CSC311 Project

Guo Qing Huai (1004916120)
Jason Li (1005787089)
Ananya Jha (1005667306)

December 2022

1. k-Nearest Neighbor

(a)

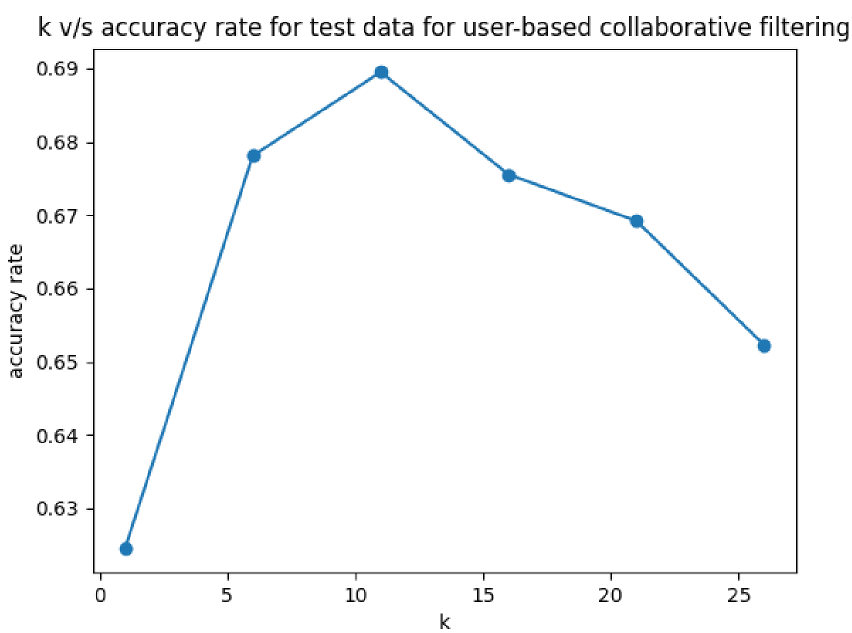


Figure 1

(b)

The chosen k* was k* = 11 with 68.42 percent test accuracy.

(c)

The assumption for item-based filtering: If question 1 is correctly/incorrectly answered by a set of students A, question 2 will be similarly correctly/incorrectly answered by the same set of students.

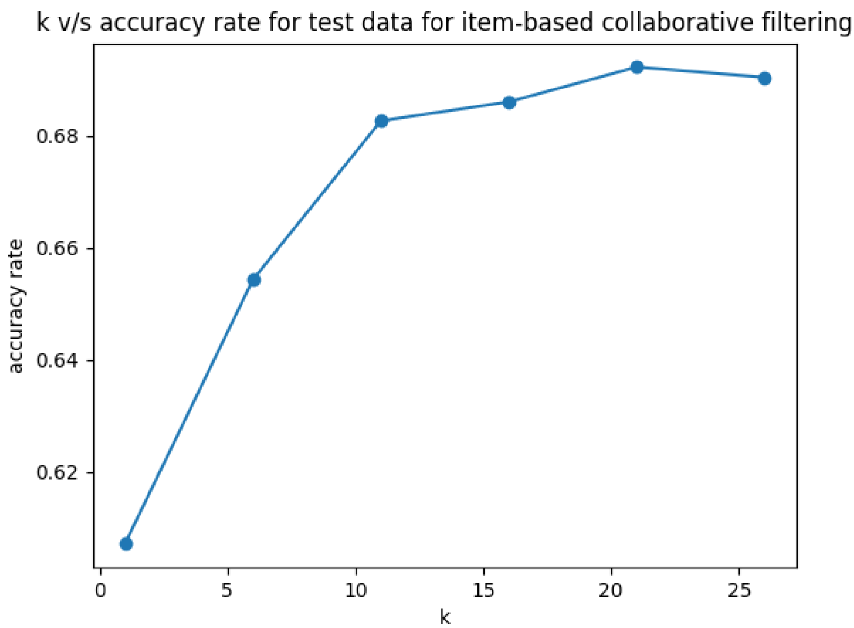


Figure 2

(d)

User-based filtering performs slightly better:

```
Test performance for user based:  0.6841659610499576
Test performance for item based:  0.6816257408975445
```

Figure 3

(e)

- Our assumption for user-based collaborative filtering does not necessarily hold true. For example, it is possible that students A and B have the same correct/incorrect answers on diagnostic questions as B but for the question that we're predicting, student A has prepared more compared to B and so performs better.
- Similarly, our second assumption for item-based collaborative filtering as well does not necessarily hold true. For example, it is possible that a set of questions have been similarly answered by a group of students but a different set of questions have not been similarly answered by the same group of students.
- Knn is also very sensitive to outliers or missing data. In our case, we do have a lot of missing data so it is bound to impact the performance of our knn algorithm.
- Some limitations of knn in general hold as well- computational cost, the curse of dimensionality in higher dimensions, and storage issues.

