CSC311 Final Project

Jiaqi Guo, Zezheng Gu, Shang Wu
December 3, 2021

Introduction

In this project, we are interested in assessing the quality of education provided by online education services. One of the methods that ensures students understand the course concepts is through an assessment composed of diagnostic questions; each is a multiple choice question with one correct answer and can reveal a common misconception in related subjects. Our aim is to build machine learning algorithms to predict whether a student can correctly answer the diagnostic questions based on the student's ability and the difficulty of questions.

The dataset is obtained from an online education platform called Eedi, which includes subsampled answers of 542 students to 1774 diagnostic questions from primary to high school. In part A, we will apply four machine learning algorithms, namely k-Nearest Neighbor, Item Response Theory, Matrix Factorization, and ensemble, to predict students' correctness of a given diagnostic question and compare their prediction accuracies. In part B, we will consider the limiting factors of performance of the algorithms and modify one of them to improve the prediction accuracy.

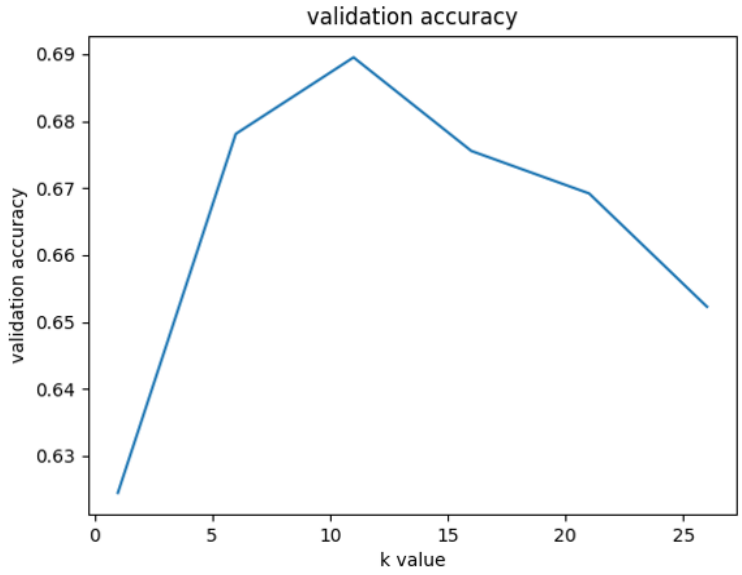
Part A

1. k-Nearest Neighbor

(a) The underlying assumption is that if student A has the same correct and incorrect answers on other diagnostic questions as student B, A's correctness on specific diagnostic questions matches that of student B.

Accuracy on the validation data as a function of given k is:
k: 1 Validation Accuracy: 0.6244707874682472
k: 6 Validation Accuracy: 0.6780976573525261
k: 11 Validation Accuracy: 0.6895286480383855
k: 16 Validation Accuracy: 0.6755574372001129
k: 21 Validation Accuracy: 0.6692068868190799
k: 26 Validation Accuracy: 0.6522720858029918

The resulting plot is



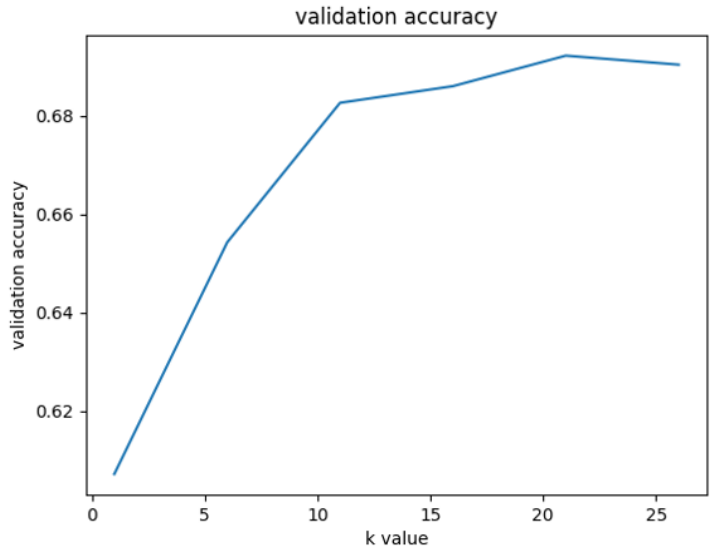
(b) $k^* = 11$ has the highest performance on validation data, which results in a 0.6895 final test accuracy.

