CSC311 Final Project

Huifeng Wu, Qien Song

December 2020

1 Part A

1.1 k-Nearest Neighbor

(a) We have run kNN for different values of $k \in \{1, 6, 11, 16, 21, 26\}$. We report the accuracy on the validation data as a function of k in Figure 1.

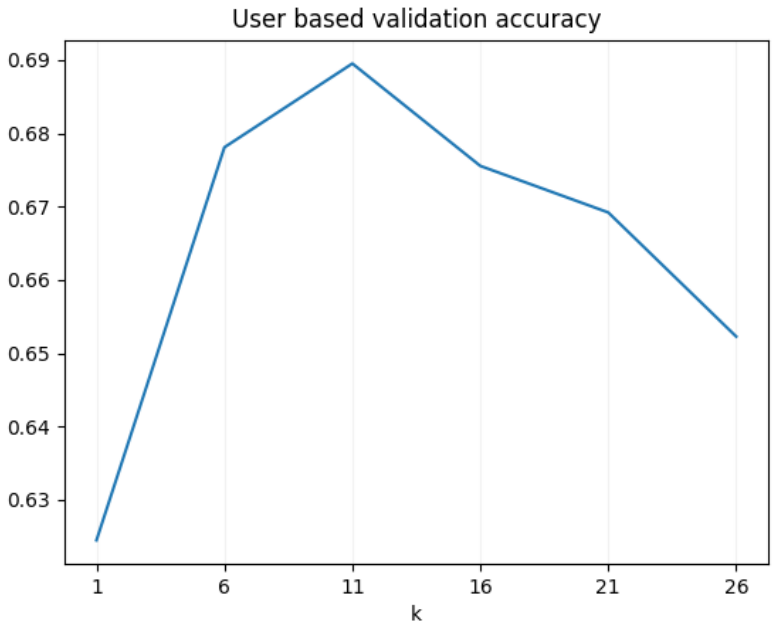


Figure 1: User-based validation accuracy

(b) $k^* = 11$ has the highest performance on validation data at 68.95% accuracy. The final test accuracy with the chosen k^* is 68.42%.

(c) Our underlying assumption on item-based collaborative filtering is if the answers by certain users to question X match those of question Y, then X's answer correctness corresponding to specific users matches that of question Y. The optimal $k^* = 21$ in this case, and the validation accuracy with the chosen k^* is 69.22%. The final test accuracy is 68.16%. The accuracy on the validation data is shown in Figure 2.

(d) Repeating questions (a) and (b) with item-based collaborative filtering, we report its accuracy on the validation data as a function of k in Figure 2, where the optimal $k^* = 21$ in this case, and the validation accuracy with the chosen k^* is 69.22%. The final test accuracy is 68.16%. Therefore, item-based performs

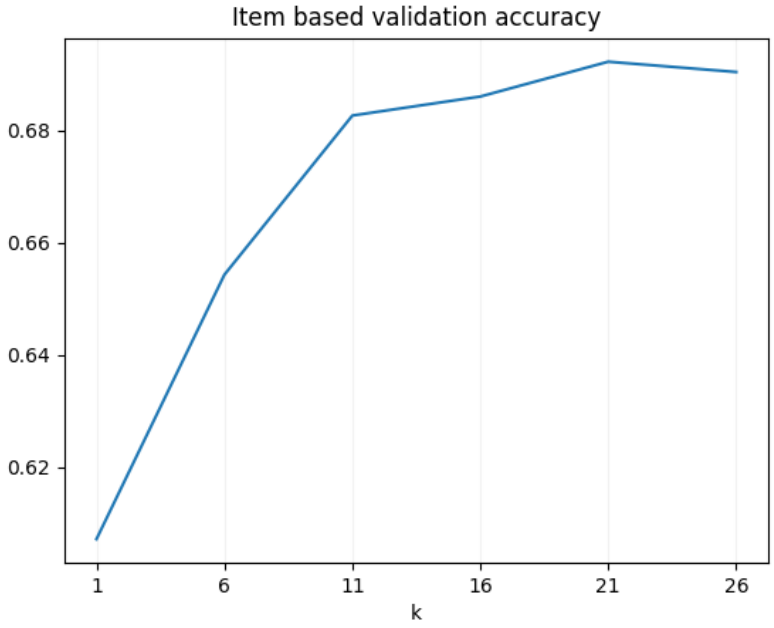


Figure 2: Item-based validation accuracy

slightly better than user-based on validation dataset, and slightly worse on test dataset. The difference in performance is insignificant.

(e) We list the following potential limitations of kNN for the task given:

- The KNN model cannot be parametrized, meaning that the entire training dataset has to be stored. It is also harder to quantify and analyze relationships between questions and students.
- Expensive computation at test time as for each test sample where a distance calculation and sorting needs to be performed. The algorithm may not be scalable on larger question banks.
- kNN only finds the closest user that similarly answered other questions or the closest question being answered by similar users, to predict the correctness. Embedded with Nan-Euclidean algorithm, the given kNN rather finds students or questions that share similar ability and difficulty, respectively. However, it does not truly consider if student A has the same answer distribution on other diagnostic questions as student B.

