CSC311F20 Final Report

Zhiyuan Lu, Ziyi Zhou, Jinyu Hou

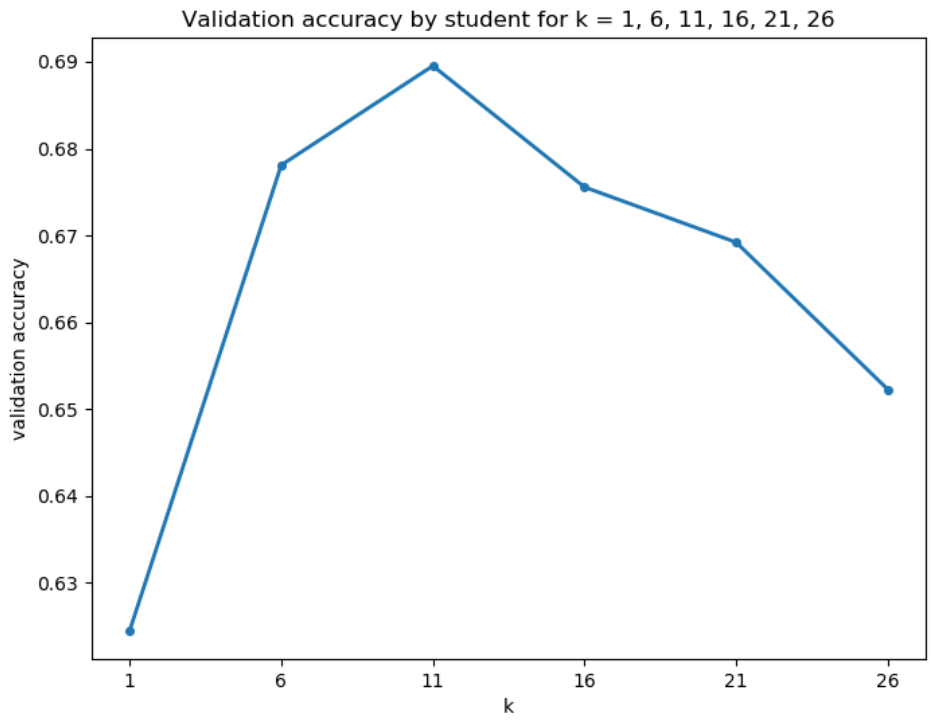
December 20, 2020

Part A

1. k-Nearest Neighbour

(a) The validation accuracy for $k \in \{1, 6, 11, 16, 21, 26\}$ is as follows:

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

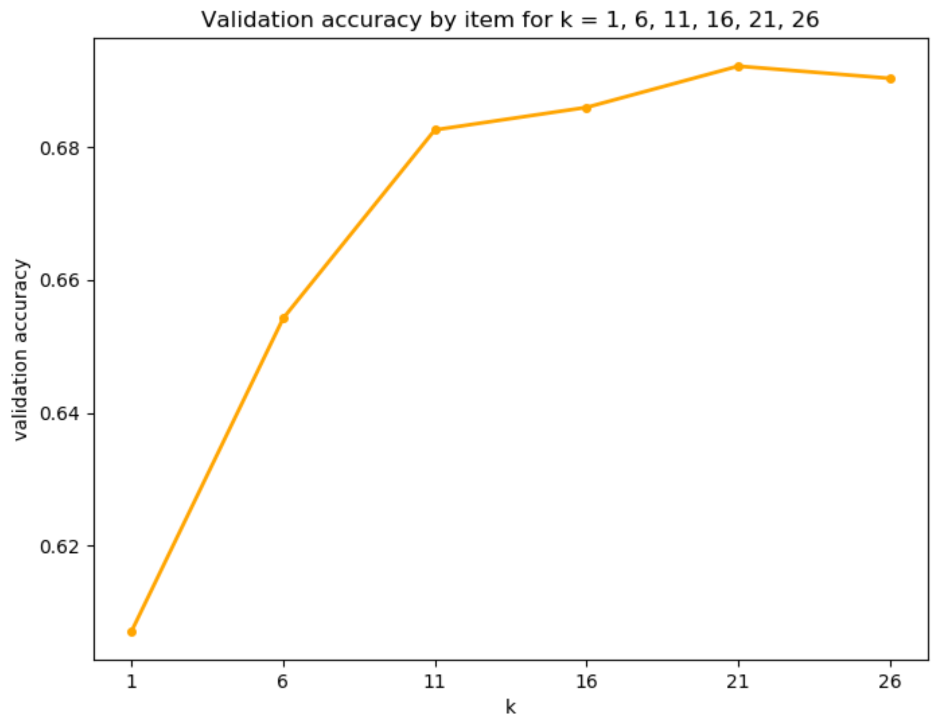


(b) k^* that has the highest performance on validation data is 11. Its test accuracy is 0.6841659610499576.

(c) Since we used the kNN Imputer to complete missing values using mean values from k nearest neighbors from the training data, the underlying assumption is that two questions are close if the answers that neither are missing are close. In other words, for each question, we can guess the answers from student who haven't answered this question with other questions that are close to it.

The validation accuracy for $k \in \{1, 6, 11, 16, 21, 26\}$ of item-based collaborative filtering is as follows:

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



k^* that has the highest performance on validation data is 21. Its test accuracy is 0.6683601467682755.

(d) The test accuracy for user-based collaborative filtering using the best performing k is 0.6841659610499576, and the test accuracy for item-based collaborative filtering is 0.6683601467682755. So user-based collaborative filtering performs better.

(e)
1. The accuracy depends on the quality of the training data. If there are many errors in the labeling of the correctness of the answers, the performance of the model will be poor.
2. There are many missing values in the training data, as each student only answered a small fraction of the questions. We used the kNN imputer to complete missing values, which assumes that students who performed similarly on questions that are already answered will also perform similarly on unanswered questions. However, this assumption may not hold in many cases. For example, two students may both perform well on an easy question subject, but perform very differently on a difficult subject because of difference in teaching quality.
3. kNN is computationally expensive. Because in each query, the distance from the instance to be classified to each item in the training set needs to be calculated and then sorted.
4. Since the data is in very high dimension, most points have approximately the same distance, so the distance metric is not very useful.

2. Item Response Theory

(a) N: number of students, D: number of questions

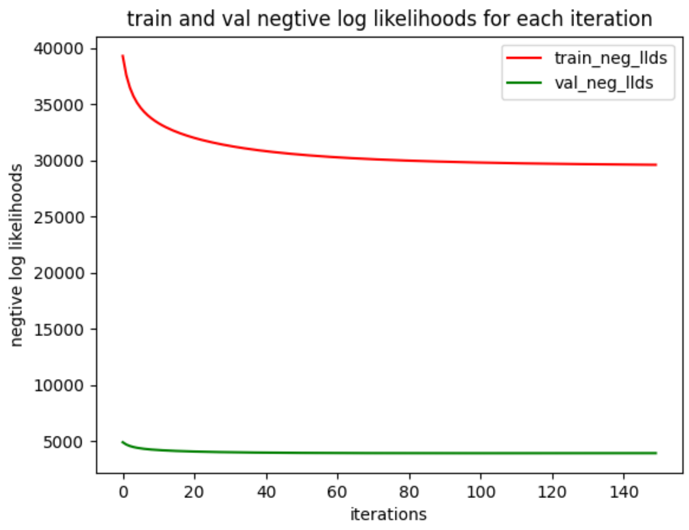
$$
\begin{aligned}
\log p(\mathbf{C}|\boldsymbol{\theta},\boldsymbol{\beta}) &= \sum_{i=1}^{N}\sum_{j=1}^{D}\log\Big[p(c_{ij=1}|\theta_i,\beta_j)^{c_{ij}}\big(1-p(c_{ij=1}|\theta_i,\beta_j)\big)^{1-c_{ij}}\Big] && (1)\\
&= \sum_{i=1}^{N}\sum_{j=1}^{D} c_{ij}\log p(c_{ij=1}|\theta_i,\beta_j) + (1-c_{ij})\log\big(1-p(c_{ij=1}|\theta_i,\beta_j)\big) && (2)\\
&= \sum_{i=1}^{N}\sum_{j=1}^{D} c_{ij}\log\big(\frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)}\big) + (1-c_{ij})\log\big(1-\frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)}\big) && (3)
\end{aligned}
$$

