Final Project

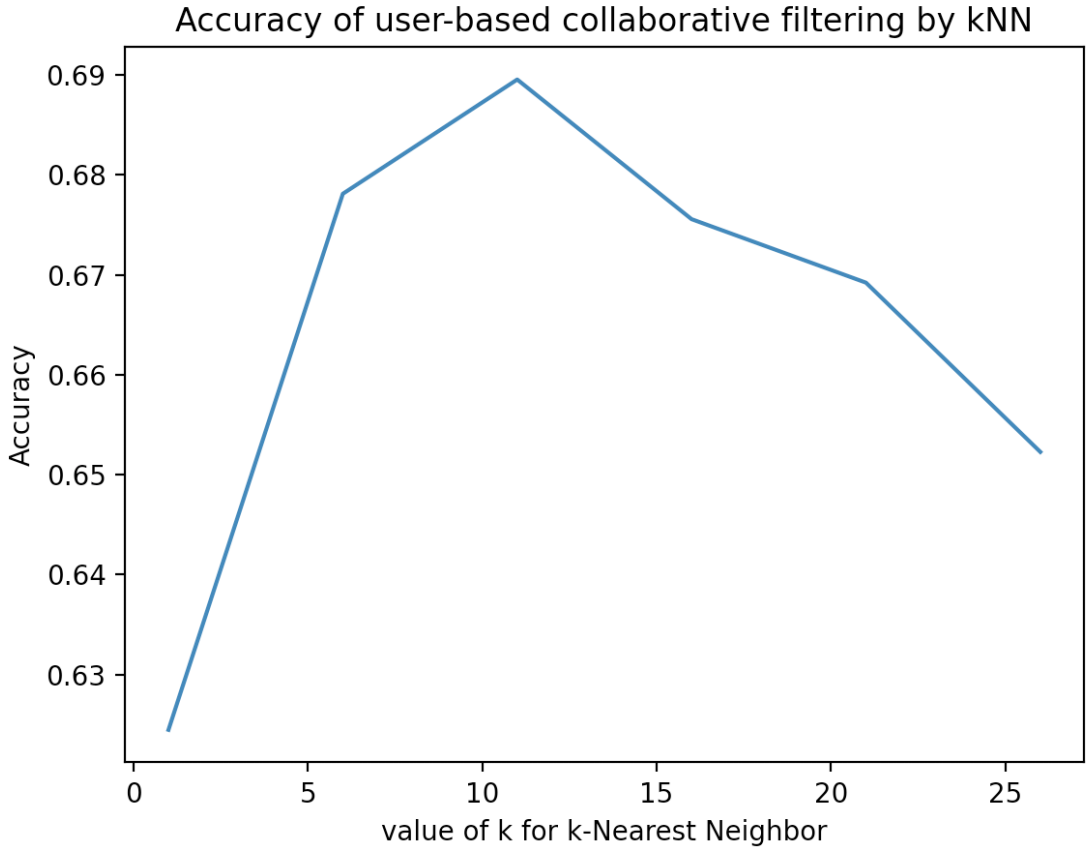
Yuchen Zeng, Zewen Ma, Haoze Huang

December 3, 2021

Part A

Question 1

(a) By using user-based collaboratvive filtering, the accuracies for kNN using different k value on validation set is as the following:

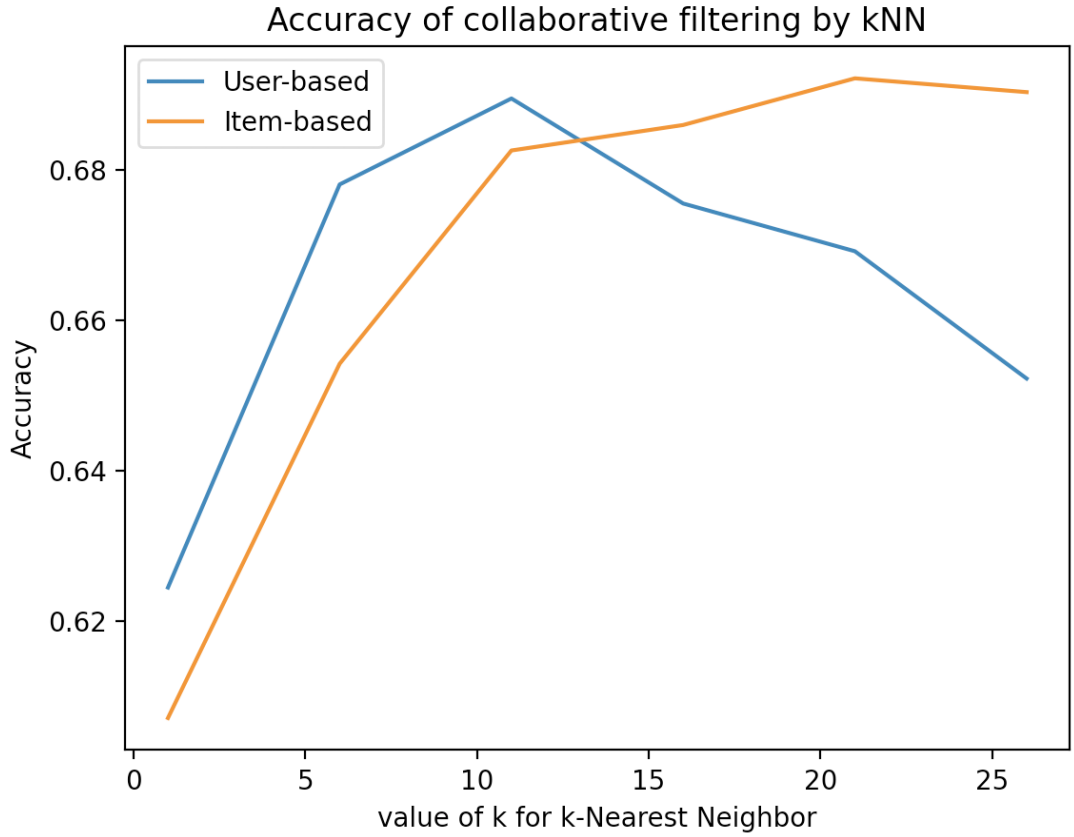


```
With k=1
Validation Accuracy: 0.6244707874682472
With k=6
Validation Accuracy: 0.6780976573525261
With k=11
Validation Accuracy: 0.6895286480383855
With k=16
Validation Accuracy: 0.6755574372001129
With k=21
Validation Accuracy: 0.6692068868190799
With k=26
Validation Accuracy: 0.6522720858029918
```