1.2 Item Response Theory

(a) Assume C is a $N \times M$ sparse matrix, where N denotes the number of students, M denotes the number of questions. Here we derive the log-likelihood log p(C| θ, β) for all students and questions, as follow:

$$
\begin{aligned}
p\left(c_{ij}=1 \mid \theta_i, \beta_j\right) &= \frac{\exp\left(\theta_i-\beta_j\right)}{1+\exp\left(\theta_i-\beta_j\right)} \\
p(C|\theta,\beta) &= \prod_{(i,j)} \left[\frac{\exp\left(\theta_i-\beta_j\right)}{1+\exp\left(\theta_i-\beta_j\right)}\right]^{c_{ij}} \left[\frac{1}{1+\exp\left(\theta_i-\beta_j\right)}\right]^{1-c_{ij}} \\
\log\ p(C|\theta,\beta) &= \sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij}\ \Big[(\theta_i-\beta_j) - \log(1+\exp(\theta_i-\beta_j))\Big] - (1-c_{ij})\log(1+\exp(\theta_i-\beta_j)) \\
&= \sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij}(\theta_i-\beta_j) - \log(1+\exp(\theta_i-\beta_j)) \\
&= \sum_{i=1}^{N}\sum_{j=1}^{M} \mathbb{1}(c_{ij}\in\{0,1\})\ \Big[c_{ij}(\theta_i-\beta_j) - \log(1+\exp(\theta_i-\beta_j))\Big]
\end{aligned}
$$

Besides, we let M_i be the number of questions user i answered correctly, whereas N_j is the number of users that answer question j correctly. We show the derivative of the log-likelihood with respect to θ_i and β_j, as follow:

$$
\begin{aligned}
\frac{\partial\ \log\ p(C|\theta,\beta)}{\partial\theta_i} &= \sum_{j=1}^{M} \mathbb{1}(c_{ij}\in\{0,1\})\ \left[c_{ij} - \frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)}\right] \\
&= M_i - \sum_{j=1}^{M} \frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)}\mathbb{1}(c_{ij}\in\{0,1\}) \\
\frac{\partial\ \log\ p(C|\theta,\beta)}{\partial\beta_j} &= \sum_{i=1}^{N} \mathbb{1}(c_{ij}\in\{0,1\})\ \left[-c_{ij} + \frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)}\right] \\
&= -N_j + \sum_{i=1}^{N} \frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)}\mathbb{1}(c_{ij}\in\{0,1\})
\end{aligned}
$$

(b) We implemented missing functions in `item_response.py` that performs alternating gradient descent on θ and β to maximize the log-likelihood. The hyperparameters we selected are learning rate = 0.1 and 20 iterations. With such configuration, we report the training curve that shows the training and validation log-likelihoods as a function of iteration. In early iterations, both training and validation loss decrease rapidly. Later on, training loss gradually decreases, but validation loss seems very consistent.

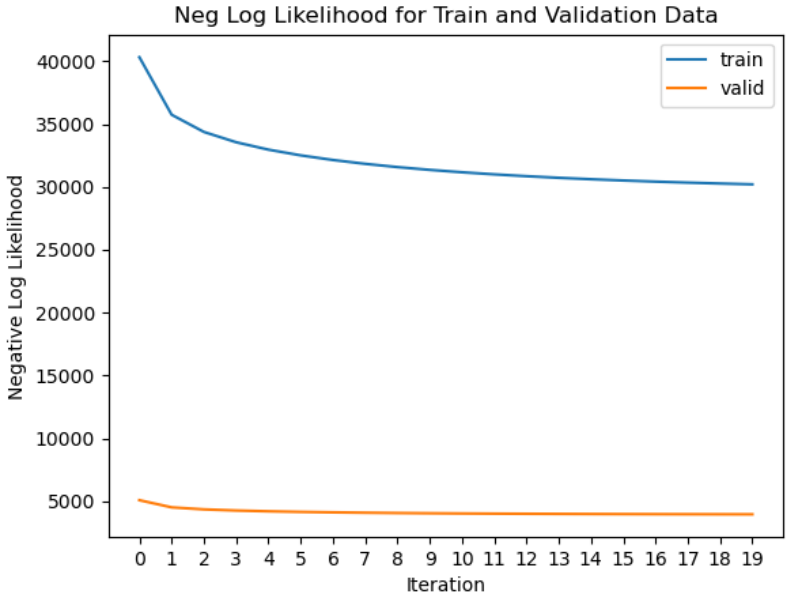


Figure 3: Neg Log-Likelihood

(c) Here we report the final validation and test accuracies are 70.66% and 70.33%, respectively.

(d) Figure 4 shows the probability of each student answering the selected questions correctly. The x-axis θ represents student's ability and the y-axis represents the probability of answering the question correctly. All of the 5 curves show that the probability increases as student's ability increases, which is natural that student ability has a positive correlation with solving the question correctly.