let $y = p(c_{ij=1}|\theta_i,\beta_j) = \frac{\exp(z)}{1+\exp(z)}$, $z = \theta_i - \beta_j$

$$
\begin{aligned}
\frac{\partial\big(\log p(\mathbf{C}|\boldsymbol{\theta},\boldsymbol{\beta})\big)}{\partial\theta_i} &= \frac{\partial\big(\log p(\mathbf{C}|\boldsymbol{\theta},\boldsymbol{\beta})\big)}{\partial y}\cdot\frac{\partial y}{\partial z}\cdot\frac{\partial z}{\partial\theta_i} && (4)\\
&= \frac{\partial\sum_{i=1}^{N}\sum_{j=1}^{D} c_{ij}\log y + (1-c_{ij})\log(1-y)}{\partial y}\cdot\frac{\partial y}{\partial z}\cdot\frac{\partial z}{\partial\theta_i} && (5)\\
&= \big(\sum_{j=1}^{D}(\frac{c_{ij}}{y}-\frac{1-c_{ij}}{1-y})\big)\cdot y(1-y)\cdot 1 && (6)\\
&= \big(\sum_{j=1}^{D}(\frac{c_{ij}}{y}-\frac{1-c_{ij}}{1-y})\cdot y(1-y)\big)\cdot 1 && (7)\\
&= \sum_{j=1}^{D} c_{ij} - y && (8)\\
&= \sum_{j=1}^{D} c_{ij} - \frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)} && (9)
\end{aligned}
$$

Similarly,

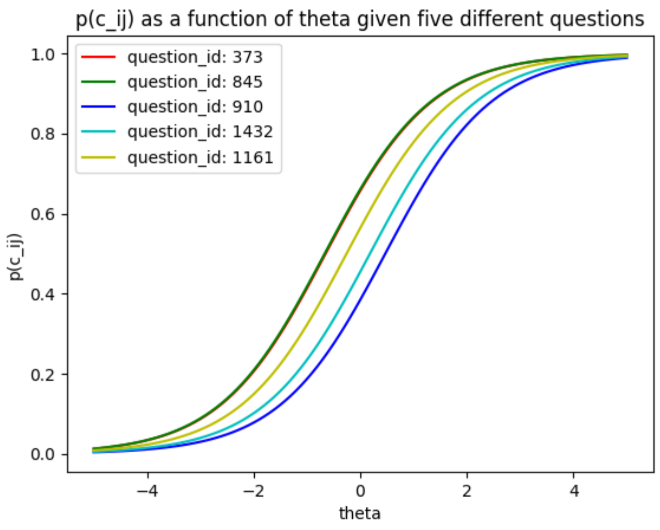
$$
\begin{aligned}
\frac{\partial\big(\log p(\mathbf{C}|\boldsymbol{\theta},\boldsymbol{\beta})\big)}{\partial\beta_j} &= \frac{\partial\big(\log p(\mathbf{C}|\boldsymbol{\theta},\boldsymbol{\beta})\big)}{\partial y}\cdot\frac{\partial y}{\partial z}\cdot\frac{\partial z}{\partial\beta_j} && (10)\\
&= \big(\sum_{i=1}^{N}(\frac{c_{ij}}{y}-\frac{1-c_{ij}}{1-y})\cdot y(1-y)\big)\cdot(-1) && (11)\\
&= \sum_{i=1}^{N} y - c_{ij} && (12)\\
&= \sum_{i=1}^{N}\frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)} - c_{ij} && (13)
\end{aligned}
$$



(b)

(c) The final validation accuracy is 0.70575783234547, the final test accuracy is 0.7061812023708721.

(d) All curves shapes like exponential function: goes up as theta becomes greater and converges to 0 as theta $\rightarrow -\infty$, converges to 1 as theta $\rightarrow \infty$.
These curves represent and compare that how likely a question selected from dataset will be answered correctly, as student's ability improves.