2. Item Response Theory

(a)

We assume that the $c_{i,j}$ are independent of each other. We define the sigmoid function as

$$\sigma(z) = \frac{e^z}{1+e^z}$$

for all values $z \in \mathbb{R}$. We can then derive the log likelihood as follows,

$$\begin{aligned}
\ell(\theta, \beta) &= \log(p(\mathbf{C})) \\
&= \log(p(\prod_{i,j} c_{i,j} \mid \theta_i, \beta_j)) \\
&= \log(\prod_{i,j} p(c_{i,j} \mid \theta_i, \beta_j)) \\
&= \log(\prod_{i,j:c_{i,j}=0} p(c_{i,j} \mid \theta_i, \beta_j)) \log(\prod_{i,j:c_{i,j}=1} p(c_{i,j} \mid \theta_i, \beta_j)) \\
&= \sum_{i,j:c_{i,j}=0} \log(p(c_{i,j} \mid \theta_i, \beta_j)) + \sum_{i,j:c_{i,j}=1} \log(p(c_{i,j} \mid \theta_i, \beta_j)) \\
&= \sum_{i,j:c_{i,j}=0} \log(1 - \sigma(\theta_i - \beta_j)) + \sum_{i,j:c_{i,j}=1} \log(\sigma(\theta_i - \beta_j))
\end{aligned}$$

For the following derivation we will use the fact that

$$\frac{d}{dz}\sigma(z) = \sigma(z)(1-\sigma(z))$$

Then by chain rule and linearity of the derivative,

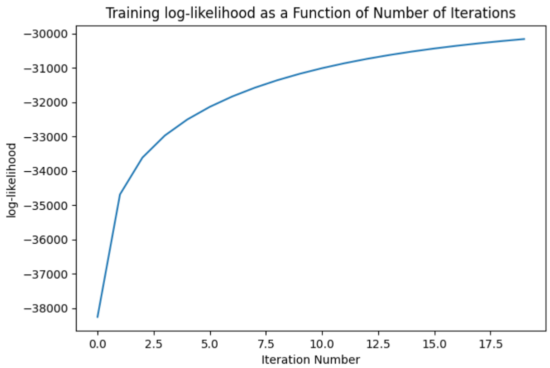
$$\frac{\partial \ell}{\partial \theta_i} = \sum_{\substack{j\text{:student } i \text{ answers} \\ \text{question } j \text{ correctly}}} (1 - \sigma(\theta_i - \beta_j)) - \sum_{\substack{j\text{:student } i \text{ answers} \\ \text{question } j \text{ incorrectly}}} \sigma(\theta_i - \beta_j)$$

$$\frac{\partial \ell}{\partial \beta_j} = -\sum_{\substack{i\text{:student } i \text{ answers} \\ \text{question } j \text{ correctly}}} (1 - \sigma(\theta_i - \beta_j)) + \sum_{\substack{i\text{:student } i \text{ answers} \\ \text{question } j \text{ incorrectly}}} \sigma(\theta_i - \beta_j)$$

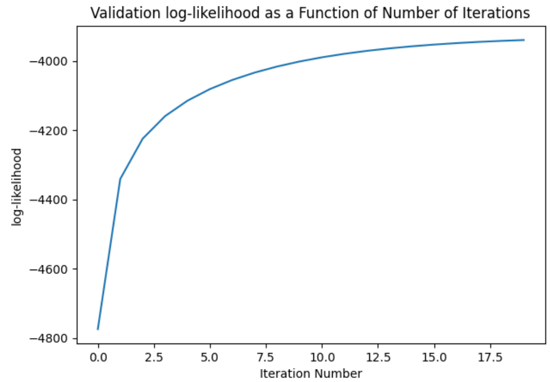
For any of the above sums, if the value is not reported in the matrix then it is simply excluded in the sum. In other words, we simply ignore the cases when $c_{i,j}$ is not a number.

