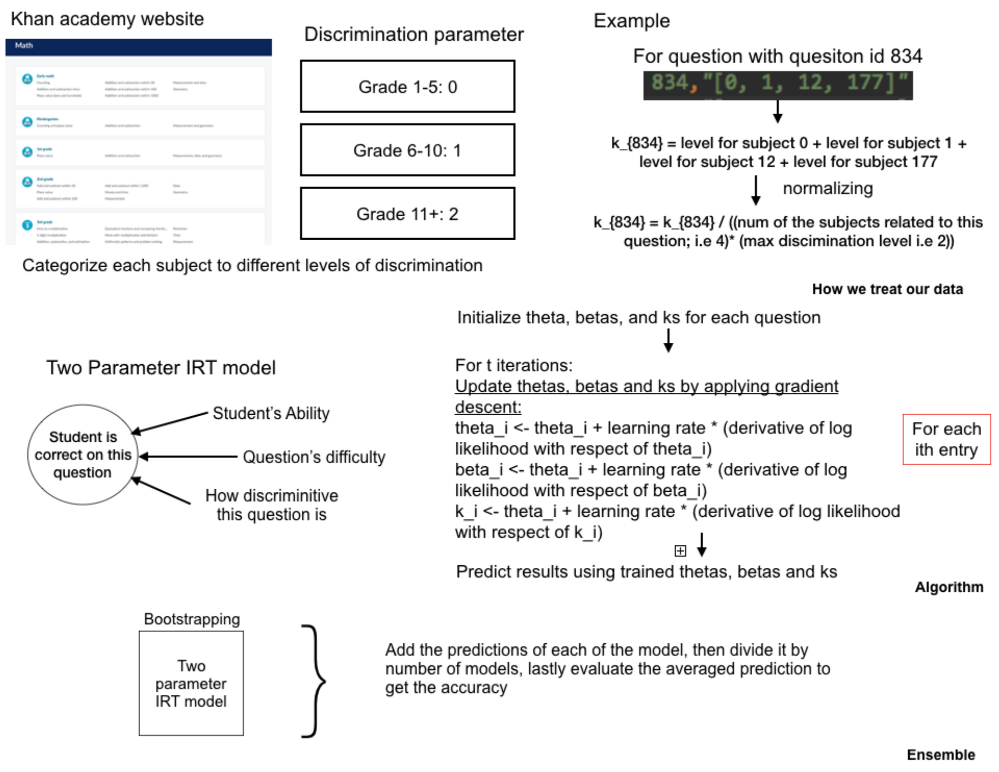
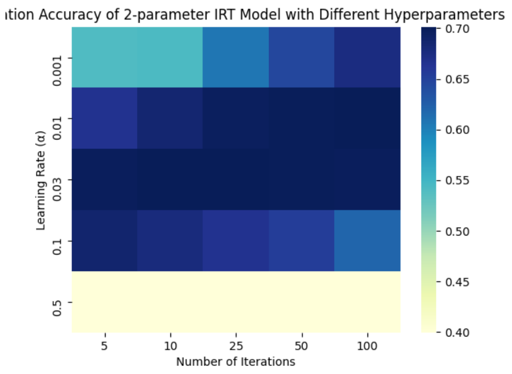

Khan academy website
Discrimination parameter
Grade 1-5: 0
Grade 6-10: 1
Grade 11+: 2
Categorize each subject to different levels of discrimination
Example
For question with quesiton id 834
834,"[0, 1, 12, 177]"
k_{834} = level for subject 0 + level for subject 1 + level for subject 12 + level for subject 177
normalizing
k_{834} = k_{834} / ((num of the subjects related to this question; i.e 4)* (max discimination level i.e 2))
How we treat our data
Two Parameter IRT model
Student is correct on this question
Student's Ability
Question's difficulty
How discriminitive this question is
Initialize theta, betas, and ks for each question
For t iterations:
Update thetas, betas and ks by applying gradient descent:
theta_i <- theta_i + learning rate * (derivative of log likelihood with respect of theta_i)
beta_i <- theta_i + learning rate * (derivative of log likelihood with respect of beta_i)
k_i <- theta_i + learning rate * (derivative of log likelihood with respect of k_i)
For each ith entry
Predict results using trained thetas, betas and ks
Algorithm
Bootstrapping
Two parameter IRT model
Add the predictions of each of the model, then divide it by number of models, lastly evaluate the averaged prediction to get the accuracy
Ensemble

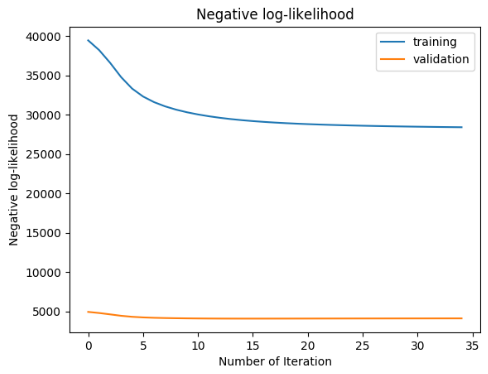
Figure 7: Description of our model

3. **Our Hypothesis:** Given more information about the complexity of the question, we should be able to make a better estimate on whether or not a student is going to answer a question correctly.