3. Neural Networks

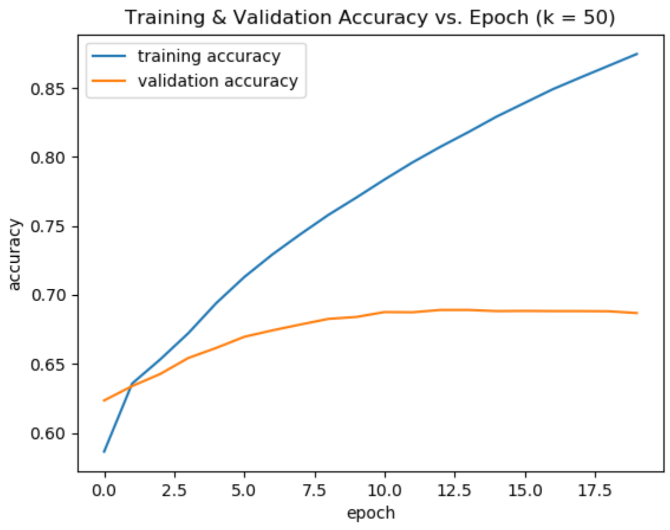
(a) Here are three differences between ALS and neural networks:

1. The latent space of ALS is linear while the latent space of neural network is not necessarily linear;
2. Neural networks applies backpropagation while ALS doesn't;
3. ALS fixes one parameter and optimize another alternatively with the gradient while neural networks optimize all parameters after backpropagation.

(b) `AutoEncoder` has been implemented in `part_a/neural_network.py`.

(c) The chosen learning rate is 0.03, the chosen number of iterations is 20, and we chose $k^* = 50$ as it gives the highest accuracy.

(d) Here is a plot of training/validation accuracy vs. iteration with $k^* = 50$:



The final test accuracy is as follows:

```
Test Acc: 0.6776742873271239
```

(e.) We chose $\lambda = 0.1$. The final validation and test accuracy are as follows:

```
Validation Acc: 0.6865650578605701
Test Acc: 0.6833192209991532
```

The model performs slightly better with the regularization penalty.

4. Ensemble

The final values of validation and test accuracy are as follows:

```
Validation Accuracy: 0.7046288456110641
Test Accuracy: 0.707310189105278
```

We selected the Item Response Theory from Part A Q2 as the base model. The ensemble process was implemented as the following:

- Bootstrap the training set by randomly selecting samples with replacement from the training set, and the sample size is the same as the size of the original training set.
- Train the IRT model with each of the 3 training sets, which generates 3 pairs of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$.
- Calculate the validation accuracy: make 3 predictions using each pair of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ for the validation data, then average the predicted correctness by taking the majority vote. In other words, if there are more 0's than 1's in the 3 predictions, then the average prediction is 0, and vice versa.
- Calculate the test accuracy: make 3 predictions using each pair of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ for the test data, then average the predicted correctness by taking the majority vote.

With the original IRT model in Q2, the final validation accuracy is 0.70575783234547, the final test accuracy is 0.7061812023708721. Since the difference in validation and test accuracy is about 0.001, we didn't obtain better performance using the ensemble. This is because we had adequate data to train the parameters, and the validation and test data has the same distribution as the training data, so that generalizing training data will not improve the accuracy of the model.

Part B

1. Formal Description

We extended the Item-Response Theory model in the following ways:

1. We added three additional parameters, $\mathbf{k}$, c and $\boldsymbol{\alpha}$.
   $\mathbf{k}$ is a vector with dimension equal to the number of questions. k_j (the jth element of $\mathbf{k}$) estimates how steep the sigmoid looks (i.e. how discriminative the question is) for question j. The entries of $\mathbf{k}$ are initialized to 0.5.
   c is a scalar estimating the probability of getting a right answer via random guesses.
   $\boldsymbol{\alpha}$ is a matrix with dimension of $num\_student \times num\_subjects$, in which α_{is} (the (i, s)th element of $\boldsymbol{\alpha}$) estimates how familiar the student i is with subject s. The entries in $\boldsymbol{\alpha}$ are initialized as
   $$\alpha_{is} = \frac{\text{\# question in subject s that student i answered correctly}}{\text{\# question in subject s that student i has answered}}$$
   For the ease of computation, we would use $\boldsymbol{\alpha'}$, which is a matrix with dimension of $num\_student \times num\_questions$, for the calculation of probability. α'_{ij} (the (i, j)th element of $\boldsymbol{\alpha'}$) computes the average familiarity of student i with the subjects to which question j belongs, which means
   $$\alpha'_{ij} = \frac{\sum_{s \in \text{subjects question j belongs to}} \alpha_{is}}{\text{\# subjects question j belongs to}}$$
2. We added L_2 regularizer with regularization penalty $\lambda = 0.1$ for $\boldsymbol{\theta}$, $\boldsymbol{\beta}$, $\mathbf{k}$ and $\boldsymbol{\alpha}$.
3. We initialized the entries of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ to 0.5.

