CSC311 Final Project

Anny Runxuan Dai, Yansong Wang, Guanglei Zhu

December 4, 2021

1 Part A

1.1 K-Nearest Neighbor

(a) Here is the plot and report of the accuracy on the validation data as a function of k.

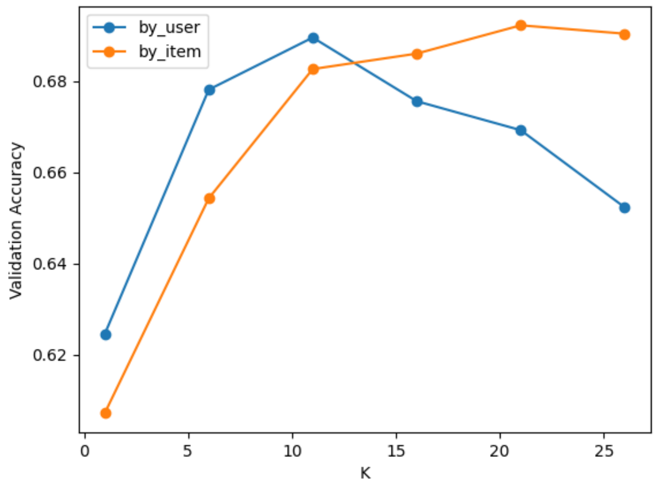


Figure 1: KNN plot

k	validation acc
1	0.6244707874682472
6	0.6780976573525261
11	0.6895286480383855
16	0.6755574372001129
21	0.6692068868190799
26	0.6522720858029918

(b) The highest performance $k^* = 11$ and its test accuracy is 0.684.

(c) The plot is already shown in figure 1, here are the accuracies on the validation data:

k	validation acc
1	0.607112616426757
6	0.6542478125882021
11	0.6826136042901496
16	0.6860005644933672
21	0.6922099915325995
26	0.69037538808919

The highest performance k^*= 21 and its test accuracy is 0.682

(d) Two potential limitations of KNN:

- Running time: all the computation happens at test time instead of training time which is undesired. For each data, need to calculate its distance to all the other data which takes huge amount of time and memory.
- Curse of high dimension: in high dimensional space, the $\ell 2$ distance does not perform well since all point have approximately equal distance.

1.2 Item Response Theory

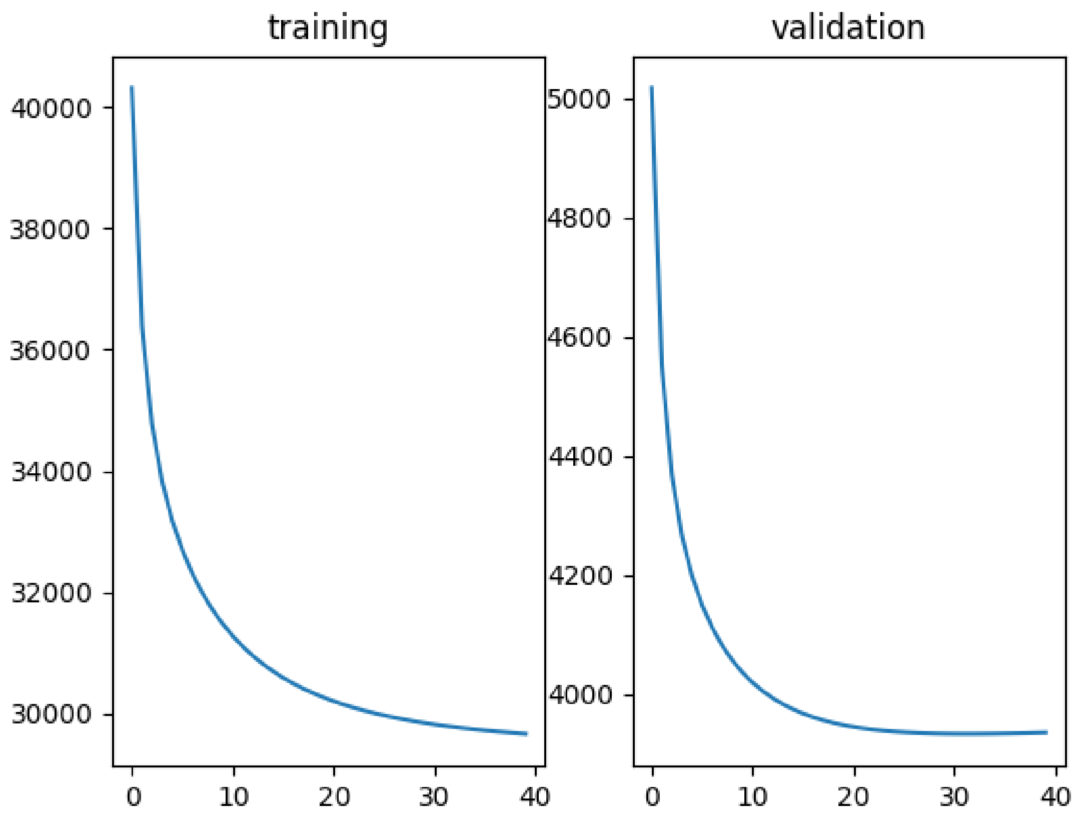
(a) The log-likelihood can be derived as the following:

$$\begin{aligned}
p(\mathbf{C} \,|\, \boldsymbol{\theta}, \boldsymbol{\beta}) &= \prod_{i,j} p(C_{ij} = 1 \,|\, \theta_i, \beta_j)^{C_{ij}} \cdot (1 - p(C_{ij} = 1 \,|\, \theta_i, \beta_j))^{1 - C_{ij}} \\
\ell(\boldsymbol{\theta}, \boldsymbol{\beta} \,|\, \mathbf{C}) &= \sum_{i,j} C_{ij} \log(p(C_{ij} = 1 \,|\, \theta_i, \beta_j) + (1 - C_{ij}) \log(1 - p(C_{ij} = 1 \,|\, \theta_i, \beta_j)) \\
&= \sum_{i,j} C_{ij} \log(\sigma(\theta_i - \beta_j)) + (1 - C_{ij}) \log(1 - \sigma(\theta_i - \beta_j))
\end{aligned}$$

note that σ is the sigmoid function, i.e. $\sigma(x) = \frac{e^x}{1+e^x}$, then differentiate wrst. θ_i, β_j

$$\begin{aligned}
\frac{d\ell}{d\theta_i} &= \sum_j C_{ij} \frac{1}{\sigma(\theta_i - \beta_j)} \cdot \sigma(\theta_i - \beta_j) \cdot (1 - \sigma(\theta_i - \beta_j)) \\
&+ (1 - C_{ij}) \frac{1}{1 - \sigma(\theta_i - \beta_j)} \cdot -(1 - \sigma(\theta_i - \beta_j)) \cdot (\sigma(\theta_i - \beta_j)) \\
&= \sum_j C_{ij} - C_{ij}\sigma(\theta_i - \beta_j) - \sigma(\theta_i - \beta_j) + C_{ij}\sigma(\theta_i - \beta_j) \\
&= \sum_j C_{i,j} - \sigma(\theta_i - \beta_j) \\
\frac{d\ell}{d\beta_j} &= \sum_i -C_{ij} \frac{1}{\sigma(\theta_i - \beta_j)} \cdot \sigma(\theta_i - \beta_j) \cdot (1 - \sigma(\theta_i - \beta_j)) \\
&- (1 - C_{ij}) \frac{1}{1 - \sigma(\theta_i - \beta_j)} \cdot -(1 - \sigma(\theta_i - \beta_j)) \cdot (\sigma(\theta_i - \beta_j)) \\
&= \sum_i -C_{ij} + C_{ij}\sigma(\theta_i - \beta_j) + \sigma(\theta_i - \beta_j) - C_{ij}\sigma(\theta_i - \beta_j) \\
&= \sum_i -C_{i,j} + \sigma(\theta_i - \beta_j)
\end{aligned}$$