(b)

My tuned hyperparamters are a learning rate of $\alpha = 0.01$ with 20 total iterations.



(a) training log-likelihood



(b) validation log-likelihood

Figure 4

(c)

As reported by my code:

```
The final validation accuracy is 0.7064634490544736
The final testing accuracy is 0.7042054755856618
```

(d)

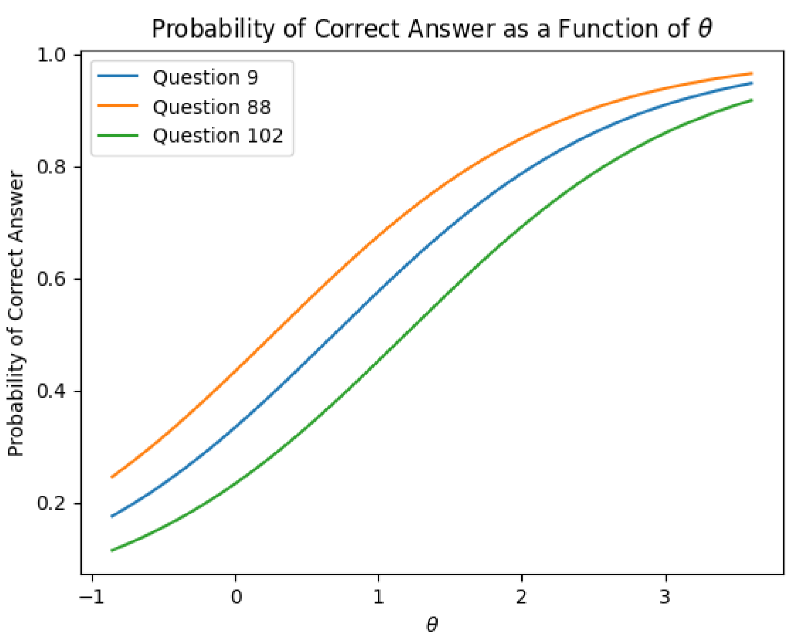


Figure 5

The curves represent the probability of a certain question being answered correctly as a function of student ability measured by variable θ.
The curves are increasing as expected because increased student ability should mean increased probability of the question being answered correctly. The curves also have an 'S' shape because the probability of a correct answer is calculated by a sigmoid function in the variable θ with some horizontal translation. Hence, we can see that our curves match the shape of $\sigma(z) = \frac{e^z}{1+e^z}$.

3. Neural Networks

(a)

Supervised learning vs unsupervised learning:
Neural networks is supervised learning method where we train the model on a large labeled dataset. While for ALS, it usually works on unsupervised learning cases where it generates predictions based on matrix decomposition.

Parameter Updates:
During the training process of neural network, it updates all of its parameters during one back propagation pass. While updating the ALS parameters requires splitting the parameters into $\boldsymbol{Z}$ and $\boldsymbol{U}$, fixing the parameters $\boldsymbol{Z}$, and update the other $\boldsymbol{U}$. Therefore instead of updating all parameters, ALS alternately updates $\boldsymbol{Z}$ and $\boldsymbol{U}$.

Linear vs non-linear:
The activation functions in neural networks creates such a non-linearity in the model, otherwise a multi-layer model can just be reduced to a single layer. While for ALS, it is a form of linear regression problem, so it is a linear method.

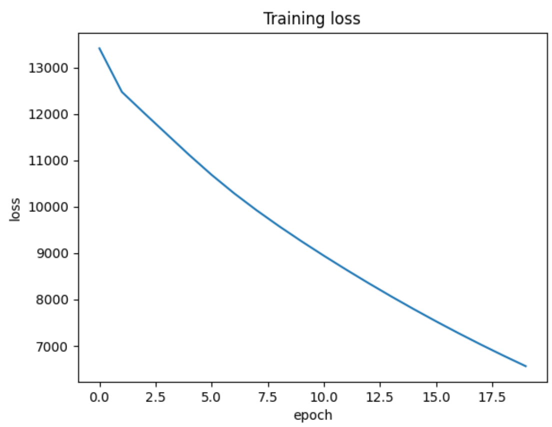
(b)

see part_a/neural_network.py

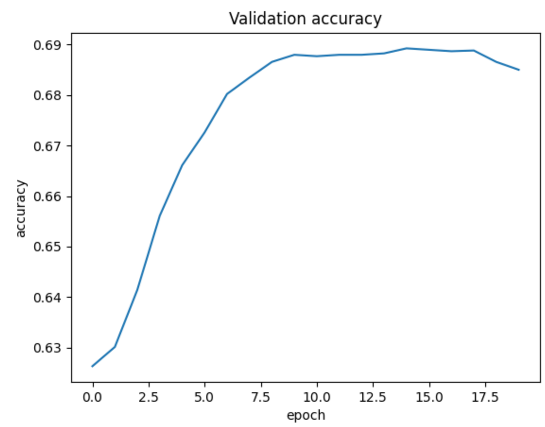
(c)