Therefore, the resulting model would be:

$$p(correct_{ij}|\theta_i, \beta_j, k_j, c, \alpha'_{ij}) = c + (1 - c) * sigmoid(k_j * (\theta_i - \beta_j + \alpha'_{ij}))$$

The resulting log-likelihood would be: (let $y = p(correct_{ij=1}|\theta_i, \beta_j, k_j, c, \alpha'_{ij})$)

$$\log p(\mathbf{correct}|\boldsymbol{\theta}, \boldsymbol{\beta}, \mathbf{k}, c, \boldsymbol{\alpha'}) = \sum_{i=1}^{N} \sum_{j=1}^{D} \left[correct_{ij} \log y + (1 - correct_{ij}) \log(1 - y) \right]$$

The resulting learning objective would be to minimize:

$$- \log p(\mathbf{correct}|\boldsymbol{\theta}, \boldsymbol{\beta}, \mathbf{k}, c, \boldsymbol{\alpha'}) + \frac{\lambda}{2} * (||\boldsymbol{\theta}||_2^2 + ||\boldsymbol{\beta}||_2^2 + ||\mathbf{k}||_2^2 + ||\boldsymbol{\alpha}||_2^2)$$

Before the extension, the model underfits the data since both training accuracy and validation accuracy are low, as follows:

```
Training Acc: 0.7393981089472199
Validation Acc: 0.70575783234547
Test Acc: 0.7061812023708721
```

Therefore, the addition of $\mathbf{k}$, c and $\boldsymbol{\alpha}$ are intended to reduce the bias of the model because they provide additional information regarding the data and makes the model more complex. More concisely:

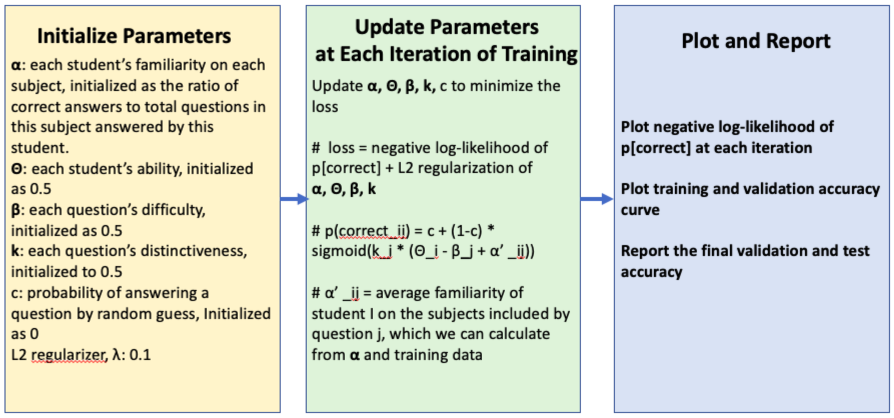
- $\mathbf{k}$ is expected to improve the model because it distinguishes each question by their learning curve of sigmoid funtion and therefore the model could perform better when the students performs influentially differently on different questions.
- c is expected to improve the model by separating out the probability that the question was answered correctly by random guess and so this could stabilize the model by reducing the instability from the rest of our model (the rest of the model, the sigmoid function involves a lot of parameters and therefore could be instable). In this way, the model could perform better when the performance of different students or the students' performance on different questions has a random pattern.