(b) By the result from (a), we chose $k* = 11$ because $k = 11$ yeilds the highest accuracy for validation set.
The final test accuracy is the following:

```
Using user-based collaborative filtering, k=11 on test set:
Test Accuracy: 0.6841659610499576
```

(c) By using item-based collaboratvive filtering, the accuracies for kNN using different k value on validation set is as the following:



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

By the result above, we chose $k* = 21$ because $k = 21$ yeilds the highest accuracy for validation set.

The final test accuracy is the following:

```
Using item-based collaborative filtering, k=21 on test set:
Test Accuracy: 0.6816257408975445
```

(d) By the result from (b) and (c), we can see that althogh item-based collaborative filtering has a higher validation accuracy of 69.22%, comparing to the accuracy of 68.95% by user-based collaborative filtering. However, the test accuracy of user-based collaborative filtering is 68.42% comparing to the accuracy of 68.16% by item-based collaborative filtering. Therefore, the test accuracy of user-based collaborative filtering is about 0.2% higher. Therefore, I think user-based collaborative filtering performs better.

(e) The first potential limitation of kNN is that it can only produces or predict a student's answer to a question based on answers from other students or the student's answer to other questions. Therefore, it is effected to missing values. Since there is many missing value in the sparce matrix, the prediction is less accurate.

The second potential limitation is large time complexity when working with large datasets. Since to perform kNN we need to calculate the distance between the observation and every other observations, the process can become extremely slow when we have a large training dataset.

Question 2

(a) I derived the formulas as following:

(a) Given $p(c_{ij}=1 \mid \theta_i, \beta_j) = \frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i - \beta_j)}$

$$p(c_{ij}=0 \mid \theta_i, \beta_j) = 1 - \frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i - \beta_j)}$$

$$= \frac{1+\exp(\theta_i - \beta_j) - \exp(\theta_i - \beta_j)}{1+\exp(\theta_i - \beta_j)}$$

$$= \frac{1}{1+\exp(\theta_i - \beta_j)}$$

$$p(c_{ij} \mid \theta_i, \beta_j) = \left[\frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i - \beta_j)}\right]^{c_{ij}} \left[\frac{1}{1+\exp(\theta_i - \beta_j)}\right]^{(1-c_{ij})}$$

Since not all student in the sparce matrix answers all questions.
Let (i,j) denote a pair of student i, question j such that i answered j, and c_{ij} represent the correctness.

$$p(C \mid \theta, \beta) = \prod_{(i,j) \in C} \left[\frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i - \beta_j)}\right]^{c_{ij}} \left[\frac{1}{1+\exp(\theta_i - \beta_j)}\right]^{(1-c_{ij})}$$

$$\log p(C \mid \theta, \beta) = \sum_{(i,j) \in C} c_{ij} \log\left[\frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i - \beta_j)}\right] + (1-c_{ij}) \log\left[\frac{1}{1+\exp(\theta_i - \beta_j)}\right]$$

Note that $\frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i - \beta_j)} = \frac{\exp(\theta_i - \beta_j + \beta_j - \theta_i)}{\exp(\beta_j - \theta_i) + \exp(\theta_i - \beta_j + \beta_j - \theta_i)} = \frac{1}{1+\exp(\beta_j - \theta_i)}$

$$\log p(C \mid \theta, \beta) = \sum_{(i,j) \in C} c_{ij} \log\left[\frac{1}{1+\exp(\beta_j - \theta_i)}\right] + (1-c_{ij})\left(-\log\left(1+\exp(\theta_i - \beta_j)\right)\right)$$

$$= \sum_{(i,j) \in C} -c_{ij} \log(1+\exp(\beta_j - \theta_i)) + (c_{ij}-1)\left(\log\left(1+\exp(\theta_i - \beta_j)\right)\right)$$

$$\frac{\partial \log p(C \mid \theta, \beta)}{\partial \theta_i} = \sum_{(i,j) \in C} \left[c_{ij} \frac{1}{1+\exp(\beta_j - \theta_i)} \exp(\beta_j - \theta_i) + (c_{ij}-1) \frac{1}{1+\exp(\theta_i - \beta_j)} \exp(\theta_i - \beta_j)\right]$$

pair of (i,j) → where i is a constant

$$= \sum_{(i,j) \in C} \left[c_{ij}\left(\frac{\exp(\beta_j - \theta_i)}{1+\exp(\beta_j - \theta_i)} + \frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i - \beta_j)}\right) - \frac{\exp(\theta_i - \beta_j)}{1+\exp(\theta_i - \beta_j)}\right]$$

$$\frac{\partial \log P(C \mid \theta, \beta)}{\partial \theta_i} = \sum_{(i,j) \in C} \left[c_{ij} \left(\frac{\exp(\beta_j)}{\exp(\beta_j)(\theta_i) + \exp(\theta_i)\exp(\beta_j)} + \frac{\exp(\theta_i)}{\exp(\beta_j)\exp(\theta_i) + \exp(\theta_i)} \right) - \frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)} \right]$$

$$= \sum_{(i,j) \in C} \left[c_{ij} - \frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)} \right]$$

$$\frac{\partial \log P(C \mid \theta, \beta)}{\partial \beta_j} = \sum_{(i,j) \in C} \left[c_{ij} \frac{-1}{1 + \exp(\beta_j - \theta_i)} \exp(\beta_j - \theta_i) + (c_{ij} - 1) \frac{-1}{1 + \exp(\theta_i - \beta_j)} \exp(\theta_i - \beta_j) \right]$$

pair of $(i, j) \Rightarrow$ where j is a constant

$$= \sum_{(i,j) \in C} -c_{ij} \left[\frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)} + \frac{\exp(\beta_j - \theta_i)}{1 + \exp(\beta_j - \theta_i)} \right] + \frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)}$$

$$= \sum_{(i,j) \in C} \left[-c_{ij} + \frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)} \right]$$

$$= \sum_{(i,j) \in C} \left[\frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)} - c_{ij} \right]$$