First we have tried with num_epoch = 10 and learning rates 0.01, 0.05, 0.09, the accuracy reached its highest at 0.6814 when k = 10 and learning rate = 0.09.
So we increased the number of epochs and decreased the learning rate to see if we can get a better accuracy.
Then we tried epoch = 20, 30, 60 and learning rates ranges randomly from 0.001 to 0.05, and we realized that with a lower number of epoch (20) and a slightly higher learning rate at 0.03, it gives the best result with a validation accuracy of 0.6881 at k=50. Also when we have a large number of epoch (60) and a low learning rate of 0.01, it also gives promising validation accuracy(0.6854), which is slightly lower than the value given before.
Here are the best parameter settings we have tested out:
$k^* = 50, num\_epoch = 20, lr = 0.03$

(d)



(a) Training loss vs number of iteration



(b) Validation accuracy vs number of iteration

Figure 6

The training cost decreases as the number of iteration increases. The validation accuracy increases rapidly in the first few iterations, and then the curve converges to a plateau at around 0.68. After the 17th iteration, it starts to decrease by a tiny bit due to overfitting. Below is the test accuracy:

Figure 7: Final test accuracy

(e)

We have implemented the regularization and tested the model with λ equal to $[0.001, 0.01, 0.1, 1]$. When $\lambda = 0.001$, it gives the best accuracy.

```
Epoch: 0        Training Cost: 13569.780998      Valid Acc: 0.6236240474174428
Epoch: 1        Training Cost: 12659.792113      Valid Acc: 0.6302568444820773
Epoch: 2        Training Cost: 12240.413792      Valid Acc: 0.6425345752187411
Epoch: 3        Training Cost: 11832.386926      Valid Acc: 0.6507197290431838
Epoch: 4        Training Cost: 11427.229522      Valid Acc: 0.659046006209427
Epoch: 5        Training Cost: 11044.978637      Valid Acc: 0.6679367767428732
Epoch: 6        Training Cost: 10699.144370      Valid Acc: 0.6709003669206887
Epoch: 7        Training Cost: 10389.262279      Valid Acc: 0.6766864239345187
Epoch: 8        Training Cost: 10107.096922      Valid Acc: 0.678662150719729
Epoch: 9        Training Cost: 9845.164623       Valid Acc: 0.6806378775049393
Epoch: 10       Training Cost: 9598.923623       Valid Acc: 0.6820491109229466
Epoch: 11       Training Cost: 9365.824176       Valid Acc: 0.6837425910245555
Epoch: 12       Training Cost: 9144.590083       Valid Acc: 0.6855771944679651
Epoch: 13       Training Cost: 8934.470662       Valid Acc: 0.6861416878351679
Epoch: 14       Training Cost: 8734.830677       Valid Acc: 0.6851538244425628
Epoch: 15       Training Cost: 8545.179783       Valid Acc: 0.6847304544171606
Epoch: 16       Training Cost: 8365.145471       Valid Acc: 0.6838837143663562
Epoch: 17       Training Cost: 8194.491448       Valid Acc: 0.6840248377081569
Epoch: 18       Training Cost: 8032.945905       Valid Acc: 0.682895850973751
Epoch: 19       Training Cost: 7880.157322       Valid Acc: 0.683460344340954
```