- $\boldsymbol{\alpha}$ is expected to improve the accuracy of the model because it provides information specifictly to each $correct_{ij}$ instead of providing information to student i or question j in general. The model becomes more complex in this way.

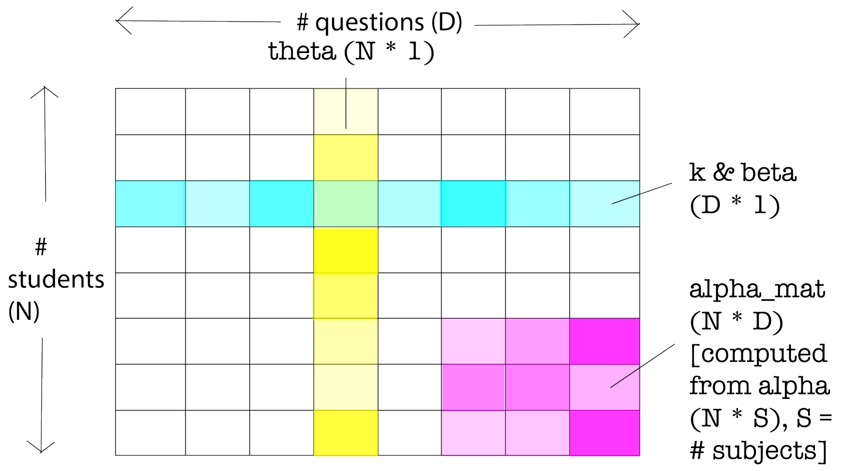
In addition, since the training accuracy is slightly larger than the validation accuracy, we added the L_2 regularizer to prevent potential overfitting issue. We also tried different parameter initialization methods and picked the best one based on the model's performance on the validation set to prevent from stucking on local optima.
Based on the above analysis, the model is expected to produce both higher training and validation accuracy.

2. Figure or Diagram

The following figure is a flowchart of our algorithm.


Initialize Parameters
α: each student's familiarity on each subject, initialized as the ratio of correct answers to total questions in this subject answered by this student.
Θ: each student's ability, initialized as 0.5
β: each question's difficulty, initialized as 0.5
k: each question's distinctiveness, initialized to 0.5
c: probability of answering a question by random guess, Initialized as 0
L2 regularizer, λ: 0.1
Update Parameters at Each Iteration of Training
Update α, Θ, β, k, c to minimize the loss
# loss = negative log-likelihood of p[correct] + L2 regularization of α, Θ, β, k
# p(correct_ij) = c + (1-c) * sigmoid(k_j * (Θ_i - β_j + α' _ij))
# α' _ij = average familiarity of student I on the subjects included by question j, which we can calculate from α and training data
Plot and Report
Plot negative log-likelihood of p[correct] at each iteration
Plot training and validation accuracy curve
Report the final validation and test accuracy

Below is a brief illustration of the parameter shape:


# questions (D)
theta (N * 1)
k & beta (D * 1)
# students (N)
alpha_mat (N * D) [computed from alpha (N * S), S = # subjects]

3. Comparison or Demonstration

3.1. Comparison of models

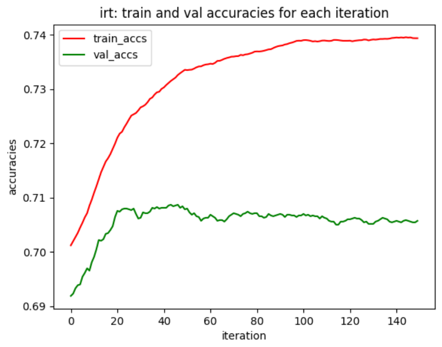
Table 1 shows the comparison of accuracies obtained from the baseline models in Part A and our extended model in Part B. (The accuracies are rounded to 4 decimal places).