Therefore:

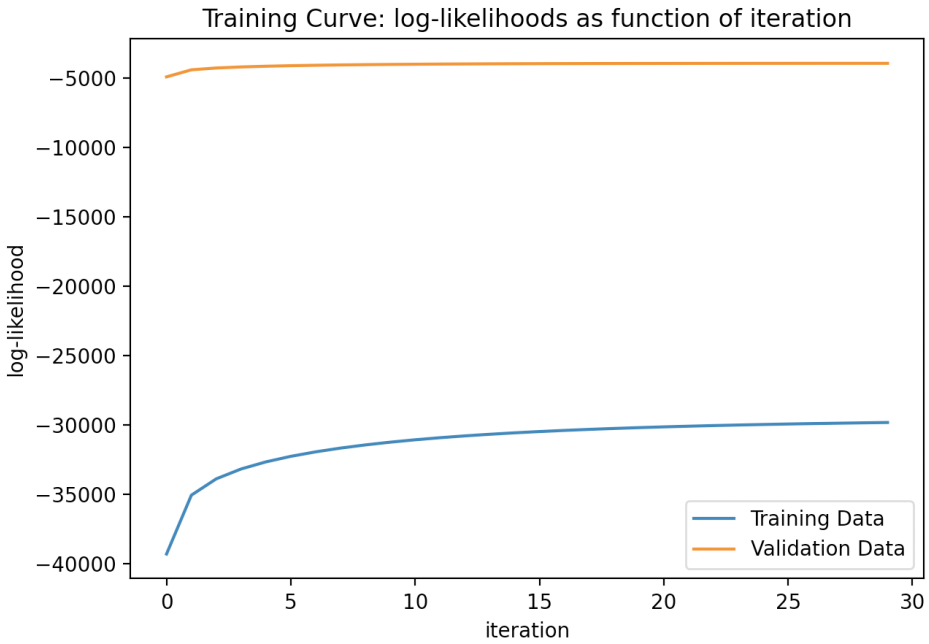
$$\log p(C \mid \theta, \beta) = \sum_{(i,j) \in C} \left[-c_{ij} \log(1 + \exp(\beta_j - \theta_i)) + (c_{ij} - 1)\left(\log(1 + \exp(\theta_i - \beta_j))\right) \right]$$

↝ total of 56688 pairs

$$\frac{\partial \log P(C \mid \theta, \beta)}{\partial \theta_i} = \sum_{(i,j) \in C} \left[c_{ij} - \frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)} \right] \quad (i \text{ is fixed})$$

$$\frac{\partial \log P(C \mid \theta, \beta)}{\partial \beta_j} = \sum_{(i,j) \in C} \left[\frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)} - c_{ij} \right] \quad (j \text{ is fixed})$$

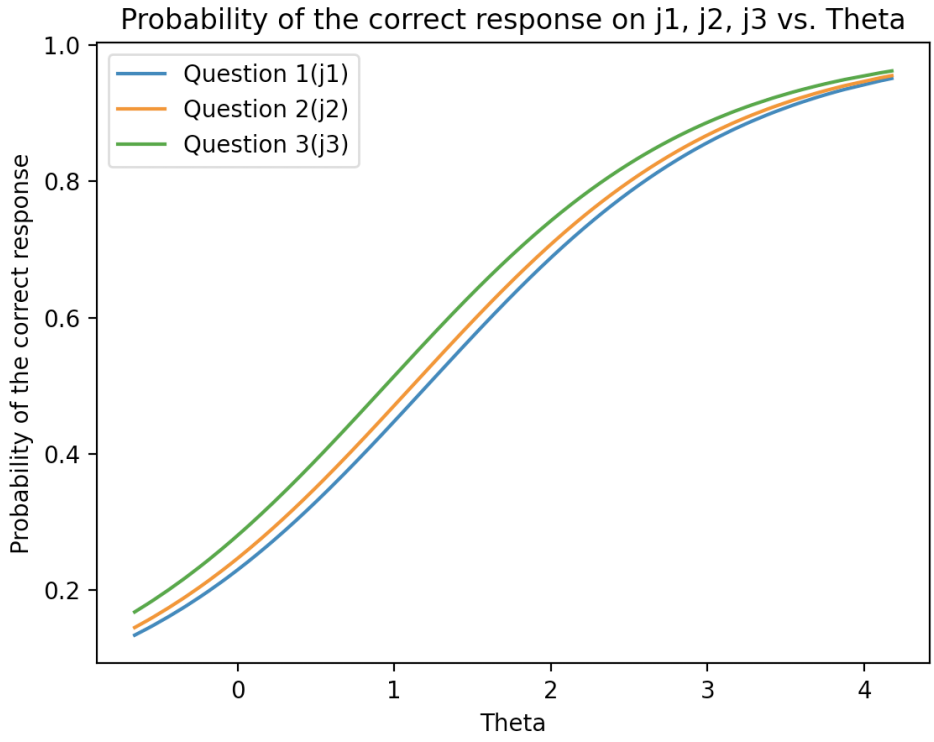
(b) I start experimenting with the following parameter:
learning rates: [0.01, 0.03, 0.05, 0.07, 0.1]
number of iterations = [10, 30, 50, 70, 100]
I initialize all θ and β to be 1.
I find that learning rate of 0.01 with 30 iterations yield the highest validation accuracy.
Therefore, I chose **learning rate=0.01, iteration=30** as my hyperparameter.
The following is training curve of training and validation log-likelihoods as a function of iteration:



(c) By using item-based collaboratvive filtering, the accuracies for kNN using different k value on validation set is as the following:

```
Validation Accuracy:  0.7067456957380751
Test Accuracy:  0.7053344623200677
```

(d) I choose the first 3 questions in the *training data* dictionary.
The shape of all curves are similar to a sigmoid. The value becomes more saturated when $\theta \to \pm\infty$. These curves represent the relationship between a student's ability and the prabability that this student answers a question correctly. The higher the ability, the more likely the student can answer the question correctly. However, the increse saturate when ability is too high.