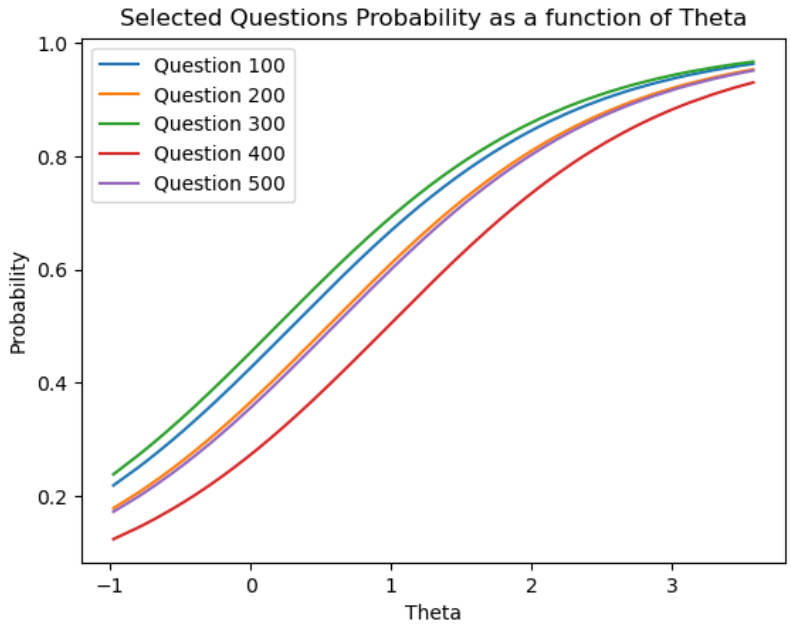


Figure 4: Student ability vs answering correctly

1.3 Neural Networks

(a) We describe some differences between Alternating Least Squares (ALS) and neural networks, as follow:

- ALS is generally regarded as an algorithm to solve an unsupervised learning task that generates predictions based on matrix decomposition. Neural networks, even autoencoders in $Q3$, treats the problem as a supervised learning task. For conventional neural networks, there is always a target (label). For the autoencoder, the input vector itself is the label.
- An alternative least square algorithm always decomposes the sparse matrix to two matrices. A neural network is much more flexible as the user can decide the number of hidden layers and the cardinality of each hidden layer. A neural network can be used for classification and segmentation tasks etc.
- Neural networks commonly use gradient descent to optimize a non-convex objective, and they do not have closed form solutions. In comparison, ALS also optimize a non-convex objective but it decomposes the objective to two alternating convex objectives with closed form solutions.

(b) We implemented a class `AutoEncoder` that performs a forward pass of the autoencoder following the instructions in the docstring.

(c) We trained the autoencoder using latent dimensions of $k \in \{10, 50, 100, 200, 500\}$. Also, we tuned optimization hyperparameters epoch $\in \{5, 20, 40\}$ and learning rate $\in \{0.001, 0.01, 0.1\}$. As shown in Figure 5, we tried different combinations of these hyperparameters and eventually set learning rate to 0.1 and the number of iterations to 40. With these hyperparameters, we obtained that $k^* = 50$ has the highest validation accuracy.

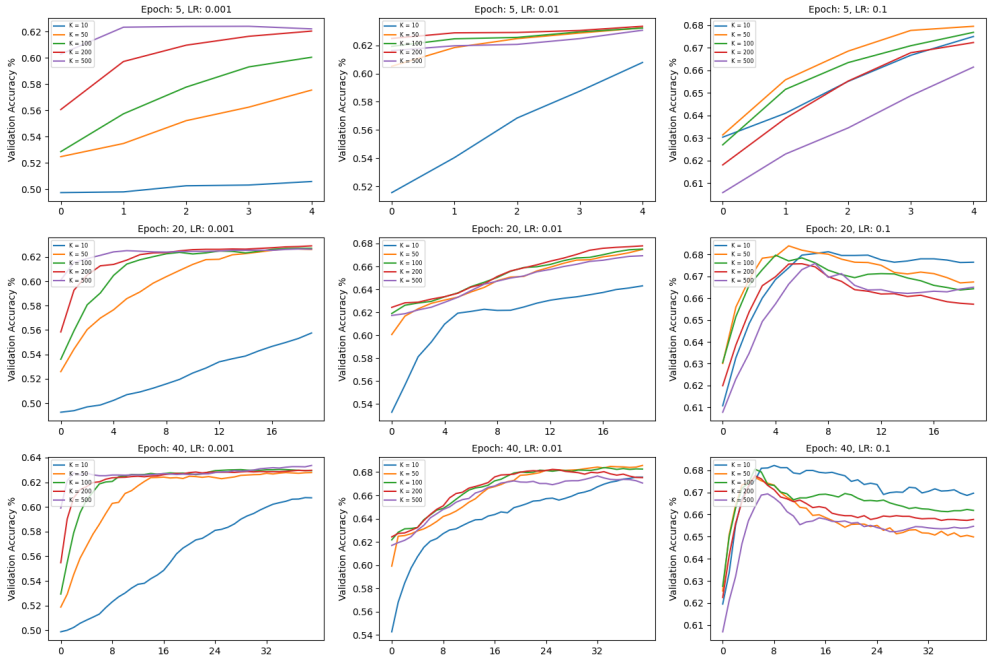


Figure 5: Tuning hyperparameters and latent dimension

(d) With the chosen $k^* = 50$, we report how the training and validation objectives changes as a function of epoch. As shown in Figure 6, training loss consistently decrease over each iteration. Validation accuracy initially increases rapidly in early epochs and gradually plateaus after the 25-th iteration. The final validation accuracy is 68.80%. The final test accuracy is 68.78%.