(a) $\lambda = 0.001$ regularization

```
Epoch: 0        Training Cost: 14858.663376      Valid Acc: 0.6281399943550663
Epoch: 1        Training Cost: 13753.551546      Valid Acc: 0.6326559412926898
Epoch: 2        Training Cost: 13261.192832      Valid Acc: 0.6385831216483207
Epoch: 3        Training Cost: 12885.787784      Valid Acc: 0.647897262207169
Epoch: 4        Training Cost: 12588.895977      Valid Acc: 0.6539655659046006
Epoch: 5        Training Cost: 12356.203448      Valid Acc: 0.6600338696020321
Epoch: 6        Training Cost: 12178.546335      Valid Acc: 0.6661021732994638
Epoch: 7        Training Cost: 12045.738456      Valid Acc: 0.6668077900084673
Epoch: 8        Training Cost: 11946.781512      Valid Acc: 0.6713237369460909
Epoch: 9        Training Cost: 11872.043348      Valid Acc: 0.67499294383291
Epoch: 10       Training Cost: 11814.375363      Valid Acc: 0.677109793959921
Epoch: 11       Training Cost: 11768.828923      Valid Acc: 0.6775331639853232
Epoch: 12       Training Cost: 11732.032513      Valid Acc: 0.6788032740615297
Epoch: 13       Training Cost: 11701.670592      Valid Acc: 0.6790855207451313
Epoch: 14       Training Cost: 11676.138619      Valid Acc: 0.6788032740615297
Epoch: 15       Training Cost: 11654.315120      Valid Acc: 0.6806378775049393
Epoch: 16       Training Cost: 11635.402915      Valid Acc: 0.6814846175557437
Epoch: 17       Training Cost: 11618.819156      Valid Acc: 0.6809201241885408
Epoch: 18       Training Cost: 11604.126361      Valid Acc: 0.68077900084674
Epoch: 19       Training Cost: 11590.988417      Valid Acc: 0.6804967541631386
```

(b) $\lambda = 0.01$ regularization

```
Epoch: 0        Training Cost: 18685.154209      Valid Acc: 0.6223539373412362
Epoch: 1        Training Cost: 13758.063638      Valid Acc: 0.6229184307084392
Epoch: 2        Training Cost: 13512.151971      Valid Acc: 0.6257408975444538
Epoch: 3        Training Cost: 13442.325893      Valid Acc: 0.6255997742026531
Epoch: 4        Training Cost: 13382.934376      Valid Acc: 0.625176404177251
Epoch: 5        Training Cost: 13326.475326      Valid Acc: 0.6261642675698561
Epoch: 6        Training Cost: 13272.104242      Valid Acc: 0.6257408975444538
Epoch: 7        Training Cost: 13219.423650      Valid Acc: 0.6254586508608524
Epoch: 8        Training Cost: 13168.150530      Valid Acc: 0.6265876375952583
Epoch: 9        Training Cost: 13118.068421      Valid Acc: 0.6250352808354502
Epoch: 10       Training Cost: 13069.009687      Valid Acc: 0.6250352808354502
Epoch: 11       Training Cost: 13020.853475      Valid Acc: 0.6237651707592435
Epoch: 12       Training Cost: 12973.534213      Valid Acc: 0.622495060683037
Epoch: 13       Training Cost: 12927.064002      Valid Acc: 0.6232006773920407
Epoch: 14       Training Cost: 12881.566349      Valid Acc: 0.622495060683037
Epoch: 15       Training Cost: 12837.309204      Valid Acc: 0.6213660739486311
Epoch: 16       Training Cost: 12794.716702      Valid Acc: 0.6232006773920407
Epoch: 17       Training Cost: 12754.321407      Valid Acc: 0.6227773073666384
Epoch: 18       Training Cost: 12716.636669      Valid Acc: 0.6236240474174428
Epoch: 19       Training Cost: 12682.003728      Valid Acc: 0.6240474174428451
```