(c) The underlying assumption of this scenario is that if question A is answered similarly by other students as question B, A's correctness on a specific student matches that of question B.

Accuracy on the validation data as a function of given k is:

k: 1 Validation Accuracy: 0.607112616426757
k: 6 Validation Accuracy: 0.6542478125882021
k: 11 Validation Accuracy: 0.6826136042901496
k: 16 Validation Accuracy: 0.6860005644933672
k: 21 Validation Accuracy: 0.6922099915325995
k: 26 Validation Accuracy: 0.69037538808919

The resulting plot is:



$k^* = 21$ has the highest performance on validation data, which results in a 0.6922 final test accuracy.

(d) Imputing by user-based filtering has an accuracy of 0.6842 on the testing set.
Imputing by item-based filtering has an accuracy of 0.6816 on the testing set.
We have a slightly better performance for user-based filtering.

(e) The first limitation is that the underlying assumptions we made for the filtering process does not apply in reality, because we cannot say students who have similar answers on a set of questions will definitely have the same answer for another specific question.
The second limitation is that this algorithm suffers from the curse of dimensionality, so most data points are approximately the same distance.

2. Item Response Theory

(a) The derivation of log-likelihood for all students and questions is:

$$p(c|\theta,\beta) = \Pi_{i,j}\left(\frac{e^{\theta_i-\beta_j}}{1+e^{\theta_i-\beta_j}}\right)^{c_{ij}}\left(\frac{1}{1+e^{\theta_i-\beta_j}}\right)^{(1-c_{ij})}$$

$$log(p(c\mid\theta,\beta)) = \Sigma_i\Sigma_j[(\theta_i-\beta_j) - c_{ij}log(1+e^{\theta_i-\beta_j}) - (1-c_{ij})log(1+e^{\theta_i-\beta_j})]$$

$$= \Sigma_i\Sigma_j[c_{ij}(\theta_i-\beta_j) - log(1+e^{\theta_i-\beta_j})]$$

The respective derivatives of log-likelihood with respect to θ_i and β_i are:

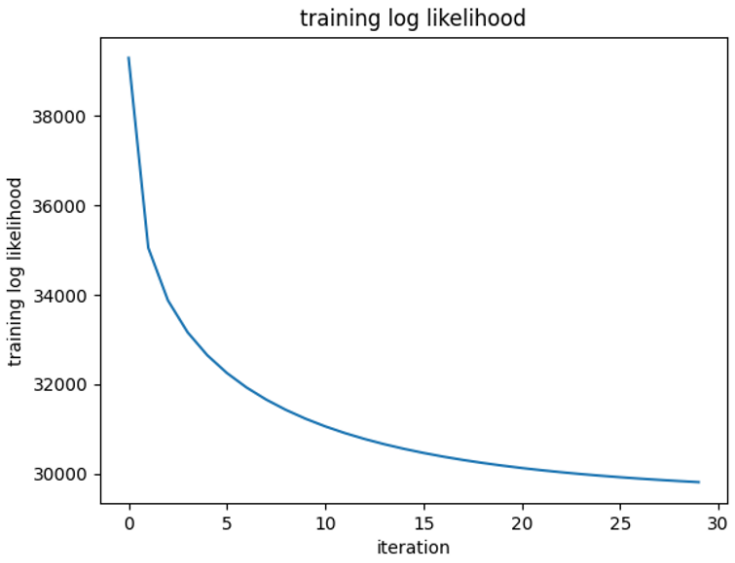
$$\frac{\partial log\, p(c|\theta,\beta)}{\partial\theta_i} = \Sigma_j\left(c_{ij} - \frac{\exp(\theta_i-\beta_j)}{1+e^{\theta_i-\beta_j}}\right)$$