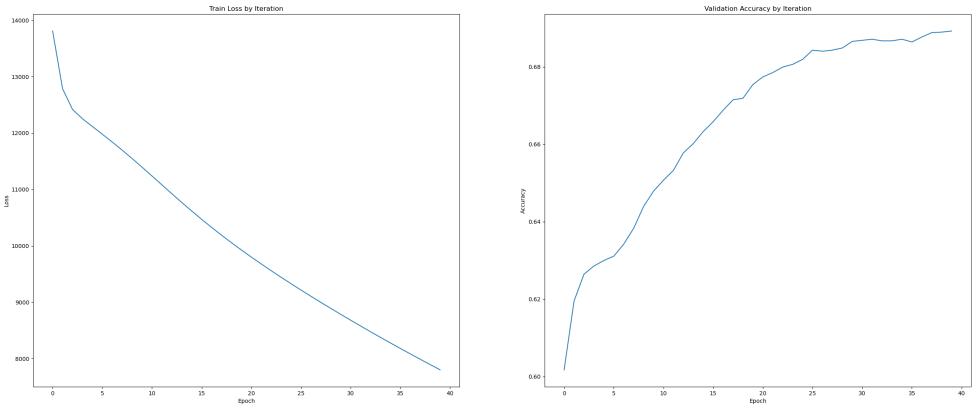


Figure 6: Train Loss and validation accuracy by number of epochs

(e) We added the regularization term to the objective function. Using the chosen k^* and other hyperparameters selected from part (d), we tuned the regularization penalty $\lambda \in \{0.001, 0.01, 0.1, 1\}$. The best λ is 0.1. The final validation accuracy is 69.15% and test accuracy is 68.22%. The model performs slightly better in validation but slightly worse in test. As shown in Figure 7, except $\lambda = 1$, regularization does not lead to a significant change in validation accuracy.

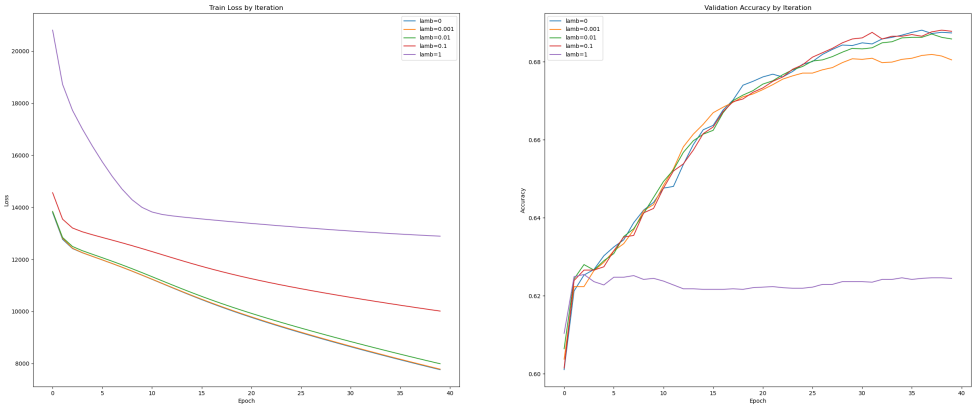


Figure 7: Tuning Regularization term

1.4 Ensemble

In this problem, we are implementing bagging ensemble to improve the stability and accuracy of the above base models. We selected item response theory as the base model and trained it by bootstrapping the training set.

The hyperparameters set-up we used for Item-Response is:

- learning rate = 0.01
- number of iterations = 20

Ensemble Process:

- Generate 3 resampled datasets using the corresponding sampling method[1].
- Train the base model (IRT) on each resampled dataset.
- As each model outputs a probability $\in [0, 1]$, we obtain the average of all 3 probabilities and use a threshold of 0.5 to make a binary prediction.
- We average the predictive correctness of each of the 3 base models on train, validation and test datasets.
- We calculate the predictive correctness of the ensemble model on train, validation and test datasets.

We attempted two sampling methods.

- Sample by student only: Out of all 542 students, we sample with replacement 542 students using a uniform distribution.
- Sample by student & question: Out of all 56688 students' answers, we sample with replacement 56688 answers using a uniform distribution.