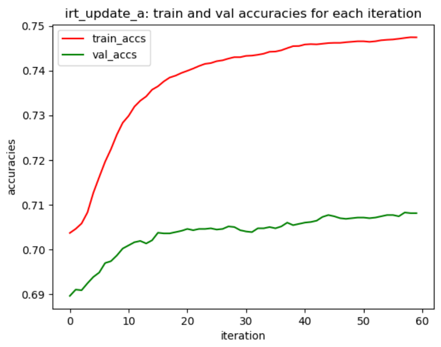
Combined with the plots on the next page, we could conclude that the extended model performs slightly better than the baseline model by producing a slightly higher validation and test accuracy. The plot on the

Model	Validation Accuracy	Test Accuracy
kNN (user-based)	0.6895	0.6842
kNN (item-based)	0.6922	0.6684
Item Response Theory	0.7058	0.7062
Neural Networks	0.6866	0.6833
Ensemble with IRT	0.7046	0.7073
Extended Item Response Theory	**0.7083**	**0.7079**

Table 1: Comparison of accuracies obtained by baseline models and extended model

left is the training and validation accuracy of the baseline IRT model, and the plot on the right is the training and validation accuracy of the extended IRT model.





3.2. Experiment

Since we have made the model more complex by adding new parameters, we hypothesize that the accuracy of our model will increase and the benefits are mainly due to optimization instead of regularization.

To test for the hypothesis, we did an experiment by removing the regularizer (i.e., by setting $\lambda = 0$) and test for accuracies. The result is as follows:

Model	Regularization Penalty	Validation accuracy	Test Accuracy
Experiment Model	$\lambda = 0$	0.7082	0.7076
Model with Regularizer	$\lambda = 0.1$	0.7083	0.7079

Table 2: Comparison of accuracies obtained by experiment model and fully extended model

The result is that the accuracy of the model only decreased by a negligible amount after removing the regularizer, compared to the improvement after the extension. Therefore, our hypothesis is supported by the result.

To further experiment for where the benefits are from, we can compare the test accuracies and training curves of the following variation of the model:

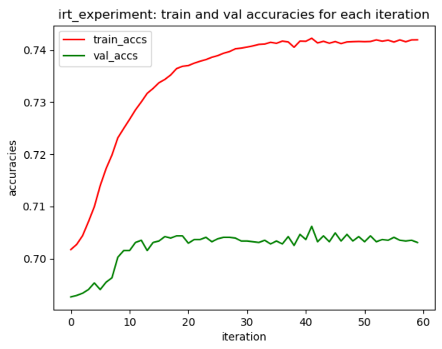
- Only add parameters $\mathbf{k}$ and c. In this case, we don't consider the difference in the familiarity to each question subject of a student. By comparing this to the fully extended model, we can test whether adding $\boldsymbol{\alpha}$ is an effective optimization.

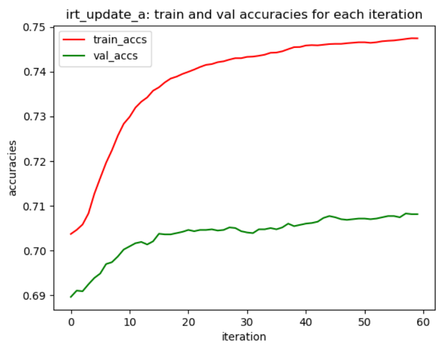
After carrying out the experiment, we compare the validation and test accuracy of the experiment model to the fully extended model in Table 3.

The plot on the next page on the left is the training and validation accuracy of the experiment IRT model without $\boldsymbol{\alpha}$, and the plot on the right is the training and validation accuracy of the extended IRT model. We can notice that the full model has less fluctuations in the training and validation accuracy.

Model	Parameters	Validation accuracy	Test Accuracy
Experiment Model	$\boldsymbol{\theta}$, $\boldsymbol{\beta}$, $\mathbf{k}$, c, λ	0.7031	0.7053
Fully extended model	$\boldsymbol{\theta}$, $\boldsymbol{\beta}$, $\mathbf{k}$, c, λ, $\boldsymbol{\alpha}$	0.7083	0.7079

Table 3: Comparison of accuracies obtained by experiment model and fully extended model





The fully extended model has a slightly higher validation and test accuracy than the model without the parameter $\boldsymbol{\alpha}$ in the experiment, which suggests that the benefits are due to optimization by students' familiarity level to different question subjects.