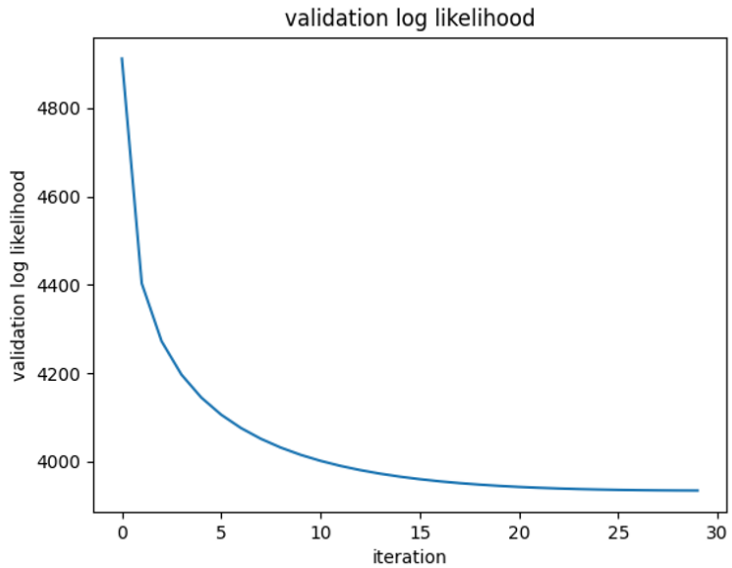
$$\frac{\partial log\ p(c|\theta,\beta)}{\partial \beta_j} = \Sigma_i(-c_{ij} + \frac{\exp(\theta_i - \beta_j)}{1 + e^{\theta_i - \beta_j}})$$

(b) The selected hyperparameters are:

- Learning rate: 0.01
- Iteration: 30

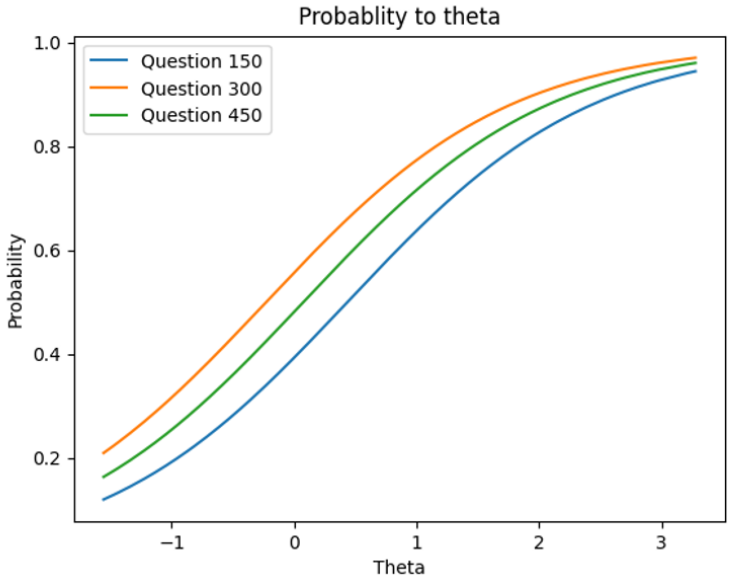
The training curves that shows the training and validation log-likelihoods as a function of iteration are:





(c) The final validation accuracy is 0. 7067;
The final test accuracy is 0. 7053.

(d) The probability of the correct response as a function of θ given question 150, question 300, and question 450 is plotted below:



All three curves in the figure above show an increasing trend. Theta represents a student's ability, and the y-axis is the probability that a student answers a question correctly. These increasing curves show that in general students with better ability have higher probability to choose a correct answer for the diagnostic questions.

3. Matrix Factorization

(a) We tried the following seven k values and the outputs are presented below:
k = 3, validation accuracy = 0.6583403895004234
k = 5, validation accuracy = 0.659046006209427
k = 7, validation accuracy = 0.6591871295512278
k = 9, validation accuracy = 0.6613039796782387
k = 11, validation accuracy = 0.656929156082416
k = 13, validation accuracy = 0.6559412926898109
k = 15, validation accuracy = 0.6565057860570138

We can observe that $k^* = 9$ gives the highest accuracy. So for $k^* = 9$, the final validation accuracy is 0. 6613, and the final test accuracy is 0. 6588.