Sampling Method	Model Type	Train	Validation	Test
Student & Question	Individual	74.79	69.82	69.71
Student & Question	Ensemble	73.36	70.41	70.08
Student Only	Individual	73.43	65.82	66.22
Student Only	Ensemble	71.40	68.13	68.78

Table 1: Model performance with different sampling methods

Analysis:
Sampling by student & question:
The ensemble model performs slightly better than the average of individual base models on validation and test set, and performs worse on the training set. The main objective of bagging ensemble is to reduce variance and this is evident here as performance on validation and test is improved. However, the improve in performance is relatively small.

Sampling by student only:
The ensemble model performs much better than the average of individual base models on validation and test set, and performs worse on training set. The reduction of overfitting is more evident using this sampling technique because when sampling by students only, bootstrapped datasets would miss certain students' data, which leads to no training of θ on specific students and a low validation accuracy. The ensemble model likely includes all students, which leads to $\sim 2.5\%$ improvement in performance.

Comparing with the results obtained in $Q2$, the ensemble method with student only sampling performs worse. This is possibly due to incomplete student data as the small number of resampling leads to underweight on certain students. To confirm this guess, we did train the base model again by resampling 20 times and the performance becomes about the same as $Q2$. With student & question sampling, the performance is nearly identical.

One reason that bagging does not improve performance here is that Item-Response Theory itself does not have an overfitting problem. When trained individually in $Q2$, the difference between training and validation accuracy is already small. Another reason is that the base model itself only has the explanatory power to predict with around 70% correctness, we would require more powerful base models to see an improvement in ensemble performance.

2 Part B

2.1 Introduction

Although online education services provide people opportunities of skill and knowledge development, it is challenging to diagnose if students fully understand concepts taught. Mostly, they are only given randomly sampled questions from a large database, as their assessment. Therefore, it is a common phenomenon in online assessment that students make careless errors or have lucky guesses. Ignoring such factors in predicting students' response may lead to high estimation bias and resulting in a low accuracy.

2.2 Method

In the existing implementation of the Item-Reponse Theory (IRT) model (which we refer to as the baseline model in this section), we have made an assumption that student's performance on a question is solely dependent on the difficulty of the question, β, and the student's ability, θ. Such representation may lead to underfitting and slow convergence, as it is an over-simplification of real life scenarios. The following three important factors are not included:

- Students make random guesses when facing difficult questions. In the baseline model, it is implicitly assumed that a student would score only by knowing how to answer a question, by which the probability of answering it correctly converge towards zero if the question is infinitely hard. However, students can either make a completely uneducated guess, or use elimination methods to narrow down to certain choices.
- It is also natural that students can make errors even with the ability to solve a question correctly. External factors such as distractions, low motivation or simply clumsiness can result in an incorrect answer.
- It is insufficient to measure question difficulty using a single parameter. There are two major latent dimensions in a question's difficulty: the amount of knowledge required to solve the question, the magnitude of increase in probability of correct response with respect to an increase in knowledge. The first dimension focuses on the possession of knowledge while the latter dimension focuses on critical thinking skills. The dimension of critical thinking can also be viewed as the slope between question difficulty and probability of correct response.

We believe that by incorporating these factors, we can build a more informative model that converges faster, and improves in accuracy. Accordingly, we introduce three additional parameters, and construct the following probability equation:

$$P(C_{ij} = 1|l, u, \alpha, \beta, \theta) = l_i + (u_i - l_i)\frac{\exp(\alpha_j(\theta_i - \beta_j))}{1 + \exp(\alpha_j(\theta_i - \beta_j))}$$

where l_i is the lower bound probability of answering correctly of the i-th student, u_i is the upper bound probability of answering correctly of the i-th student, α_j is the slope (critical thinking) of the j-th question, β_j is the difficulty (possession of knowledge) of the j-th question, and θ_i is the ability of the i-th student.

We use the same optimization algorithm and loss function as the base model, by minimizing the negative log-likelihood function using gradient descent (Derivative of each parameter in Appendix). The log-likelihood function is presented as follows:

$$p(C|l, u, \alpha, \beta, \theta) = \prod_{(i,j)} \left[l_i + (u_i - l_i)\frac{\exp(\alpha_j(\theta_i - \beta_j))}{1 + \exp(\alpha_j(\theta_i - \beta_j))}\right]^{c_{ij}} \left[1 - u_i + (u_i - l_i)\frac{1}{1 + \exp(\alpha_j(\theta_i - \beta_j))}\right]^{1-c_{ij}}$$

$$\log\ p(C|l, u, \alpha, \beta, \theta) = \sum_{i=1}^{N}\sum_{j=1}^{M} \mathbb{1}(c_{ij} \in \{0, 1\})$$
$$\left(c_{ij} log \left[l_i + (u_i - l_i)\frac{\exp(\alpha_j(\theta_i - \beta_j))}{1 + \exp(\alpha_j(\theta_i - \beta_j))})\right] + (1 - c_{ij}) log \left[1 - u_i + (u_i - l_i)\frac{1}{1 + \exp(\alpha_j(\theta_i - \beta_j))}\right]\right)$$

2.3 Results

2.3.1 Baselines Comparison

We report the results of the proposed model and compare it with the baselines in Part A in Table 2. The IRT-based models outperform other methods significantly. Comparing with the base model, the test accuracy of the new model is worse while the validation accuracy is better. Overall, the final performance between the two models is nearly identical.