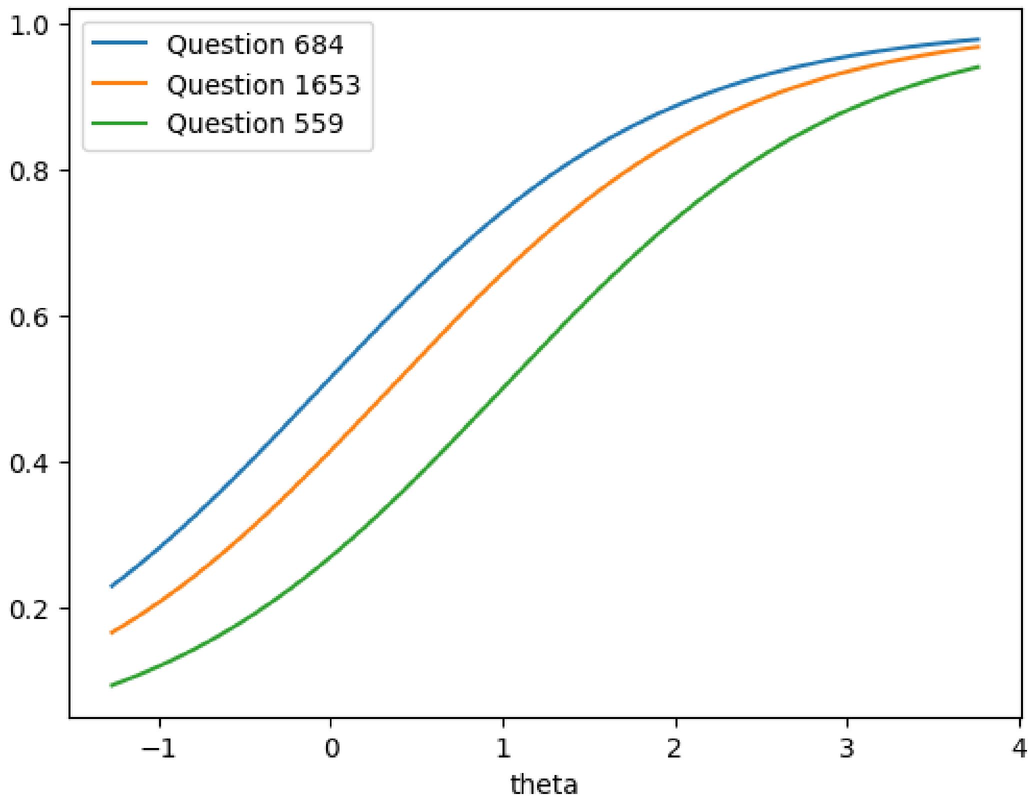
(b) We have chosen a learning rate of 0.01 with 40 iterations.



(note that the scale on the graphs are not the same for the training and the validation set.)

(c) Training Accuracy: 0.7394510302003952
Validation Accuracy: 0.7058989556872707
Test Accuracy: 0.7081569291560824

(d) We have randomly chosen Questions 684, 559, and 1653.



The shape of the curves looks very similar to a sigmoid function. The middle point represent the difficulty of the question. If a student's ability θ_i is greater than the middle point, it's guaranteed to correctly answer the question after thresholding. We can also see that question 684 is the easiest since the probability of solving it is always the highest.

1.3 Option 2: Neural Networks

(a) Describe differences between ALS and neural networks.

1.In neural networks, we update all the parameters at the same time by using matrix form, while ALS update one parameter and fixes others.

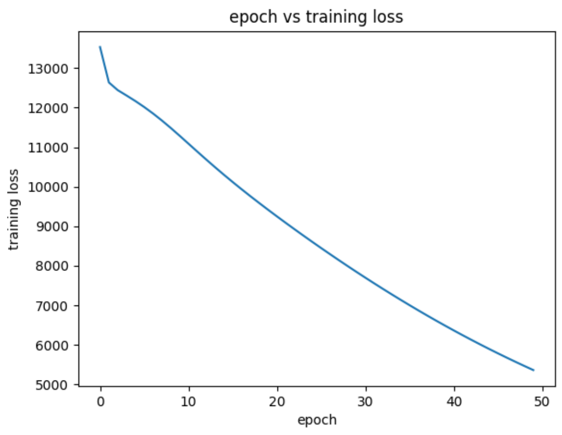
2.Alternating Least Square (ALS) is also a matrix factorization algorithm, but Neural Networks could be more complex.