(b) One possible limitation of the SVD is that missing values in the sparse matrix are filled with the mean value of the existing values in the corresponding column. The underlying assumption is that if a student gives correct answers to half of the questions he attempts to do, he will answer all questions correctly, which is not true and could lead to lower accuracy.

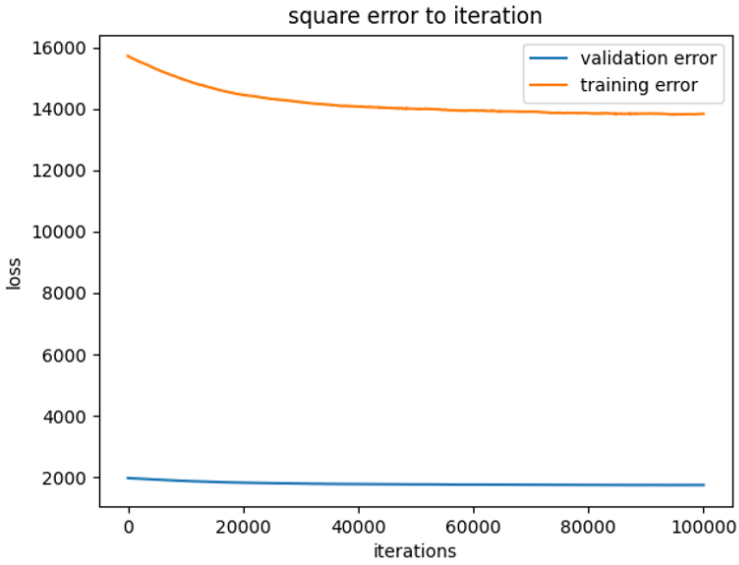
(c) Coding.

(d) The chosen hyperparameters are:
- Learning rate: 0.05
- Iterations: 100000

We tried the following nine k values and the outputs are presented below:
k = 3, validation accuracy = 0.6953147050522156
k = 5, validation accuracy = 0.6922099915325995
k = 7, validation accuracy = 0.698842788597234
k = 9, validation accuracy = 0.6958791984194186
k = 11, validation accuracy = 0.6965848151284222
k = 13, validation accuracy = 0.6867061812023709
k = 15, validation accuracy = 0.6994072819644369
k = 17, validation accuracy = 0.6930567315834039
k = 19, validation accuracy = 0.6915043748235958

We can observe that $k^* = 15$ gives the highest accuracy. With $k^* = 15$, the final validation accuracy is 0. 7003, and the final test accuracy is 0. 6918. Here the best k would sometimes be 7 or 9 due to the randomness introduced by stochastic gradient descent.

The plot below shows how the training and validation squared-error-losses change as a function of iteration. Both the training loss and the validation loss decrease as the gradient descent iterates.



4. Ensemble

Firstly, we resampled the training set to three smaller sets with replacement, and we ran the IRT algorithm with each resampled set. Next, we generated the binary predictions on the same test set with each resulting IRT algorithm and made the final decision according to the majority vote of each models' prediction. So, if two of the three predictions have the same outcome, then it will be the final prediction by ensemble. Finally, we calculated the accuracy on the training set, validation set and test set as shown below:

Original IRT
Train Score: 0.7388688964154672
Validation Score: 0.7068868190798758
Test Score: 0.7053344623200677

Ensemble IRT
Train Score: 0.7168889359300028
Validation Score: 0.6951735817104149
Test Score: 0.6931978549252046

Final validation accuracy: 0.6952
Final test accuracy: 0.6932

By comparing the final ensemble validation and test accuracies with the original statistics, we did not receive a better performance. This could be explained by the nature of the Item Response Theory (IRT), which is trying to learn the ability of students (θ) and the difficulty of questions (β). Bagging in this algorithm would reduce the information we can learn on a student or a question because of reduced size of training data, which might cause the algorithm to underestimate or overestimate the parameters θ and β. If this is the case, the accuracy of each model could decrease, which causes the overall accuracy to become lower.