Model	Iterations	Train NLLK	Validation Acc	Test Acc
Base (IRT)	20	30184.96	70.66%	70.33%
Proposed	10	32504.85	70.94%	69.54%
KNN	N/A	N/A	69.22%	68.42%
Neural Network	40	N/A	69.15%	68.78%
Ensemble	20	N/A	70.41%	70.08%

Table 2: Model comparison with baseline models whose best accuracies are selected.

The new model noticeably converges faster as shown in Figure 8. Training and validation loss starts to increase at the 10-th iteration for the new model. A possible explanation is that the extra parameters benefits the model from processing more information at each iteration, which leads to rapid convergence.

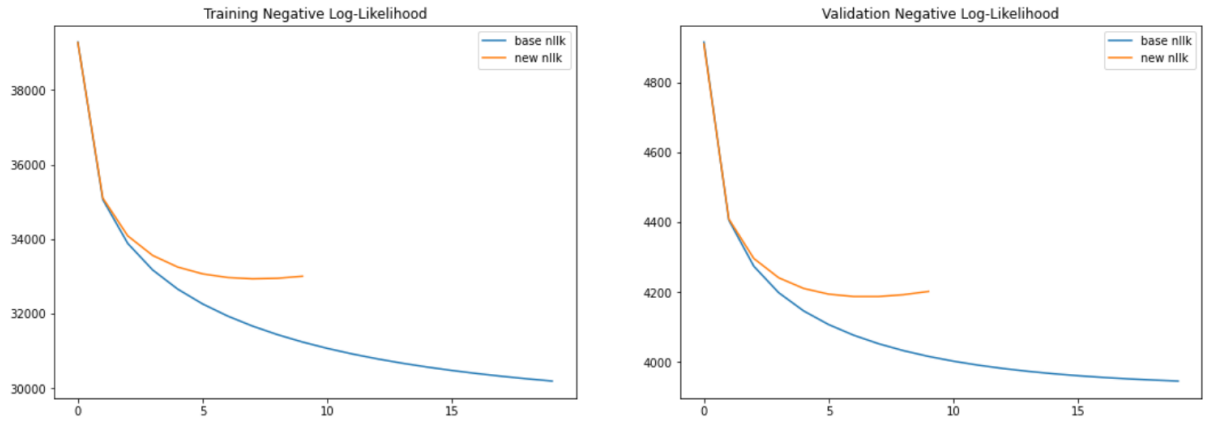
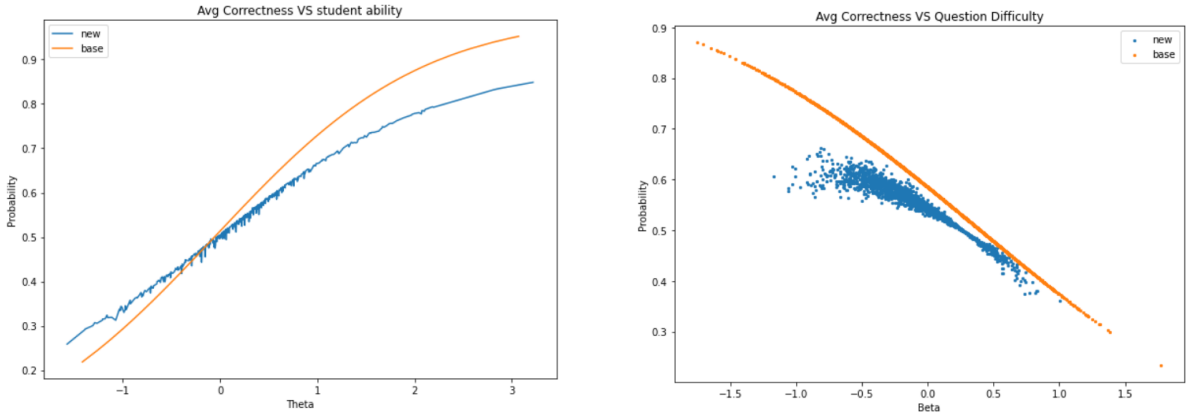


Figure 8: Base and proposed model train/validation loss

2.3.2 Parameter Comparison

In Figure 9(a), we show the average probability of correct response as a function of student ability θ across all questions. The result largely matches our model building objective, which is to include lower and upper bounds on probability estimation. For low θ values, the chance of answering a question correctly is increased from 20% to 25%. As demonstrated, we have successfully incorporated the aspect of random guessing to modelling responses. For high θ values, the probability curve decreases from 95% to 80%, which indicates an inclusion of factors such as carelessness and distractions.