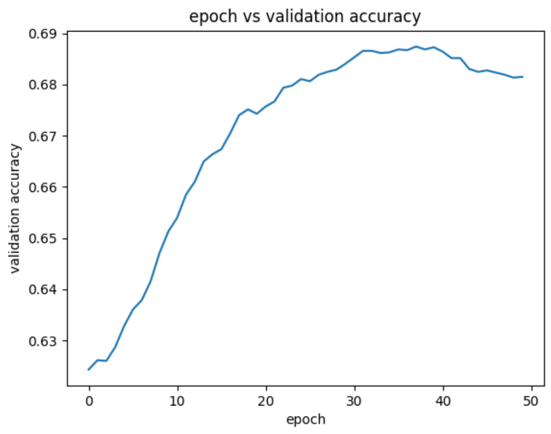
3.ALS is a linear regression, so there exist a optimal solution. But Neural Networks does not.

(b) Implement a class Auto-Encoder

(c) Train the auto-encoder using latent dimensions of $k \in 10, 50, 100, 200, 500$.
After we tried multiple different hyper-parameters, we found our highest validation accuracy: 0.687 with $k^* = 100$, number of epoch $= 38$, and learning rate $= 0.01$.

(d) training and validation objectives changes as a function of epoch





From graphs, we can tell the training loss decrease when the number of epoch increase. The validation accuracy increase when the number of epoch smaller than 37 and start to decrease after 37. The highest validation accuracy: 0.687. The final test accuracy: 0.679.

(e) L2 regularization

λ	validation acc	test acc
0.001	0.689	0.685
0.01	0.683	0.6827
0.1	0.6257	0.6264
1	0.6253	0.621

The highest validation accuracy and test accuracy we find are 0.689 & 0.685 with $\lambda = 0.001$, $k^* = 100$, number of epoch = 50 and learning rate = 0.05. Neural Networks with L2 is slightly better than it without L2.

1.4 Ensemble

We will use Item Response Theory along with alternating gradient descent for gradient decent as our base model.
For the bagging or bootstrap part, we will be using random sampling with replacement when choosing our training sampled data set. The size of the three data sets will be the same as the original training sets.
After we run the sampling three times with the different data sets we generated. We will be taking a vote based on the three results we generated. This means that if two of the results from different data set is 1, we will be setting the final result to 1. Then we will take the average of the validation accuracies and the test accuracies that we got using the result from training the model.
Our result is:
Validation Accuracy: 0.7033587355348575
Test Accuracy: 0.7027942421676545

We can see that the accuracies were about the same and even a little smaller than what we got without using the bootstrap method.
We think this happens because doing a bootstrap aggregation can make the variance decrease but the bias stays the same. Also, the new data sets that we generated were not independent. Thus, there are no improvements to the accuracies.