Part B

Introduction

In order to enhance the performance of our machine learning model, we modified the core equation of item response theory (IRT) to catch more information from the datasets:

$$p(c_{i,j} = 1 | \theta_i, \beta_j, a_i, k_i) = a_i + (1 - a_i) \frac{\exp(k_i(\theta_i - \beta_j))}{1 + \exp(k_i(\theta_i - \beta_j))}$$

This equation would possibly perform better than the original one since it takes the randomness of correct guesses and how discriminative a question is into account. It is unlikely that a student has no possibility to answer a multiple choice question correctly even if the question is extremely difficult, so we added parameter *a* which represents the lowest possibility a student can answer a given question correctly via a random guess. This is straightforward. Meanwhile, how a question can test a student's ability also matters. Some questions can reflect a student's ability better than others, so we added the parameter *k* which differentiates the quality of a question. That is, a question with larger *k* could signify the influence of a student's ability and the question's difficulty on the possibility a student can answer it correctly.

Furthermore, since we found that these variables are different in terms of stability in gradient descent, we set different learning rates for different variables. This is intended to improve the optimization and accuracy of the prediction.

Algorithm

```
// Training

For i in range(iteration):
    For j in data:
        a += derivative multiplied by learning rate,
        k += derivative multiplied by learning rate,
        θ += derivative multiplied by learning rate,
        β += derivative multiplied by learning rate
        Check if a is between 0 and 1

Evaluate(a, k, θ, β)    """ Evaluate the accuracy with the updated a, k, θ,
β using our newly defined probability density function."""

"""Count the number of correct predictions where the calculated probability
≥ 0.5 for positive cases counts as correct and vice versa. """
```

Partial Derivatives for Gradient Descent

$$p(c \mid \theta, \beta, a, k) = \Pi_{i,j} \left(a + (1-a)\frac{\exp(k_i(\theta_i-\beta_j))}{1+\exp(k_i(\theta_i-\beta_j))}\right)^{c_{ij}} \cdot \left(1 - a - (1-a)\frac{\exp(k_i(\theta_i-\beta_j))}{1+\exp(k_i(\theta_i-\beta_j))}\right)^{1-c_{ij}}$$

$$log(p(c \mid \theta, \beta, a, k)) = \sum_i \sum_j [c_{ij} log\left(a + (1-a)\frac{\exp(k_i(\theta_i-\beta_j))}{1+\exp(k_i(\theta_i-\beta_j))}\right) + (1-c_{ij}) log\left(1 - a - (1-a)\frac{\exp(k_i(\theta_i-\beta_j))}{1+\exp(k_i(\theta_i-\beta_j))}\right)]$$

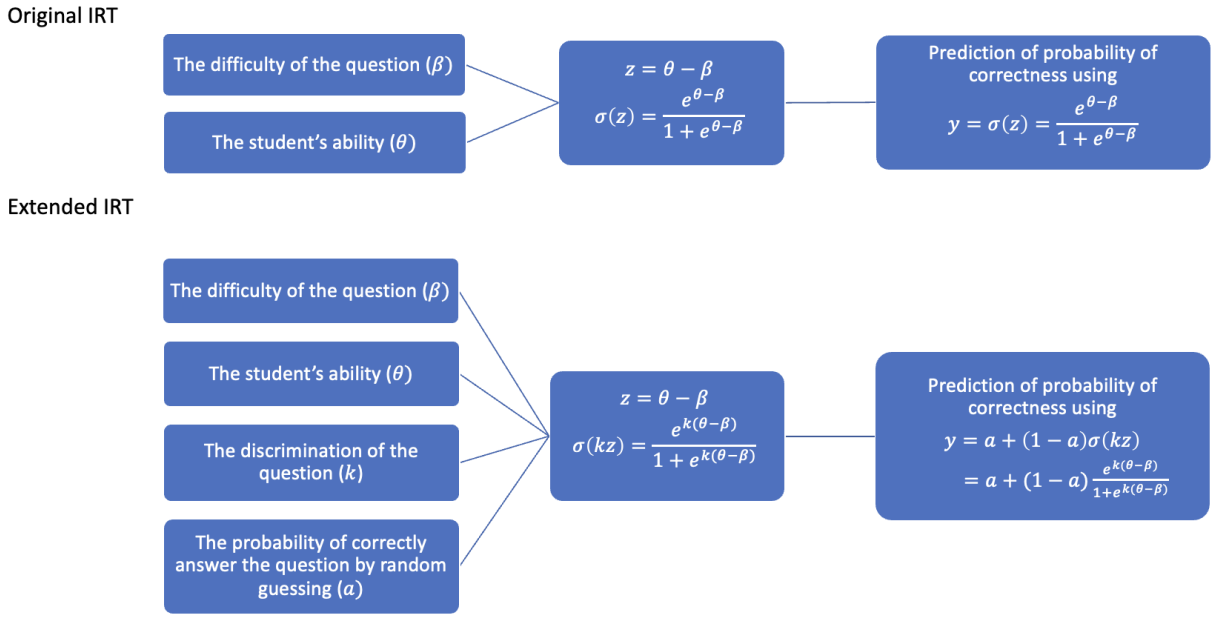
$$\frac{\partial}{\partial \theta_i} log(p(c \mid \theta, \beta, a, k)) = \sum_j \frac{k_j e^{\theta i}((c_{ij}-1)k_j e^{\theta i} + (c_{ij}-a_j)e^{\beta j})}{(k_j e^{\theta i} + e^{\beta j})(k_i e^{\theta i} + a_j e^{\beta j})}$$