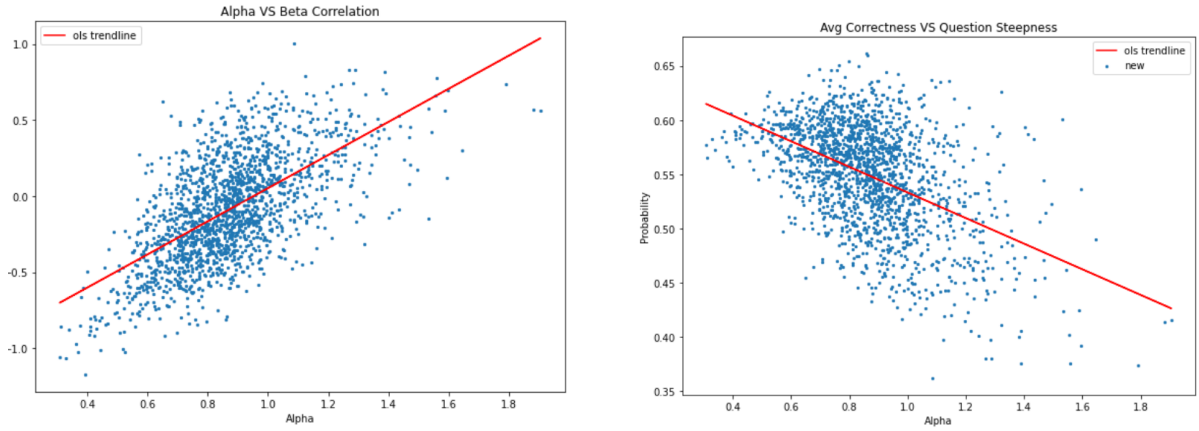
Similarly in Figure 9(b), we present the average probability of correct response as a function of question difficulty (β) across all students. For harder questions, the probability of correct response is increased from 30% to about 40%, while the upper end is decreased from 90% to 65%. The pattern demonstrates the effect of "guessing" given hard questions, and the effect that prevents better scores given easy questions.



(a) Student ability

(b) Question difficulty

Figure 9: Average probability of correct response



(a) Critical thinking (alpha) vs beta

(b) Critical thinking (alpha)

Figure 10: Critical thinking & possession of knowledge

While the new β is still negatively correlated with probability, we observe a more concentrated distribution of β towards 0. This is a result of introducing α in the new model, which reduces the explanatory power of β. In Figure 10(b), we observe that α is negatively correlated with probability. In Figure 10(a), we observe that α and β have a positive correlation, which matches the intuition that a question requiring more knowledge would also require more critical thinking. This also illustrates the two dimensions, that were originally encompassed within one parameter in the base model, can be modeled separately.

2.4 Experiments

The introduction of upper and lower bounds, and slope (critical thinking) is designed to particularly improve model robustness in extreme scenarios, such as questions with maximum and minimum difficulties, and students with maximum and minimum abilities. Therefore, we conducted an accuracy comparison with the tail section of the parameters to evaluate the effectiveness of the proposed model.

We examined the students with trained θ value below the 5-th and above the 95-th percentile, and questions with trained β value in the same tail region with the results in Table 3. The new model shows $\sim 2\%$ improvement on predictions of "easy questions", and $\sim 2\%$ deterioration on predictions of hard questions. This

shows that the inclusion of an upper bound parameter is an effective way to account for inadvertent errors in hard questions, while the lower bound is ineffective in capturing the probability of random guesses. In addition to question-wise comparison, there is $\sim 1.5\%$ improvement on predictions of "bad students", which shows that the lower bound increases the model's explanatory power. In comparison, the upper bound does not lead to a change in accuracy on "good student".

Overall, the analysis on the tail section of the data indicates that both lower and upper bound parameters improve a tail segment of the data. Upper bound is more effective as it does not lead to deterioration of a segment of the data.

Model	Percentile	Parameter	Characteristic	Validation Acc	Test Acc
Baseline	Below 5^{th}	β	Easy Question	80.93	82.32
New	Below 5^{th}	β	Easy Question	82.56	84.34
Baseline	Above 95^{th}	β	Hard Question	70.84	68.97
New	Above 95^{th}	β	Hard Question	70.02	65.52
Baseline	Below 5^{th}	θ	Bad Student	70.48	67.96
New	Below 5^{th}	θ	Bad Student	71.47	69.34
Baseline	Above 95^{th}	θ	Good Student	88.76	88.89
New	Above 95^{th}	θ	Good Student	88.76	88.89

Table 3: Tail distribution comparison with the base model. Note that blue font indicates improvement, red font indicates deterioration

2.5 Limitations and Improvements

2.5.1 Model Assumption

Although the proposed model yields competing results and hold highly interpretive parameters, our underlying assumption is naive. We simply assume each response is independent of another, which hardly is true in reality. For example, students may face an insolvable question and get extremely nervous afterwards, which affects their ability to answer other questions.

In this way, our assumption is compromised, since data from each user are either longitudinal or correlated. The result of maximum likelihood estimation (MLE), which is the optimization objective, is no longer valid. Here we suggest obtain more information on the students' sequence of response to verify the independence assumption, or we could attempt time series and Markov modelling.