2 Part B

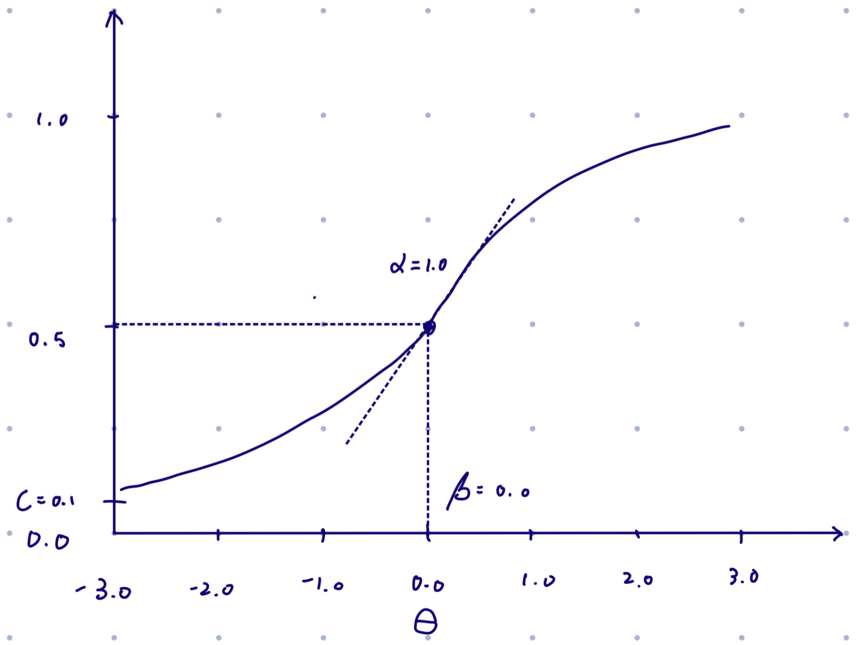
2.1 Formal Description

In this part, we will be modifying the one-parameter Item Response Theory that were implemented in part 1.2.
Instead of the one-parameter logistic model that we used in part 1.2, we will be using a three-parameter logistic model instead.
Also, we are intending to improve the optimization. For every 10 epoch, we will be decreasing the learning rate by half of its value. This can make the gradient descent approach faster to the best value and reduce overshooting.
The equation of the model will be:

$$\begin{aligned} p(C_{i,j} = 1 | \theta_i, \alpha_j, \beta_j, c_j) &= c_j + \frac{1 - c_j}{1 + \exp(-\alpha_j(\theta_i - \beta_j))} \\ &= c_j + \frac{(1 - c_j)\exp(\alpha_j(\theta_i - \beta_j))}{1 + \exp(\alpha_j(\theta_i - \beta_j))} \end{aligned} \tag{1}$$

In the above equation, θ_i represents the student's ability where with a higher value of θ the student perform better in the test and more likely to get a correct answer.
β represents the hardness of the question such that if β is higher, then the question is harder.
α represents the discrimination, scale, or slope of the probability distribution at β. This means that it will represent the easiness to answer the question. This means that if α is higher, the chance of being correct varies a lot wrst. student's ability.
c represents the probability of pseudo-guessing correct solution. In other word, it is the probability of guessing the question correct no matter how low the ability is. I.e. $p(-\infty) = c$, given α_j, β_j are fixed.

We can visualize it through a diagram:



To perform gradient descent, notice that the log-likelihood:

$$\begin{aligned} \ell(\theta, \alpha, \beta, c \,|\, \mathbf{C}) &= \log(p(\mathbf{C} \,|\, \theta, \alpha, \beta, c)) \\ &= \log(\prod_{i=1}^{542}\prod_{j=1}^{1774}(c_j + \frac{(1 - c_j)\exp(\alpha_j(\theta_i - \beta_j))}{1 + \exp(\alpha_j(\theta_i - \beta_j))})^{C_{i,j}} + (1 - c_j - \frac{(1 - c_j)\exp(\alpha_j(\theta_i - \beta_j))}{1 + \exp(\alpha_j(\theta_i - \beta_j))})^{1 - C_{i,j}} \\ &= \sum_{i=1}^{542}\sum_{j=1}^{1774}(C_{i,j}\log(c_j + \exp(\alpha_j(\theta_i - \beta_j))) + (1 - C_{i,j})\log(exp(\alpha_j(\theta_i - \beta_j)) \\ &+ c_j\exp(\alpha_j(\theta_i - \beta_j))) - \log(1 + \exp(\alpha_j(\theta_i - \beta_j)))) \end{aligned}$$

We also need to find the derivatives wrst. the variables to perform gradient descent.

$$\frac{d\ell}{d\theta_i} = \sum_j (\exp(\alpha_j(\theta_i - \beta_j)) \cdot \alpha_j)(\frac{C_{i,j}}{c_j + \exp(\alpha_j(\theta_i - \beta_j))} - \frac{1}{1 + \exp(\alpha_j(\theta_i - \beta_j))})$$
$$\frac{d\ell}{d\alpha_j} = \sum_i (\exp(\alpha_j(\theta_i - \beta_j)) \cdot (\theta_i - \beta_j))(\frac{C_{i,j}}{c_j + \exp(\alpha_j(\theta_i - \beta_j))} - \frac{1}{1 + \exp(\alpha_j(\theta_i - \beta_j))})$$
$$\frac{d\ell}{d\beta_j} = \sum_i (\exp(\alpha_j(\theta_i - \beta_j)) \cdot \alpha_j)(\frac{1}{1 + \exp(\alpha_j(\theta_i - \beta_j))} - \frac{C_{i,j}}{c_j + \exp(\alpha_j(\theta_i - \beta_j))})$$
$$\frac{d\ell}{dc_j} = \sum_i \frac{C_{i,j}}{c_j + \exp(\alpha_j(\theta_i - \beta_j))} - \frac{1 - C_{i,j}}{1 - c_j}$$

Algorithm Box:
The algorithm that updates the parameters of the probability equation.