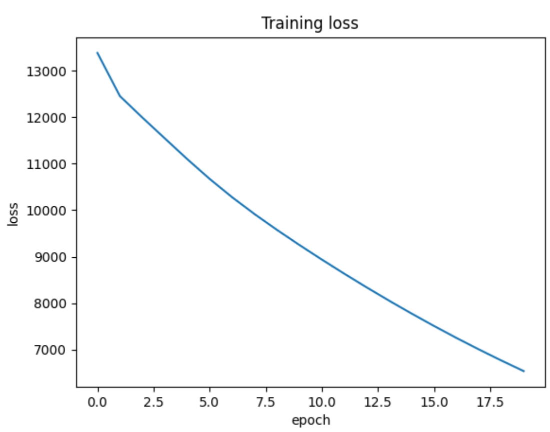
(c) $\lambda = 0.1$ regularization

```
Epoch: 0        Training Cost: 19387.090334      Valid Acc: 0.5777589613322044
Epoch: 1        Training Cost: 13945.894320      Valid Acc: 0.6054191363251482
Epoch: 2        Training Cost: 13703.984192      Valid Acc: 0.6090883432119673
Epoch: 3        Training Cost: 13511.115146      Valid Acc: 0.6130397967823878
Epoch: 4        Training Cost: 13355.747422      Valid Acc: 0.6136042901495907
Epoch: 5        Training Cost: 13229.303974      Valid Acc: 0.617132373694609
Epoch: 6        Training Cost: 13125.389679      Valid Acc: 0.6181202370872142
Epoch: 7        Training Cost: 13039.198568      Valid Acc: 0.6184024837708157
Epoch: 8        Training Cost: 12967.084690      Valid Acc: 0.6199548405306238
Epoch: 9        Training Cost: 12906.253152      Valid Acc: 0.6220716906576348
Epoch: 10       Training Cost: 12854.541264      Valid Acc: 0.6223539373412362
Epoch: 11       Training Cost: 12810.259507      Valid Acc: 0.6227773073666384
Epoch: 12       Training Cost: 12772.076138      Valid Acc: 0.6217894439740334
Epoch: 13       Training Cost: 12738.933314      Valid Acc: 0.6212249506068304
Epoch: 14       Training Cost: 12709.983654      Valid Acc: 0.6220716906576348
Epoch: 15       Training Cost: 12684.544479      Valid Acc: 0.6222128139994355
Epoch: 16       Training Cost: 12662.061514      Valid Acc: 0.6226361840248377
Epoch: 17       Training Cost: 12642.082052      Valid Acc: 0.6222128139994355
Epoch: 18       Training Cost: 12624.234342      Valid Acc: 0.6234829240756421
Epoch: 19       Training Cost: 12608.211138      Valid Acc: 0.6234829240756421
```

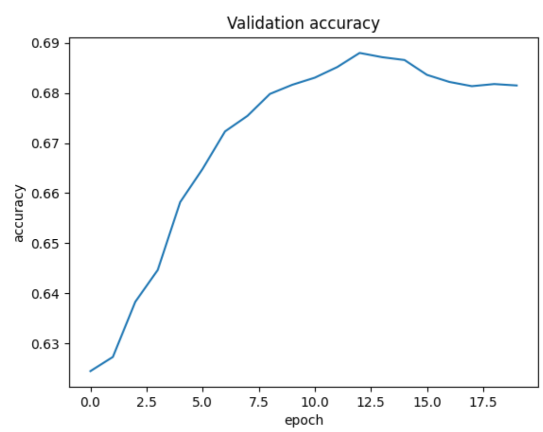
(d) $\lambda = 1$ regularization

Figure 8: Model with regularization

Here are the best parameter settings we have tested out:
$k^* = 50, num\_epoch = 20, lr = 0.03, lamb = 0.001$



(a) Training loss vs number of iteration



(b) Validation accuracy vs number of iteration

Figure 9: Model with regularization

```
Test acc: 0.6886819079875811
```

Figure 10: Final test accuracy

Based on the final test accuracy, the model with regularization penalty is slightly better than the model without regularization.

4. Ensemble

```
Validation accuracy: 0.7036409822184589
Testing accuracy: 0.7011007620660458
```

Figure 11: Ensemble final accuracy

First we imported the training, validation and test data, and sampled three training data with replacement from the original training data. The three bootstraps have the batch size same as the original training dataset. The base models we selected are three IRT models with learning rates: 0.005, 0.01, 0.015, and iterations 30, 30, 30 respectively. We trained the three IRT models using the three bootstrapped datasets generated previously, and used the trained beta and theta values to make predictions. For each model separately, ff the probability is greater or equal to 0.5, the prediction is considered a value 1, otherwise, it will be stored as 0. To predict the correctness, we took the generated 3 predictions and averaged the predicted correctness.
The predicted results based on the ensemble process does not give better results. The ensemble algorithm reduces variance by bootstrapping the train data and averaging the results, but it does not reduce the bias.

Part B: Modified IRT

1.

