Part B

Algorithm Description

Algorithm Idea

In this part, we modified the **item response** algorithm and we would expect the item response algorithm with our extension should have a better performance. There are two main modifications in our algorithm:

- Apply item response algorithm on each category of the questions.

  Explanation: Split the questions into different categories: **Number, Algebra, Geometry, Statistics and Other**. The reason why we split dataset in this way is when we deal with the dataset, it is interesting to find even all the questions are belong to at least one of those four categories. Then based on those questions, split the original dataset into 5 sub-datasets. (i.e. For each sub-dataset, it only records the responses to that category of question.) For example, if question 1 is a **Number** question, then all responses of question 1 should be in the sub-dataset that records **Number** question.

- Update the equation of the probability of calculation, that is using $p(c_{ij}|\theta_i, \beta_j) = c + (1 - c) \times sigmoid(k_j(\theta_i - \beta_j))$

  Explanation: Compared to the original version of item response, θ_i and β_j represents the ability of the student with student_id i and the difficulty of the question with question_id j respectively. Then c is the probability of getting question right via random guess. k_j is how steep the sigmoid looks (i.e. how discriminative the question is)

Algorithm Analysis

As reported in the **Question 2**, we know the metrics of the **Original Item Response** as following:

	training	validation
Accuracy	0.739	0.705
NLLK	29718.216	3935.151

Test Accuracy: 0.7047

In addition, if we observe the training and validation curves during the training process, we can see there is no big difference of the accuracy on training dataset, on validation dataset and on the test dataset. This means the variance of the model is low and the phenomenon of the overfitting to this model is not severely. However, the accuracy of training dataset is only 0.739, this means the model is suffering from underfitting. In this way, in order to improve the model, we will increase the complexity of the **Original Model**. Therefore, we come up with two modifications.

- In the first modification, before we train the model, we do a clustering for the questions. That is we can **classify questions into different categories.** The advantage of this step is we can reduce the noise of the data. To be more specific, the distribution of $p(\theta, \beta|some\ category)$ might be different, then we can consider those seperately. If we consider this model intuitively, θ represents the ability of the student and β indicates the difficulty of the question. However, some students might good at algebra and statistics but they are not good at geometry, or some people might be interested in number but not keen on algebra. In this way, we will calculate the ability of the student when they do algebra questions and evaluate the difficulty of the question given all algebra questions. Instead of using overall ability of the student and overall difficulty of the questions, we can use more detailed information to analysis the result. This improve the complexity of this model and it supposed to have a better performance.

- In the second modification(mentioned in tutorial), we make the model more complex than the original one. Since for each question, if the student do not know anything and finish the question by guessing, the probability that they can get the correct answer is not 0. In this way, we can add a constant c which represents the probability to get the correct answer for the questions by guessing. In addition, different questions also have different feature, then we use k to describe how discriminative the question is. This methods can help the model to make more precise prediction on the each questions, therefore it should be have a better performance.

Algorithm Box

Notice: Specific derivations can be found on the last page of the report.

Result: Split the datasets
Input : Original Dataset
Output: Sub-datasets for each category

for *datapoint* $\in$ *OriginalDataset* **do**
  Initialize NumberData, GeometryData, AlgebraData, StatisticsData, OtherData
  if *datapoint can be classified as number, geo, alg, sta question* **then**
    put the datapoint into corresponding dataset
  else
    put the datapoint into OtherData
  end
end
return *NewDatasets*

Algorithm 1: Split Dataset

Result: Calculate Accuracy for Dataset
Input : Dataset, C, θ, β,
Output: Accuracy of the Dataset

total $= 0$, correctCount $= 0$
for *datapoint* $\in$ *OriginalDataset* **do**
  Find the category of the datapoint
  Calculate the probability of the datapoint with corresponding C, θ, β, k *by* $p(c_{ij}|\theta_i, \beta_j) = c + (1 - c) \times sigmoid(k_j(\theta_i - \beta_j))$
  $predictedResult = \lfloor probability + 0.5 \rfloor$
  Compare predictedResult and true result of the datapoint
  update total and correctCount
end
return $\frac{correctCount}{total}$

Algorithm 2: Calculate Accuracy

Result: Train Model
Input : Dataset, learningRate, iterations, C
Output: θ, β, k

Initialize θ, β, k
while *iterations* > 0 **do**
  Calculate θ, β, k by the formula on the last page
  $\theta = \theta - lr \times d\theta$
  $\beta = \beta - lr \times d\beta$
  $k = k - lr \times dk$
  $iterations = iterations - 1$
end
return θ, β, k

Algorithm 3: Train Model

Result Comparison

Overall, the test accuracy of the **original model** is 0.7047 and the accuracy of the **modified model** is 0.7144. We can see a improvement of our modified model. To be more detailed, I will show metrics for each category of my **modified model** and **original model** as following:

Modified Model

	Training Accuracy	Validation Accuracy	Test Accuracy
Number	0.7364	0.72062	0.7182
Algebra	0.7567	0.70	0.71842
Statistics	0.7531	0.7003	0.6918
Geometry	0.7417	0.7040	0.6962
Other	0.7913	0.6379	0.4643

Note: In order to improve some low accuracy, I use the **modified probability model**, to train a proper θ, β, k and I use the newly trained parameters to predict when the categorical model performance is not good. That is the reason why I can get 0.7144 test accuracy in the end.

Original Model

	Validation Accuracy	Test Accuracy
Number	0.7202	0.7007
Algebra	0.6887	0.7079
Statistics	0.7186	0.6950
Geometry	0.6952	0.71786
Other	0.65517	0.6428

Number of datapoints for each category:

number: 25073, algebra: 12187, geometry: 14388, statistics: 5177, other: 508

Compared the result above, it is obviously for us to observe that our modified model can acheive a much better result in **Number, Algebra** categories than the original model while others might not that good or there is no huge difference. Since we split the data into different categories, the number of data for each category will be influenced, especially for Other category, where we can observe a overfitting. Since if we do not have enough data for some categories, then we cannot extract the feature for that category of questions and students abilities, therefore, it is better for us to use the **overall** evaluation to evaluate whether the student can answer the question correctly.

Overall, our modified model can acheive a better performance than the original model.

Limitations

Based on our modified model, since we need to split the data into different categories, then the quantity of data in each category should be smaller than the original category. In this way, our model might suffer from overfitting and it will result in a bad performace. Then if the data is not enough, it is better not to use the modified algorithm.

However, to solve this problem, we can **use all data to train the model and get some parameters**. For those categories with less data, we can use those parameters trained by all data to predict the result since the overall ability of students and difficulties of questions also can reflect the result of the performance of that students on some questions to some extent.

Another method to solve for lack of data is we can use **bagging** to improve the accuracy of the model. Since as we know, bagging technique can be helpful to reduce the variance of the model. Since lack of data always result the model in overfitting, the bagging might be helpful to solve this problem.

In addition, to extend the model further, we only use metadata of questions to improve the performance of the model, we can also use metadata of students to improve the performance of the model. For example, we know the age of the students in the students metadata, based on common sense, if the student is too young, the student is not able to solve some hard questions especailly for some difficult math questions. Then in this way, we can consider find more detailed classes based on both age of students and question category. However, this methods might also suffer from lack of data for each sub-category.

θ, β, k Updates