Algorithm 1 *updateParams*$(datas, \alpha, \beta, c, \theta, learning\_rate)$

```
for data in datas do
    i ← user id of current data
    j ← question id of current data
    $\alpha[j] \leftarrow \alpha[j]$ + learning_rate * $\frac{d\ell}{d\alpha_j}$
    $\beta[j] \leftarrow \beta[j]$ + learning_rate * $\frac{d\ell}{d\beta_j}$
    $c[j] \leftarrow c[j]$ + learning_rate * $\frac{d\ell}{dc_j}$
    $\theta[i] \leftarrow \theta[j]$ + learning_rate * $\frac{d\ell}{d\theta_j}$
end for
return $\alpha, \beta, c, \theta$
```

The IRT algorithm to generate the final parameters

Algorithm 2 *IRT*$(data, validation\_data, learning\_rate, iteration)$

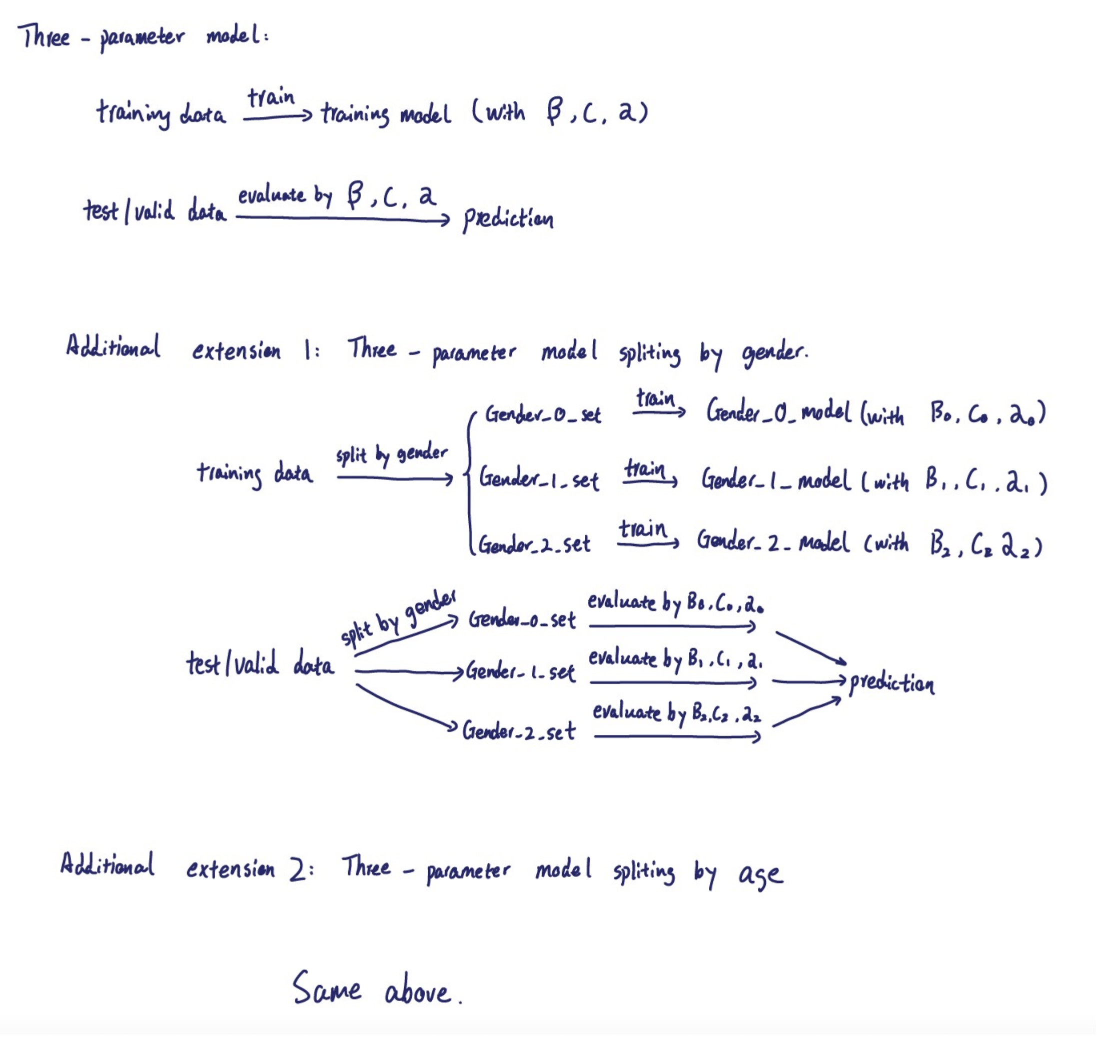
```
$\alpha$ ← a list of 1774 ones
$\beta$ ← a list of 1774 zeros
$c$ ← a list of 1774 0.1s
$\theta$ ← a list of 542 zeros
val_acc_lst ← []
for i in iteration do
    $\alpha, \beta, c, \theta$ ← updateParams(data, $\alpha, \beta, c, \theta$, learning_rate)
    append evaluate(validation_data, $\alpha, \beta, c, \theta$) to val_acc_lst
end for
return $\alpha, \beta, c, \theta$, val_acc_lst
```