Question 3

We choose option 2: Neural Networks.

(a) Differences:
1. Neural networks update all parameters at the same time using a matrix representation, while ALS updates one parameter at a time.
2. Neural networks uses backward propagation to get gradient, while ALS doesn't.
3. Neural networks is used for supervised learning, while ALS is used for unsupervised learning.

(b) The code is implemented in the python file.

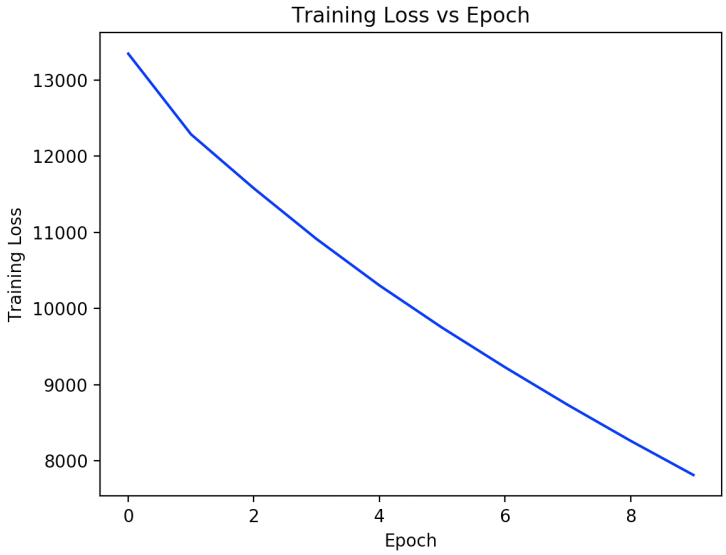
```
def forward(self, inputs):
    """ Return a forward pass given inputs.

    :param inputs: user vector.
    :return: user vector.
    """
    ###########################################################################
    # TODO:                                                                   #
    # Implement the function as described in the docstring.                   #
    # Use sigmoid activations for f and g.                                    #
    ###########################################################################
    act_g = F.sigmoid(self.g(inputs))
    act_f = F.sigmoid(self.h(act_g))
    out = act_f
    ###########################################################################
    #                       END OF YOUR CODE                                  #
    ###########################################################################
    return out
```

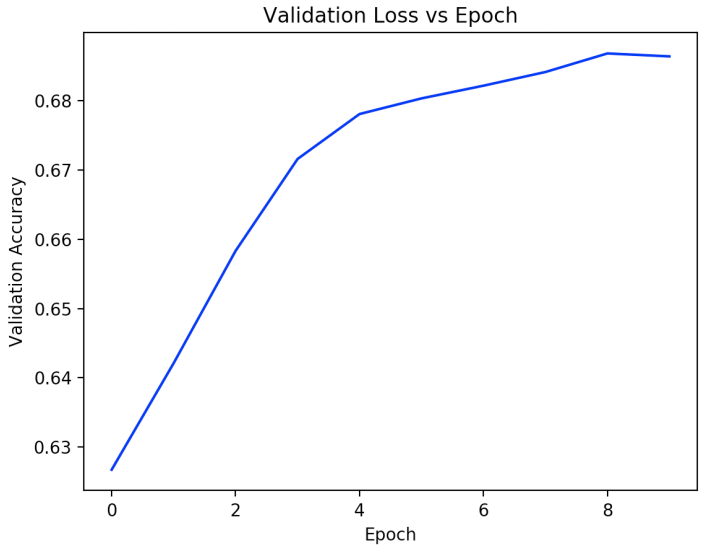
(c) When working on question (c), we get the following warning:
UserWarning: nn.functional.sigmoid is deprecated. Use torch.sigmoid instead. warnings.warn("nn.functional.sigmoid is deprecated. Use torch.sigmoid instead.")
We choose learning rate of 0.05, num of epoch of 25, and lambda of 0.01 for the k-values in [10, 50, 100, 200, 500]. The best validation accuracy of each k are shown below:

k	Epoch	Accuracy
10	11	0.6833
50	10	0.6868
100	8	0.6818
200	7	0.6767
500	9	0.6684

Therefore we choose $k = 50$, which has the highest accuracy of 0.6868, according to the result table above.

(d) The graphs of changes of training and validation objectives depending on epoch are shown below:





The final test accuracy is 0.6830.

(e) The final validation and accuracy of each λ is shown below:

λ	Test Accuracy	Validation Accuracy
0.001	0.6881	0.6881
0.01	0.6827	0.6754
0.1	0.6216	0.6212
1	0.6252	0.6232

Based on the table shown above, I would choose $\lambda = 0.001$, which has the highest validation and test accuracy of 0.6881.

According to the result, it seems that the model with regularization penalty performs better. I think that is because regularization reduces parameters and simplifies the model, which avoids overfitting by the penalizing high-valued coefficients.

Question 4

For this question, I will use the neural network model which is done in question 3 as the base model.
There are two functions in file `ensemble.py`, which are the `generate_sample` and `bagging`.
The `generate_sample` function first randomly generate a sample with size of `num_student`, then will build corresponding new 2D FloatTensors (`new_zero_train_matrix` and `new_train_matrix`). Then, using the k-value, learning rate, number of epoch, and λ we choose in question 3 ($k = 50$, learning rate $= 0.05$, epoch $= 10$, $\lambda = 0.001$) to build a sample model. The function will return the model and `new_zero_train_matrix`.
The function `bagging` takes three base models corresponding to the three groups of sample data. Then evaluate the predictions and return accuracy.
The **final validation accuracy** is 0.5975.
The **final test accuracy** is 0.6102.
Both accuracies are smaller than the neural network model in question 3, I think that is because, during sampling, the data are not recorded in all three samples.