$$\frac{\partial}{\partial \beta_j} log(p(c \mid \theta, \beta, a, k)) = -\sum_i \frac{k_j e^{\theta i}((c_{ij}-1)k_j e^{\theta i} + (c_{ij}-a_j)e^{\beta j})}{(k_j e^{\theta i} + e^{\beta j})(k_j e^{\theta i} + a_j e^{\beta j})}$$

$$\frac{\partial}{\partial a_j} log(p(c \mid \theta, \beta, a, k)) = \sum_j \frac{e^{\beta j} a_j - c_{ij} e^{\beta j} + (1-c_{ij})e^{\theta i}}{(a_j-1)(k_j e^{\theta i} + a_j e^{\beta j})}$$

$$\frac{\partial}{\partial k_j} log(p(c \mid \theta, \beta, a, k)) = \sum_j \frac{e^{\theta i}((c_{ij}-1)e^{\theta i} k_j + (c_{ij}-a_j)e^{\beta j})}{(k_j e^{\theta i} + e^{\beta j})(k_j e^{\theta i} + a_j e^{\beta j})}$$

Diagram of the modified model



Comparison and analysis

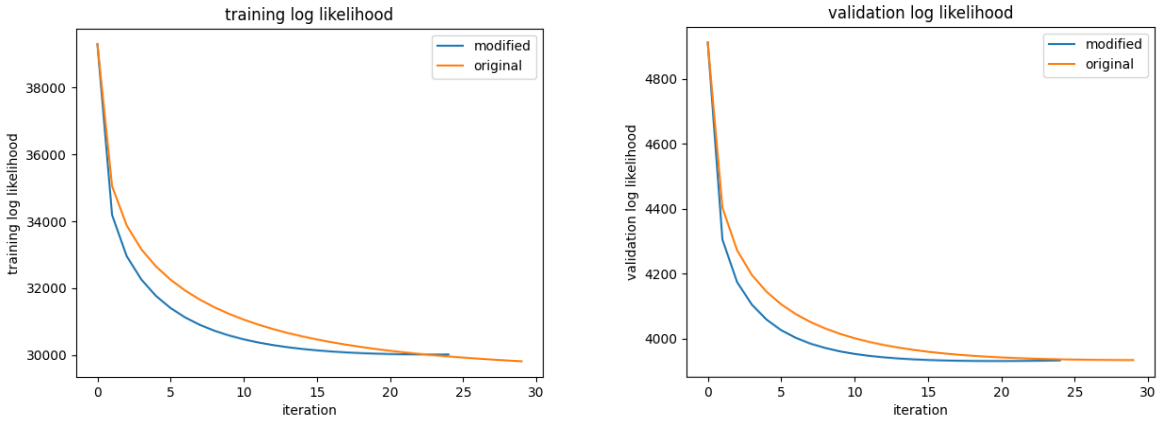
Hyperparameters:
Original:
- Iteration: 30
- Learning rate for θ and β: 0.01

Modified:
- Iteration: 25
- Learning rate for θ and β: 0.015
- Learning rate for k: 0.0015
- Learning rate for a: 0.00075