The algorithm to evaluate the accuracy from the data

Algorithm 3 *evaluate*$(datas, \alpha, \beta, c, \theta)$

```
result ← 0
for data in datas do
    if the probability get using the parameters is larger than 0.5 then
        pred ← 1
    else
        pred ← 0
    end if
    if pred = real result then
        result ← result + 1
    end if
end for
return result / number of data
```

2.2 Figure or Diagram



In three parameters, we train three parameters $\boldsymbol{\beta}$,$\boldsymbol{\alpha}$ and $\mathbf{c}$, instead of only train β in parameters.
For all details of implementations are in improve_irt.py

2.3 Comparison or Demonstration

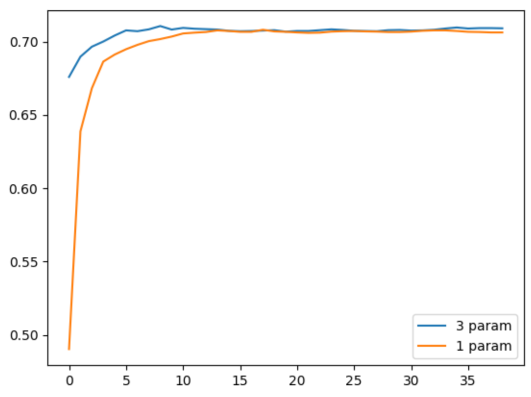
Train three-parameter($\boldsymbol{\beta}$,$\boldsymbol{\alpha}$ and c) model:
In one parameter IRT, we have a β representing the question's difficulty. But there is some limitation:

1. If the question is extremely hard, and a student with very low ability, then the student will never get correct on this question. In general, this student should always have a chance to get correct by guessing.

2. Some questions are hard to answer, so students with high ability can perform much better than low ability students. Which implies the accuracy of discrimination is higher others, but all discrimination is the same in this model.

So we came up with more parameters to predict each student's answer for each question since the difficulty may not be the only factor. Then we decide to add two parameters, $\mathbf{c}$(chance to guess the correct answer) and $\boldsymbol{\alpha}$(easiness to answer the question). Here we have our three-parameter IRT:

After we run our code with the three-parameter IRT, we got the following results:

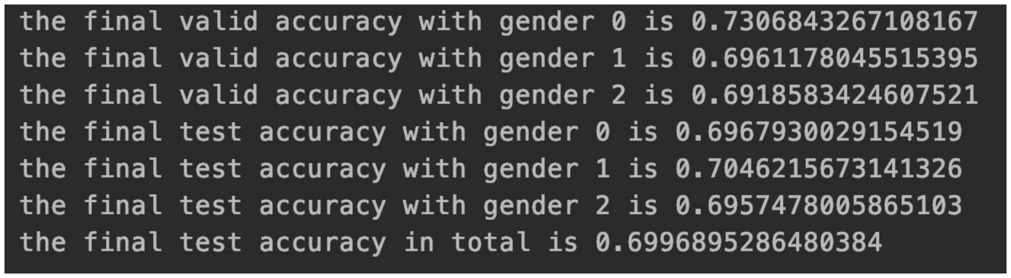


In the Above figure, the x axis represents the number of iterations and the y axis represents the accuracy.