We will try to makes adjustments to the item response theory model introduced in part A to improve under-fitting. The IRT model introduced in part A is probabilistic model that tries to maximize a likelihood function. However, question subject is a data set available to us not being utilized. We can use this metadata to form some hypothesises to try and improve the performance of our model. We will assume students preform better in certain subjects better than others. Hence, we can assign a subject specific performance value to each student. We will also assume that if a subject is covered in a question, it is equal to all other subjects covered in that question i.e. each subject is equally weighted in the case of a question with multiple subjects. This is to make our model simpler.
So,

$$\theta = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} & \dots & \theta_{1M} \\ \theta_{21} & \theta_{22} & \theta_{23} & \dots & \theta_{2M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_{N1} & \theta_{N2} & \theta_{N3} & \dots & \theta_{NM} \end{bmatrix}$$

Where $\theta_{i,k}$ is student i's performance in subject k, N is the total number of students, and M is the total number of subjects.
We define,

$$p(c_{ijk} = 1 \mid \theta_{i,k}, \beta_j) = \frac{\exp(\theta_{i,k} - \beta_j)}{1 + \exp(\theta_{i,k} - \beta_j)}$$

Similarly to the question 2 from the last part, we can derive the log-likelihood $\log p(\mathbf{C} \mid \theta, \beta)$ and derivatives.

$$\ell(\theta, \beta) = \sum_{i,j,k:c_{i,j}=0} \log(1 - \sigma(\theta_{i,k} - \beta_j)) + \sum_{i,j,k:c_{i,j}=1} \log(\sigma(\theta_{i,k} - \beta_j))$$

$$\frac{\partial \ell}{\partial \theta_{i,k}} = \sum_{\substack{j\text{:student } i \text{ answers} \\ \text{question } j \text{ correctly} \\ \text{in subject } k}} (1 - \sigma(\theta_{i,k} - \beta_j)) - \sum_{\substack{j\text{:student } i \text{ answers} \\ \text{question } j \text{ incorrectly} \\ \text{in subject } k}} \sigma(\theta_{i,k} - \beta_j)$$

$$\frac{\partial \ell}{\partial \beta_j} = - \sum_{\substack{i\text{:student } i \text{ answers} \\ \text{question } j \text{ correctly} \\ \text{in subject } k}} (1 - \sigma(\theta_{i,k} - \beta_j)) + \sum_{\substack{i\text{:student } i \text{ answers} \\ \text{question } j \text{ incorrectly} \\ \text{in subject } k}} \sigma(\theta_{i,k} - \beta_j)$$

Finally, we define the probability that the question j with subjects $\mathbf{k}$ is correctly answered by student i as

$$p(c_{ij\mathbf{k}} = 1 \mid \theta_{ik}, \beta_j) = \operatorname*{mean}_k (p(c_{ijk} = 1 \mid \theta_{i,k}, \beta_j))$$