Part B

1. Formal Description

We have tried various ways to modify the algorithms from part A using concepts from turotials and lectures, as well as research on different variations of a specific algorithm. More specifically, we have tried to use concepts from turotial 2 which mentioned cosine similarities, which is a measure of the dot product and magnitude of each vector, can also be used as a way to distinguish the nearest neigbours. Further more, article from Medium Magzine suggested that in some cases (when euclidean distance are the same), cosine has some meaningful semantics for ranking similar observations, hence increasing the prediction score. Therefore, the first modification we have done is to use cosine similarties to implement k-nearnest-neighbours.

When implementing item response theory, we used β for difficulty value to formulate a probability distribution. The second modification was inspried by an artical from Public Health Columbia where it suggested instead of 1 parameter β implemented in part a, this article also mentioned a way to implement 2 parameter logsitic model for item-response theory using β and α. The probability for the response are now weighted by the item discrimination factor for each response. In addition to θ and β, the new discrimination parameter α which can be changed by items. We predicts that the new model would improve the overall accuracy since the new paramter allows us to find additonal pattern with in each questions, so that it will have different curve for each questions in regrards to students abilities and question difficulties.

2 Parameter Item Respose Theory Model:

$$P_{ij}(\theta_j, b_i, a_i) = \frac{\exp[a_i(\theta_j - b_i)]}{1 + \exp[a_i(\theta_j - b_j)]}$$

Where j refers to user_ids from meta data, $j \leq 542$,
i refers to question from meta data, $i \leq 1774$.

Item information function:

$$I_i(\theta, b_i, a_i) = a_i^2 P_i(\theta, b_i) G_i(\theta, b_i)$$

as the discriminating factor is allowed to vary between items now.

We have also considered the improvement for neural network algorithm by adding one hidden layer, since we hypothesised that the layers in implementation from part a's linear representation might be not enough to approximate the output, so adding one more linear representation has some chances to improve the accuracy.
Adding a hidden layer involves a series of changes, therefore we created a separate file called `hidden_layer.py` in which the basic functionalities are kept expect adding an extra hidden/transition layer. We keep the chosen variables in part A question 3: $k = 50$, learning rate $= 0.05$, epoch number $= 10$, $\lambda = 0.001$.
The **final validation accuracy** is 0.6849 and the **final test accuracy** is 0.6785, which are a bit smaller than what we get in question 3 (recall the validation accuracy of 0.6881 and test accuracy of 0.6881).

We have also decided to explore on the *student_meta.csv* which records students' id, gender and date of birth. A study from Hyde & Lamon concluded that males perform better on mathematics tests than females do, as our data from eedi contains all mathematical diagnostic questions, gender might be a factor that affect the accuracy of answering questions correctly. Also, people who are elder tends to have more knowledge in the respected field, consequantly have better score. Therefore, age might be

a factor that affect the accuracy of answering questions corretly. Lastly, accoridng to Economic Policy Insituite, being in financially disadvantage is a main factor affecting student's academic performance.

By the evidence from research, we decided to split the datasets by each of the critarion and test the accuracy individually to see if there is any increase in the accuracy score. Due to we do not have enough observation in the dataset for financially disadvantaged students, the result we obtains from spliting by this criteria is insufficent. We have also tried split dataset into age groups by 4 quartiles, and also by teens (12+). Both the split criteria (gender, and ages) has significant higher training accuracy (increased by 8%), but in average lower and test accuracy (decreased by 1%). Therefore, we concluded that splitting by these criteria may have overfit the model.

Finally, **we picked 2 parameter item response model** for modification to predict students' answers to the disgnostic questions. This decision was based on the final test accuracy as well as the efficiency of the program, since adding one paramter to the equation of gradient decent has relatively less runtime than cosine similairty and adding layers and 2 paramter item response model also resulted in more promising accuracy.

2. Figures or Diagram

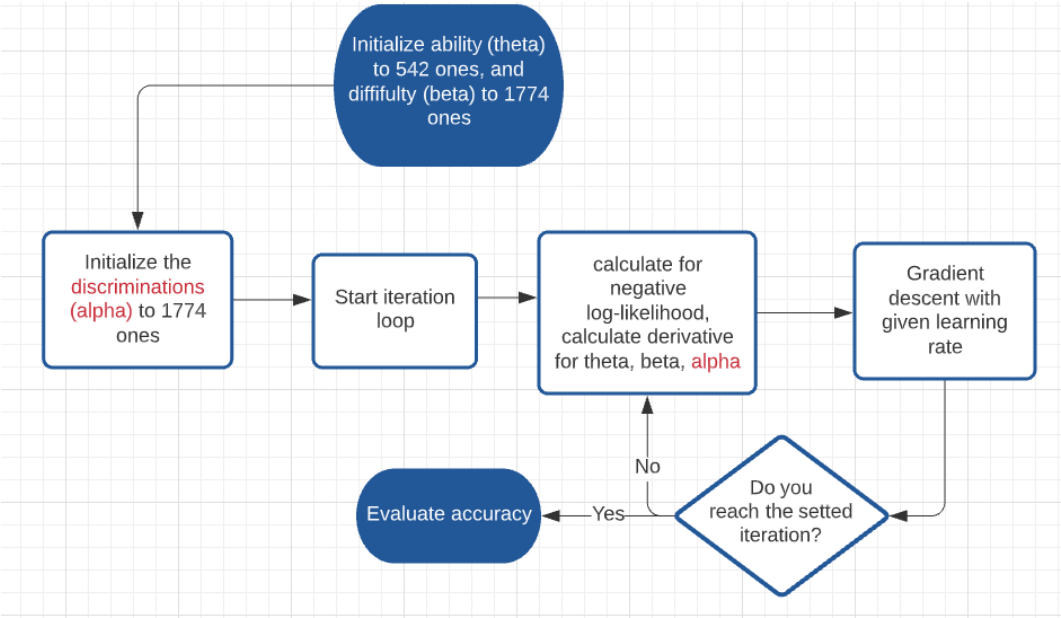


Figure 1: Flow chart of 2 parameter IRT model

From the original base IRT model to 2 paramter IRT model, we have all the majority steps such as gradient descent unchanged, the newly added paramter α is coded in red to reflect the difference between procedures of two algorithms.