	one-param IRT	three-param IRT
Final Validation Accuracy	0.7033587355348575	0.7090036692068868
Final Test Accuracy	0.7027942421676545	0.7056167090036692

We can see that the validation accuracy improved by about 0.5% and the test accuracy improved by about 0.3%. By introducing parameter c, the accuracy for the initial iteration is significantly higher. The optimization process also converges faster due to introduction of learning rate decay. Note that in actual training, we fixed $\boldsymbol{c}$ as a hyper parameter (around 15-20%) because of numerical instability.

Train three-parameter($\boldsymbol{\beta}$,$\boldsymbol{\alpha}$ and c) by splitting group:
First attempt: We use the hold data set to train the hardness of the question(β), discrimination(α), and probability of pseudo-guessing(C) parameters on the previous page. This section trained that three-parameter based on the different gender groups (gender = 0, 1, 2) and performed validation/test set on a different model split by gender. As a result:

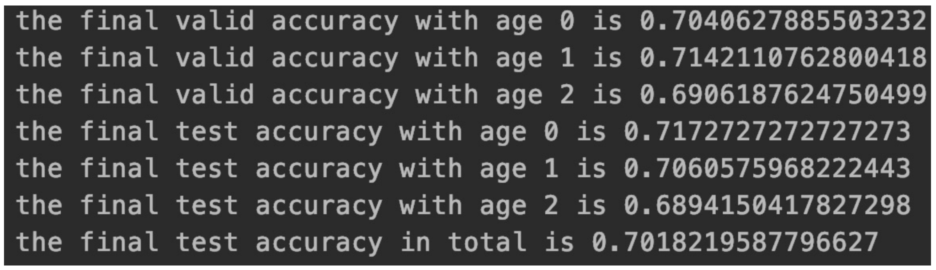


```
the final valid accuracy with gender 0 is 0.7306843267108167
the final valid accuracy with gender 1 is 0.6961178045515395
the final valid accuracy with gender 2 is 0.6918583424607521
the final test accuracy with gender 0 is 0.6967930029154519
the final test accuracy with gender 1 is 0.7046215673141326
the final test accuracy with gender 2 is 0.6957478005865103
the final test accuracy in total is 0.6996895286480384
```

The final test accuracy in total is 0.06996 by gender split, which did not improve our original model. List of reasons that may cause this result :

1. There is no correlation between performance and gender.
2. After we split data, each model will have fewer data set to train then they may not fit very well.

Second attempt: We tried split data set by age, where age 0 = unspecified, age 1 = age under 14, and age 2 = age higher than 14:



```
the final valid accuracy with age 0 is 0.7040627885503232
the final valid accuracy with age 1 is 0.7142110762800418
the final valid accuracy with age 2 is 0.6906187624750499
the final test accuracy with age 0 is 0.7172727272727273
the final test accuracy with age 1 is 0.7060575968222443
the final test accuracy with age 2 is 0.6894150417827298
the final test accuracy in total is 0.7018219587796627
```

It was also worse than the model before the split due to similar reasons.
In general, we would choose to use the original three-parameter instead of split by gender our split by age in this case.

2.4 Limitations

In summary, We improve the overall performance by adding two parameters on one-parameter IRT, which mean the easiness to learn(discrimination) and guessing will affect the final prediction.

We also tried to separate the data using the gender or the age of the students. However, the result did not came out as what we've expected. The reason that this happened might be the size of the data was not large enough after separated. Also, out assumption that age or gender of a student might influence the ability of the student might be wrong. However, if these variable have a large influence, our model might not be accurate.

Our model did not consider of the randomness of the student's ability. Notice that in real life situations, a student can have different ability in different dimensions. Also, there might be some randomness when the student answer the questions. Thus, our model might not perform well when the ability of a student is very extinct for different dimension. For example, if a student is very good at logical thinking but bad at algebra. Since our model is evaluating the student's overall ability, it might predict that the student did the logical thinking question wrong and the algebra question correct.

In our model and what we've tried, we did not use the subject meta. Thus, if the difficulty of a question or the ability of a student is strongly correlated with the subject meta, our prediction might be wrong.