2. Model diagram

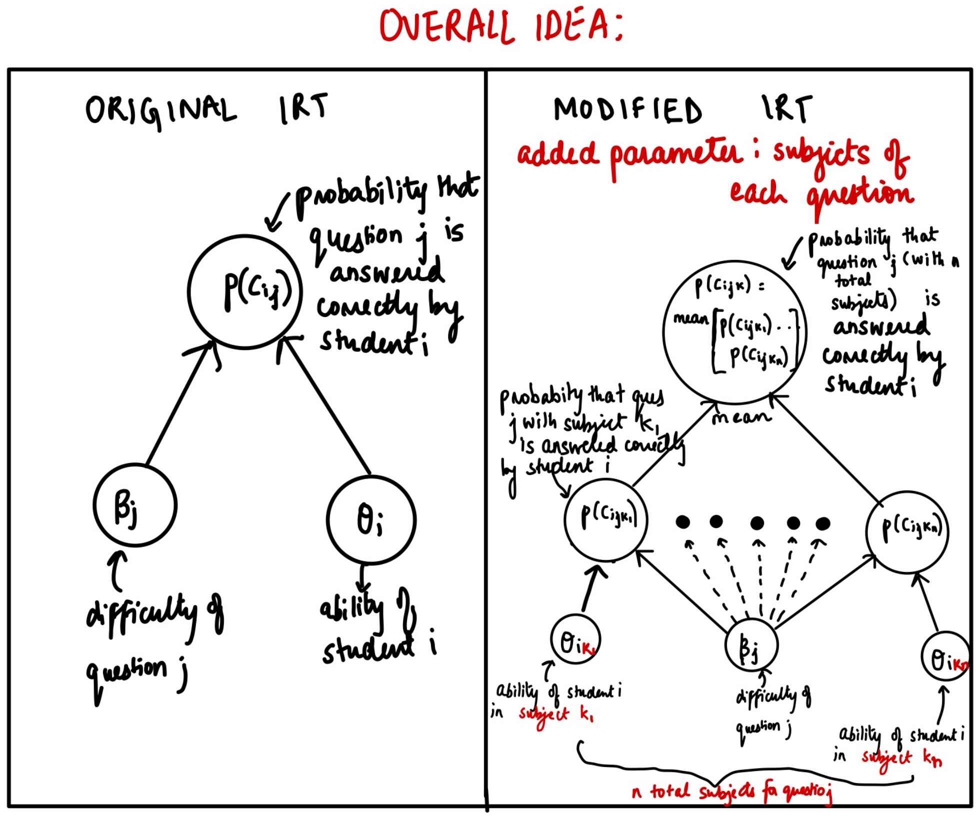
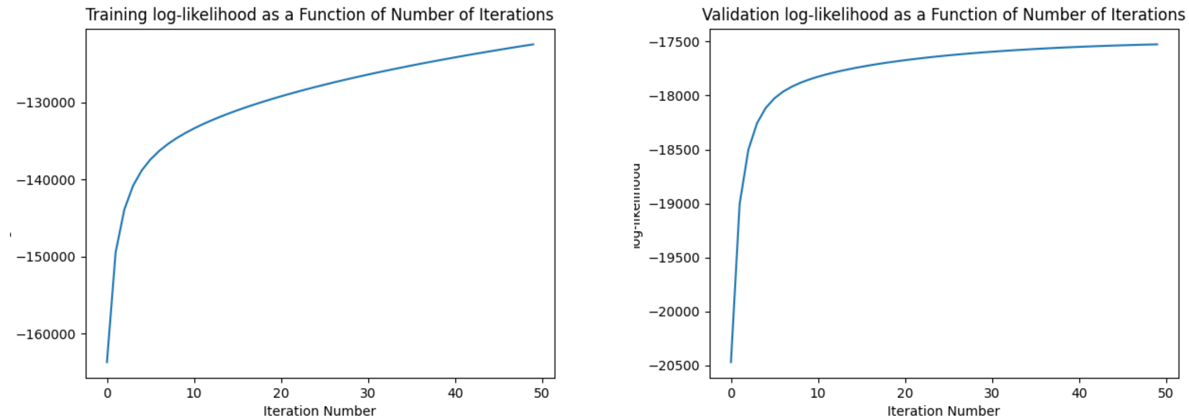


Figure 12

3. Comparison



(a) (b)

Figure 13

Model/Dataset	Validation	Test
Baseline IRT	0.7058989556872707	0.7064634490544736
Modified IRT	0.6916454981653966	0.6991250352808355

After modification, the accuracy of both validation set and test set dropped by 1%. The modification is not perfect and, we will discuss in the limitation section.

4. Limitations

- It is possible that in some cases, or for some questions, a subject is more emphasized or weighted higher than other subjects. In this case, our assumption doesn't hold true. To address this, we can add an additional dimension with a matrix that has weights of different subjects for each question depending on how much that subject is tested in the particular question.
- We have not considered the gender, date of birth, and premium pupil of the students in the model. These factors may potentially influence the prediction if there exists a correlation between them and their ability to solve a problem. One possible extension is to add all this information as a given factor to help us improve the prediction.
- We need a large amount of data to train if we want to get a good model. Since we group the questions into subjects based, therefore each question may have a small amount of data, and we need a large amount of data to train the model. In the theta matrix, we observe a large number of ones, which indicates data sparsity and 74% of all data are not modified from the initial value, that is a certain student has not performed in a certain subject of the question. The more data we have, the better model we can get. If we have a large amount of data, we can use cross-validation to get a better model. With fewer data in a question subject, one possible extension is to bootstrap the data to get more data to train the model.

Another possible extension is to use prior knowledge to help us get a better model, similar to Maximum A-Posteriori estimation, we can use prior knowledge on the question subjects to have a better model.

- The model is not flexible enough. The model we use is a linear model, which means the relationship between the response and the predictors is linear. One possible solution is to use a non-linear model, such as a neural network. However neural networks are more complicated and require more data to train.
- Another limitation is we have not considered the correlation between the subjects of the questions, in the question metadata, most questions are linked to multiple subjects. If some subjects are highly correlated, we have to consider the problem of multicollinearity. One possible extension is to add covariance between each subject (association between subjects) as given factor to help us improve the prediction.

Contributions:

Part A.1 Ananya
Part A.2 Jason
Part A.3 Guo
Part A.4 Guo
Part B.1 Jason
Part B.2 Ananya
Part B.3 Guo
Part B.4 Guo, Ananya, Jason