The equations below demonstrate the derivation of the new model, then we performed alternting gradient descent on all the parameters θ, β, α with the learning rates, updating the paramter in each iteration and consequantly fit into the model and predict for testing.

Given the original model implemented in Part A. $P_{ij}(\theta_j, b_i) = \frac{\exp(\theta_j - b_i)}{1+\exp(\theta_j - b_i)}$

We adapted to model $P_{ij}(\theta_j, b_i, a_i) = \frac{\exp(\theta_j - b_i)}{1+\exp(\theta_j - b_i)}$

$$\Rightarrow P(C_{ij}=1 \mid \theta_j, B_i, a_i) = \prod_{j=1}^{542}\prod_{i=1}^{1774}\left(\frac{\exp a_i(\theta_j - B_i)}{1+\exp a_i(\theta_j - B_i)}\right)^{C_{ij}} \cdot \left(1 - \frac{\exp a_i(\theta_j - B_i)}{1+\exp a_i(\theta_j - B_i)}\right)^{1-C_{ij}}$$

$$\Rightarrow \log P(C_{ij}=1 \mid \theta_j, B_i, a_i) = \log \prod_{j=1}^{542}\prod_{i=1}^{1774}\left(\frac{\exp a_i(\theta_j - B_i)}{1+\exp a_i(\theta_j - B_i)}\right)^{C_{ij}} \cdot \left(1 - \frac{\exp a_i(\theta_j - B_i)}{1+\exp a_i(\theta_j - B_i)}\right)^{1-C_{ij}}$$

$$\Rightarrow \sum_{j=1}^{542}\sum_{i=1}^{1774}\left[C_{ij}\log\left(\frac{\exp a_i(\theta_j - B_i)}{1+\exp a_i(\theta_j - B_i)}\right) + (1-C_{ij})\cdot\log\left(1 - \frac{\exp a_i(\theta_j - B_i)}{1+\exp a_i(\theta_j - B_i)}\right)\right]$$

Since $1 - \frac{x}{1+x} = \frac{1+x}{1+x} - \frac{x}{1+x} = \frac{1}{1+x} \Rightarrow 1 - \frac{\exp a_i(\theta_j - B_i)}{1+\exp a_i(\theta_j - B_i)} = \frac{1}{1+\exp a_i(\theta_j - B_i)}$

$$\Rightarrow \sum_{j=1}^{542}\sum_{i=1}^{1774}\left[C_{ij}\left[a_i(\theta_j - B_i) - \log(1+\exp a_i(\theta_j - B_i))\right] + (1-C_{ij})\cdot\left[\log 1 - \log(1+\exp a_i(\theta_j - B_i)\right]\right]$$

$$\Rightarrow \sum_{j=1}^{542}\sum_{i=1}^{1774}\left[C_{ij}\left[a_i(\theta_j - B_i) - \log(1+\exp a_i(\theta_j - B_i))\right] + C_{ij}\cdot\left(\log(1+\exp a_i(\theta_j - B_i) - \log(1+\exp a_i(\theta_j - B_i)\right)\right)$$

$$\Rightarrow \sum_{j=1}^{542}\sum_{i=1}^{1774} C_{ij}(a_i(\theta_j - B_i)) - \log(1+\exp a_i(\theta_j - B_i))$$

For updating θ, B, a. We need to find the derivatives $\frac{\partial \mathcal{L}}{\partial \theta_j}$, $\frac{\partial \mathcal{L}}{\partial B_i}$, $\frac{\partial \mathcal{L}}{\partial a_i}$

$$\frac{\partial \mathcal{L}}{\partial \theta_j} = \sum_{i=1}^{1774}\left(C_{ij}\cdot a_j - \frac{a_i \cdot \exp(\theta_j - B_i)}{1+\exp a_i(\theta_j - B_i)}\right)$$

$$\frac{\partial \mathcal{L}}{\partial B_i} = \sum_{j=1}^{542}\left(C_{ij}\cdot -a_i - \frac{a_i \cdot \exp(\theta_j - B_i)}{1+\exp a_i(\theta_j - B_i)}\right)$$

$$\frac{\partial \mathcal{L}}{\partial a_i} = \sum_{j=1}^{542}\left(C_{ij}(\theta_j - B_i) - \frac{(\theta_j - B_i)\cdot\exp(\theta_j - B_i)}{1+\exp a_i(\theta_j - B_i)}\right)$$

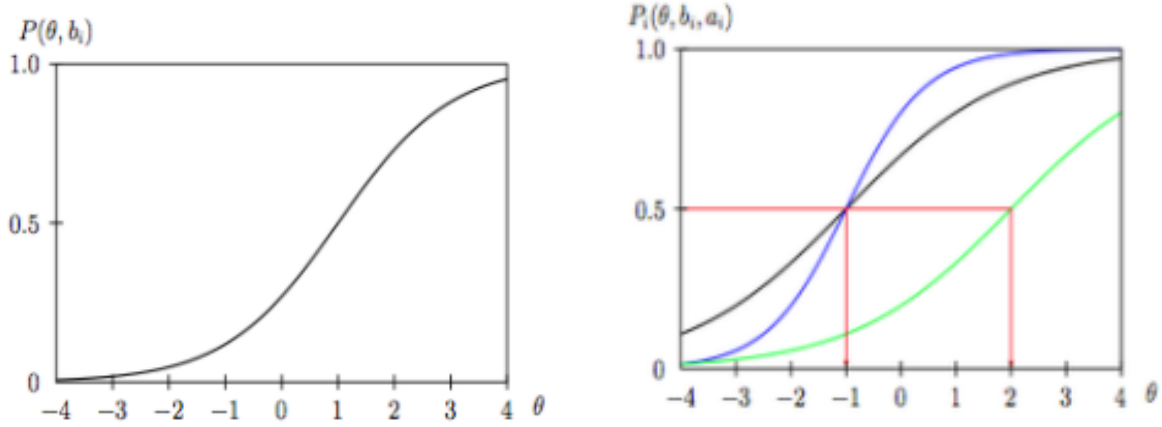
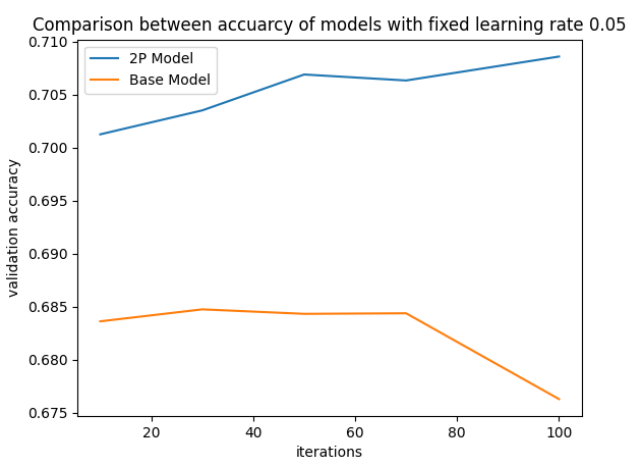