   We first determined what the best hyper-parameter choices were for the 2-parameter IRT model. The grid map below shows the validation accuracies for different pairs of hyper-parameters. The negative log likelihood plot shows that the rate of decrease is higher in the beginning and the converging negative log likelihood is smaller than what we have in Figure 3 in question 2. It also takes more iterations to converge.



(a) Figure 8: Validation accuracy for different hyper-parameters using our 2 parameter IRT model



(b) Figure 9: Negative log likelihood for our 2 parameter IRT model

We picked the learning rate to be 0.03. For the number of iterations, since the trade off in accuracy and time between 10 and 25 iterations is not significant, we picked 10 iterations so that our model could run faster for the ensemble.

Next, we ran our 1-parameter and 2-parameter IRT models and an ensemble of 10 1-parameter IRT models and an ensemble of 10 2-parameter IRT models. The table below shows the accuracy of the training, validation and testing data sets of the models tested:

Model	Learning rate	Iterations	Train acc.	Validation acc.	Test acc.
1-param IRT	0.01	10	0.7276	0.7072	0.6997
1-param IRT ensemble	0.01	500	0.7370	0.7073	0.7031
2-param IRT	0.03	25	0.7423	0.7005	0.7012
2-param IRT ensemble	0.03	10	0.7386	0.7029	0.7052

Table 1: Accuracies of training,validation and testing data sets of the models

As observed from Table 1, the 2-parameter IRT model and its corresponding ensemble both achieve a greater training accuracy. This is important as it suggests a decrease in our bias from the 1-parameter IRT model. In addition, the testing accuracy for the 2-parameter IRT model and its corresponding ensemble also increases, suggesting that our model is better in generalizing than our 1-parameter IRT model. However, we also observed that there is a drop in our validation accuracy from the 1-parameter IRT model to the 2-parameter IRT models. This might be because we are now considering one more parameter in our new model, which increases our variance.

4. Currently, our model does not perform a lot better than our model from Part A. This may be due to the violation of assumptions that we have to make prior to using an IRT model [2]. The first key assumption is that there is a single latent trait within the model (i.e. unidimensionality). This might not be satisfied as there can be other latent traits that can affect the probability of the student getting a question correct; an example of such a trait is how wealthy a student is. A student's financial situation can definitely play a role in how much time and resources they can allocate towards their education, thereby affecting their ability to answer a question correctly.

   The second key assumption is local independence, which assumes that a student's response to one question is not going to affect the student's response to other questions. This means that only the student's ability and the discriminability and the difficulty of that particular question will be considered when determining if a student will answer the question correctly. This would be an issue as the data we have might not satisfy this assumption. There are many questions that are related to each other as they all test similar concepts. In this case, it is possible that a student's response to question A is going to affect their response to question B and so on. As a result, whenever the this local independence assumption is violated, regardless of how we train our model, any statistical analysis on our model would indicate a high bias.

   Therefore, to improve our model accuracy, we have to account for local dependence and the possibility of having more than one latent variable. In terms of considering more than one latent variable, we can use a multidimensional IRT model. Moreover, we can also introduce regularization to penalize unwanted complexity so as to avoid overfitting. To solve the local dependence issue, we might want to add a new variable to specify whether a question is related to another question, and then try fitting our model with questions that are not related to other questions.

3 Contributions of each team member

1. Parinita Edke
    (a) Primarily worked on coding and writing up Part A Questions 1, 4
    (b) Implemented Part B modified algorithm
    (c) Worked on testing Part B modified algorithm
    (d) Proofread the report
2. Choy Kwan Kiu
    (a) Primarily worked on coding and writing up Part A Questions 2, 3
    (b) Implemented Part B modified algorithm
    (c) Worked on analysis on Part B modified algorithm

4 References

[1] Khan academy, https://www.khanacademy.org/math

[2] Michigan State University, https://msu.edu/~dwong/StudentWorkArchive/CEP900F04-RDP/Mihn-ItemResponseTheory.htm