Comparison between the original model and the modified model:

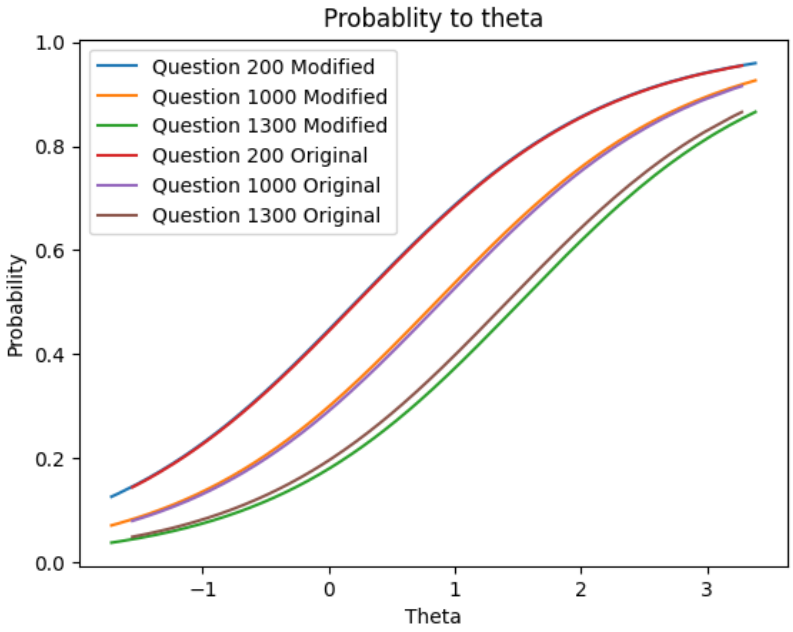
	Test accuracy	NLLK	Validation accuracy
Original IRT	0.7053344623200677	29814.94707407383	0.7067456957380751
Modified IRT	0.7061812023708721	30019.36892369842	0.7066045723962744

The comparison of the result of the base model and result shown as a graph of extended:



From the figures above, we can see that our new model learns quicklier than the original one, which is a result of a higher learning rate of theta and beta. Although our model has a higher NLLK in the end, we have a similar final validation accuracy and higher test accuracy. At iteration 25, our model started to show a trend of overfitting. This indicates that our new model has been optimized.

From the figure below we can see that both theta and the probability in our modified model generally have wider distribution. Compared with the original model, our model is able to predict the outcome more accurately since it covers a wider distribution of probability. This might also indicate that our model is extracting more information from the dataset.



Limitations

One of the settings no model would work is that the student is guessing completely randomly and no model should be able to predict that. This is purely just the nature of this scenario.

- Running time could be a big limitation upon large datasets when training since the modified version of gradient descent is now more complicated.

- a and k are very sensible to learning rate comparing to Θ and β. That is, our newly introduced variables are more likely to behave unstably under the same learning rate as Θ and β.

- There could be invalid values for variable a which could cause the log function to fail (i.e. a should always be in [0, 1]), so we had to constrain this variable in gradient descent, which might cause an under-generalizing problem and lower the overall performance.

- The underlying assumption is that one student's ability applies to every question, but it could be the case that a student is very good at one subject but could perform devastatingly bad at the other.

One possible extension to solve the above limitation is to give an ability-to-subject pair to represent the ability of the student given the subject of the problem better, and do alternating gradient descent on it.

Contributions

ZeZheng Gu:

- Responsible for code proofreading and refinements.
- Optimized solutions of IRT and Matrix Factorization.
- Completed the formal description in part B and drew the figure.
- Showed limitations of our approach and explained why.
- Came up with possible extensions to address the limitations.

Jiaqi Guo:

- Algorithm implementation in both part A & part B
- Calculus for gradient descent
- Tuning hyperparameters for best performance
- Generating results and figures for the report
- Described the modified algorithm in part B of the report

Shang Wu:

- Responsible for optimizing the answer to our solutions
- Generalized conclusions to part A of the report
- Optimized, and organized the presentation of report
- Generated diagrams for the original and modified models
- Solved complicated IRT gradient descent to improve accuracy of our extended solution

All of our team members participated in each and every problem solving process throughout the project.