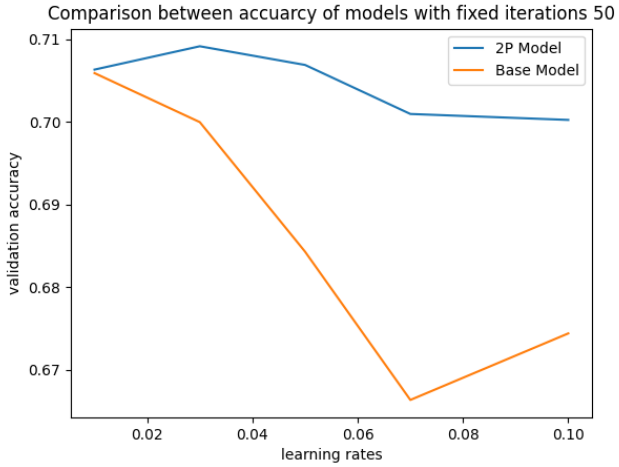
Figure 2: IRT without discrimination VS IRT Model with discrimination (by Public Health Columbia)

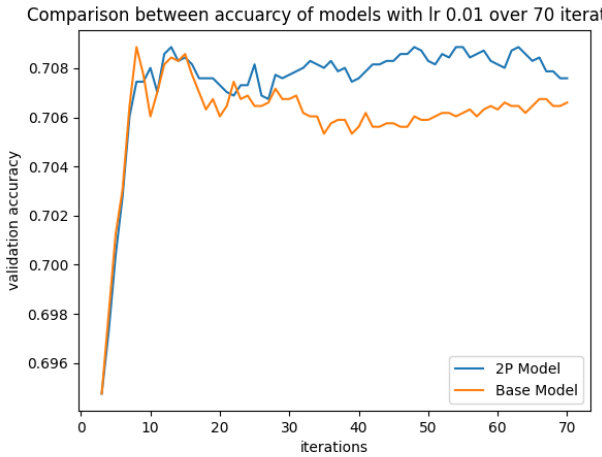
The figures from Public Health Columbia demonstrated the idea of 2 parameter IRT model's characteristic of having the discrimination parameter which can have multiple probability curves to distinguish between the difficulties of the questions instead that there is only one for original IRT model. For example, the blue curve on the right shows it has higher discrimination of the item.

3. Comparison or Demonstration

Model	Iteration	Learning Rate	Validation Accuracy
Baseline IRT	0.01	70	0.705898
2Parameter IRT	0.01	70	0.709144
Baseline IRT	0.01	30	0.706745
2Parameter IRT	0.01	30	0.707592
Baseline IRT	0.01	100	0.706181
2Parameter IRT	0.01	100	0.708580
Baseline IRT	0.03	30	0.701241
2Parameter IRT	0.03	30	0.703499
Baseline IRT	0.03	70	0.699971
2Parameter IRT	0.03	70	0.706322

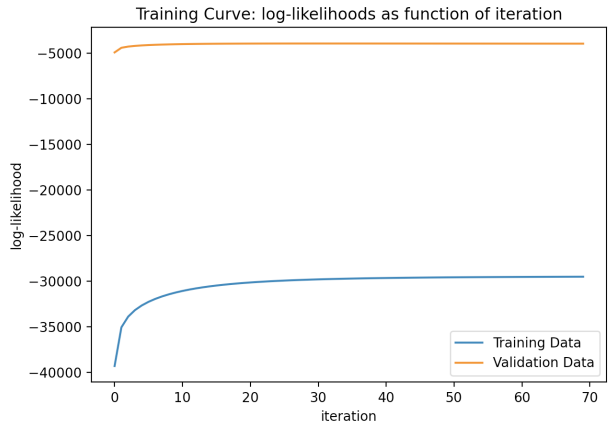


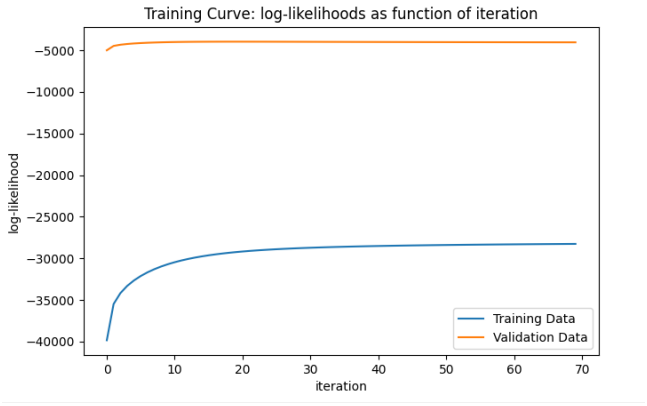




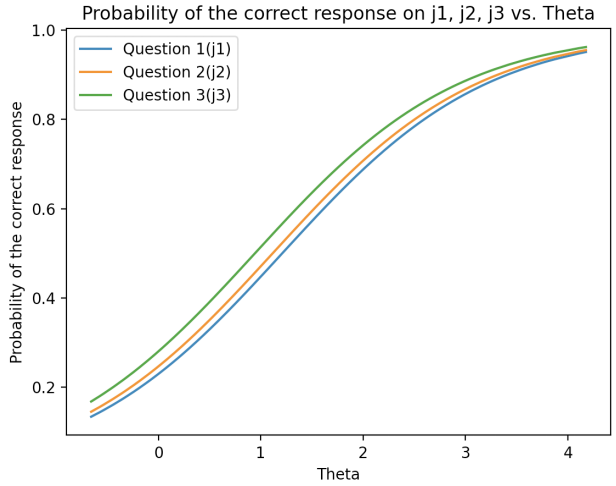
Above are tables and plots that shows the difference in validation accuracy between the base IRT model and 2 parameter IRT model, from the table, we included the top 5 hyperparamters that fit into the