2.5.2 Parameter Estimation

As highlighted in our experiment, We intend to increase accuracy of how students perform with hard questions by capturing "lucky" guesses. Nevertheless, the proposed model underperforms for above 95 percentile of β, compared to the base model. The inconsistency might be caused by the model overestimating the importance of the lower bound. It could also be caused by the complicated interaction effect after introducing α.

We can potentially set prior distribution for the parameters, and integrate the estimation with Bayesian inference. Moreover, we can construct a Bayes classifier and Maximum a-posteriori (MAP) estimation. It is particularly worth noting that these two methods are less sensitive to data sparsity. Although calculating a lower/upper bound for each student may be over-complicating the model, the average lower bound is roughly 20% or 25% for most students in real life, for instance.

2.5.3 Model Architecture

The added parameter α, slope (critical thinking), is correlated with β, difficulty (possession of knowledge), and the interaction effect between the two parameters may lead to a loss in accuracy. One possible extension

is to add an interaction term between β and α in the model. Simplify α to a binary indicator of whether the question requires critical thinking or not, or create a polynomial term to account for the interaction effect.

2.6 Conclusion

To summarize, we created a more complex model by parametrizing the aspect of random guessing, accidental response and critical thinking. The model not only sees a significant improvement in convergence speed but also achieves significant improvement on segments of the data that were over/underestimated by the baseline model. However, the overall accuracy across all segments of data sees minimal improvement. This 5-parameter IRT model can still be greatly improved by obtaining more relevant information on the data and finetuning modeling assumptions.

Author Contributions

Both authors Qien Song and Huifeng Wu closely participated in problem solving and model implementation for Part A. For part B, Qien Song contributed to literature research, model proposal and drafted the report. Huifeng Wu helped shape the overall experiment and report, as well as the analysis of the results.

References

Liao, W., Ho, R., Yen, Y., amp; Cheng, H. (2012). The Four-Parameter Logistic Item Response Theory Model As a Robust Method of Estimating Ability Despite Aberrant Responses. Social Behavior and Personality: An International Journal, 40(10), 1679-1694. doi:10.2224/sbp.2012.40.10.1679

3 Appendix

Derivatives of Model Parameters

* We will use k to denote $\alpha_j(\theta_i - \beta_j))$

$$\frac{\partial \ \log \ p(C|l,u,\alpha,\beta,\theta)}{\partial \theta_i} = \sum_{j=1}^{M} \mathbb{1}(c_{ij} \in \{0,1\})X$$

$$X = \begin{cases} -\frac{(u_i - l_i)\alpha_j exp(k)}{(1+exp(k))(l_i exp(k)+u_i)} & \text{if } c_{ij} = 0 \\ \frac{(u_i - l_i)\alpha_j exp(k)}{(1+exp(k))(u_i exp(k)+l_i)} & \text{if } c_{ij} = 1 \end{cases}$$

$$\frac{\partial \ \log \ p(C|l,u,\alpha,\beta,\theta)}{\partial \beta_j} = \sum_{i=1}^{N} \mathbb{1}(c_{ij} \in \{0,1\})Y$$

$$Y = \begin{cases} \frac{(u_i - l_i)\alpha_j exp(k)}{(1+exp(k))(l_i exp(k)+u_i)} & \text{if } c_{ij} = 0 \\ -\frac{(u_i - l_i)\alpha_j exp(k)}{(1+exp(k))(u_i exp(k)+l_i)} & \text{if } c_{ij} = 1 \end{cases}$$

$$\frac{\partial \ \log \ p(C|l,u,\alpha,\beta,\theta)}{\partial \alpha_j} = \sum_{i=1}^{N} \mathbb{1}(c_{ij} \in \{0,1\})Z$$

$$Z = \begin{cases} -\frac{(u_i - l_i)(\theta_i - \beta_j) exp(k)}{(1+exp(k))(l_i exp(k)+u_i)} & \text{if } c_{ij} = 0 \\ \frac{(u_i - l_i)(\theta_i - \beta_j) exp(k)}{(1+exp(k))(u_i exp(k)+l_i)} & \text{if } c_{ij} = 1 \end{cases}$$

$$\frac{\partial \ \log \ p(C|l,u,\alpha,\beta,\theta)}{\partial u_i} = \sum_{j=1}^{M} \mathbb{1}(c_{ij} \in \{0,1\})V$$

$$V = \begin{cases} -\frac{1}{l_i exp(k)+u_i} & \text{if } c_{ij} = 0 \\ -\frac{exp(k)}{u_i exp(k)+l_i} & \text{if } c_{ij} = 1 \end{cases}$$

$$\frac{\partial \ \log \ p(C|l,u,\alpha,\beta,\theta)}{\partial l_i} = \sum_{j=1}^{M} \mathbb{1}(c_{ij} \in \{0,1\})W$$

$$W = \begin{cases} \frac{exp(k)}{l_i exp(k)+u_i} & \text{if } c_{ij} = 0 \\ \frac{1}{u_i exp(k)+l_i} & \text{if } c_{ij} = 1 \end{cases}$$