4. Limitation

Despite the extension, the model performed only slightly better than the baseline model. There is still lots of space for improvement. Here are some additional limitations:

- In the case that one of *num_student* and *num_questions* is small (e.g., 1000 students and 3 questions), the model could perform poorly. In this case, $\boldsymbol{\beta}, \mathbf{k}$ (in the case that *num_students* is small) and $\boldsymbol{\theta}$ (in the case that *num_questions* is small) would not be able to estimate the corresponding parameters accurately because it would be hard to learn a pattern from limited data. (e.g., it is hard to tell a student's ability from their performance on 3 questions) Most of the existing model should fail in this case.

- In the case that the data sparsity of the training matrix is large (there are lots of hold-outs or unknown response), the model could perform poorly. The reason is similar to above. For example, it is hard to tell if a student is likely to answer a question correctly from their response on the other 2 questions.Most of the existing model should fail in this case.

- In the case that there are no obvious difference between questions or students, the model could perform poorly. In this case, the entries of $\boldsymbol{\beta}, \mathbf{k}$ (in the case that questions are similar) or $\boldsymbol{\theta}$ (in the case that students are performing similarly) would be similar and there would not be a clear pattern that could be learned. A model which explores alternate relationship in the data (e.g., neural network) may perform better in this case.

  The IRT would fail because there would not be any obvious pattern in the data and the randomness of whether a student could answer a question correctly would be high. The parameters of IRT so far are all either student-wise or question-wise. In the case that both students are questions are similar, the parameters would become similar and produce similar results around 0.5 and the log-likelihood of the model would also be high.

As a further extension, we could consider exploring further relationships within the data (e.g., students' age or gender).
Any additional further extension would be an open problem.

Appendix

Contributions

Part A

- Q1: Zhiyuan Lu
- Q2: Ziyi Zhou
- Q3: Jinyu Hou
- Q4: Zhiyuan Lu

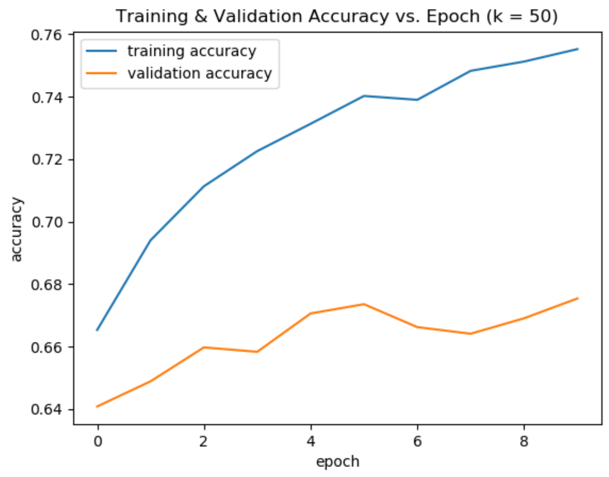
Part B

- Discussion: Jinyu Hou, Ziyi Zhou, Zhiyuan Lu
- Algorithm Implementation: Ziyi Zhou, Jinyu Hou
- Report Write-up: Jinyu Hou, Ziyi Zhou, Zhiyuan Lu

Some Additional Notes

During the exploration for potential extension for part B, we also considered extending `AutoEncoder` by applying `VariationalAutoEncoder`, which is a VAE model that has an additional layer to take the output from the `Encoder` as input and outputs parameters of a latent distribution (we applied Multivariate Gaussian distribution and therefore the parameters are $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$). Then, the distribution constructed from the parameters is used to sample for the input to the `Decoder`. $kl - divergence$ between the latent distribution and standard normal distribution was added to the loss function to reduce variance. Additionally, we also applied dropout to decrease the variance.

Since the original `AutoEncoder` model overfits the data, this would theoretically regularize the model to some extent. However, after several rounds of hyperparameter tuning, we didn't find a model with obvious improvement. So we didn't choose the `VariationalAutoEncoder` model at the end. Below is the accuracy vs. iteration plot: The code for `VariationalAutoEncoder` is included in `part_b/vae.py`.


Training & Validation Accuracy vs. Epoch (k = 50)
training accuracy
validation accuracy
accuracy
epoch