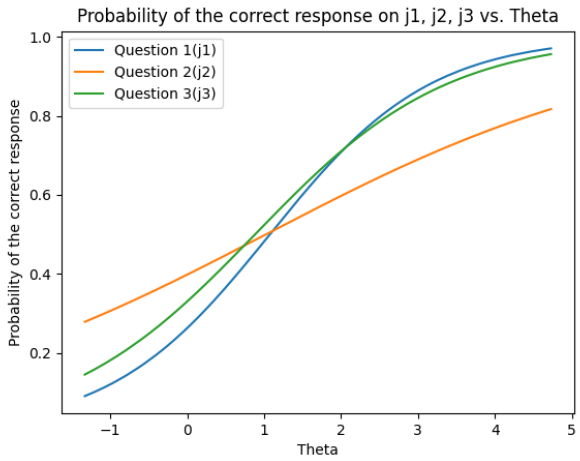
model to generate highest validation accuracies. We can see that the final accuracy are very similar between these two models, and by calculation, there is only a 1.2% increase in average validation accuracy of 2 paramter IRT model.





The graph on the right indicates the training curve of negative log-likelihood as function over 70 iterations, and the graph on the left is for the base IRT model. We can see that the training curve for 2 parameter IRT model converges more than the base model.





The above plots shows the comparison between probability of the correct response as a function of θ for a given question j. The right side is for 2 parameter IRT model, and base model is on the left. It is observed that with 2 parameter IRT model, the questions are more different to each other between each questions, representing the discrimination is effective.

4. Limitations

- We added a new discrimination parameter into the IRT model, since item response models uses gradient descent for optimization, choosing a good hyperparameter is important because too large or too small of learning rate would update the model poorly in respect to the chosen iteration. On the other hand, the number of iterations can not be too large to small either since we may not get the maximized log-likelihood then it would overfit the model if too many iterations or underfit if too few iterations. The limitation for the IRT models is that although we have tried 25

combinations of learning rates and iteration times to choose the relatively better hyperparamter, we may not get the best hyperparamter for optimization. However, trying too many potential hyperparameter is very time consuming and insufficient.

- We added a hidden layer to the neural network model for comparision. The result turns out that adding one hidden layer did not improve the algorithm. One possible reason is that since k may lead an overfitting on the model, and adding an extra hidden layer requires the hyperparameter to be tuned again. If we want to discover whether adding more than hidden layers can help to improve the model, further steps should be required.

- Recall that it was mentioned about splitting the dataset by criterias such as gender, age and financially disadvantages students, but the accuracy obtained about splitting by these criteria overfits the data, given the much higher training accuracy of 0.75 in average, but 0.69 validation and testing accuracy. It might due to we do not have enough sampling observations to begin with the splitting, as the observations in each category are much less than the original data which may lead to high varience.

- One possible extension could be by taking a deeper look to the meta datas, including questions and students information, we can possibly find a relationship between the students' information and the question performance. By using decision trees, we might be able to include the features and seperate the dataset by the combined crtieria instead of splitting the variables indivually. Additionally, larger sample size would also help to reduce varience lead by splitting data too deep.

- Another possible extension is inspired again by Public Health Columbia, which we can include the 4th parameter: gussing parameter that can decrease the information if the respondents is by guessing, so that it can restrict the odds of obtaining a correct response.

Contributions

Zewen Ma:
- Part A Question 3 4
- Part B neural network hidden layer part
- `neural_nerwork.py`
- `ensemble.py`
- `hidden_layer.py`

Yuchen Zeng:
- Part A Question 1 2
- Part B content discussion
- `knn.py`
- `item_responsese.py`

Haoze Huang:
- Part A question 2 debug
- Part B writeup
- `item_response_2p.py`
- `plot_diff.py`
- `split_by_age.py`
- `split_by_gender.py`

References

1. Morsy, Rothstein, L. M. R. R. (2015, June 10). Five Social Disadvantages That Depress Student Performance: Why Schools Alone Can't Close Achievement Gaps. Economic Policy Institute. Retrieved December 3, 2021, from *https* : *//www.epi.org/publication/five−social−disadvantages−that−depress−student−performance−why−schools−alone−cant−close−achievement−gaps/*

2. Hyde, J. S., Fennema, E., Lamon, S. J. (1990). Gender differences in mathematics performance: a meta-analysis. Psychological bulletin, 107(2), 139–155. *https* : *//doi.org*/10.1037/0033−2909.107.2.139

3. Item Response Theory. (n.d.). Columbia Public Health. Retrieved December 3, 2021, from https://www.publichealth.columbia.edu/research/population-health-methods/item-response-theory

4. Nagella, V. S. (2019, December 26). Cosine Similarity Vs Euclidean Distance - Vijaya Sasidhar Nagella. Medium. Retrieved December 3, 2021, from *https* : *//medium.com/@sasi*24/*cosine − similarity − vs − euclidean − distance − e*5*d*9*a*9375*fc*8

5. Prabhakaran, S. (2021, November 14). Cosine Similarity - Understanding the math and how it works? (with python). Machine Learning Plus. Retrieved December 3, 2021, from *https* : *//www.machinelearningplus.com/nlp/cosine − similarity/*

6. https://www.linkedin.com/pulse/cosine-similarity-classification-michael-lin/