Project Report

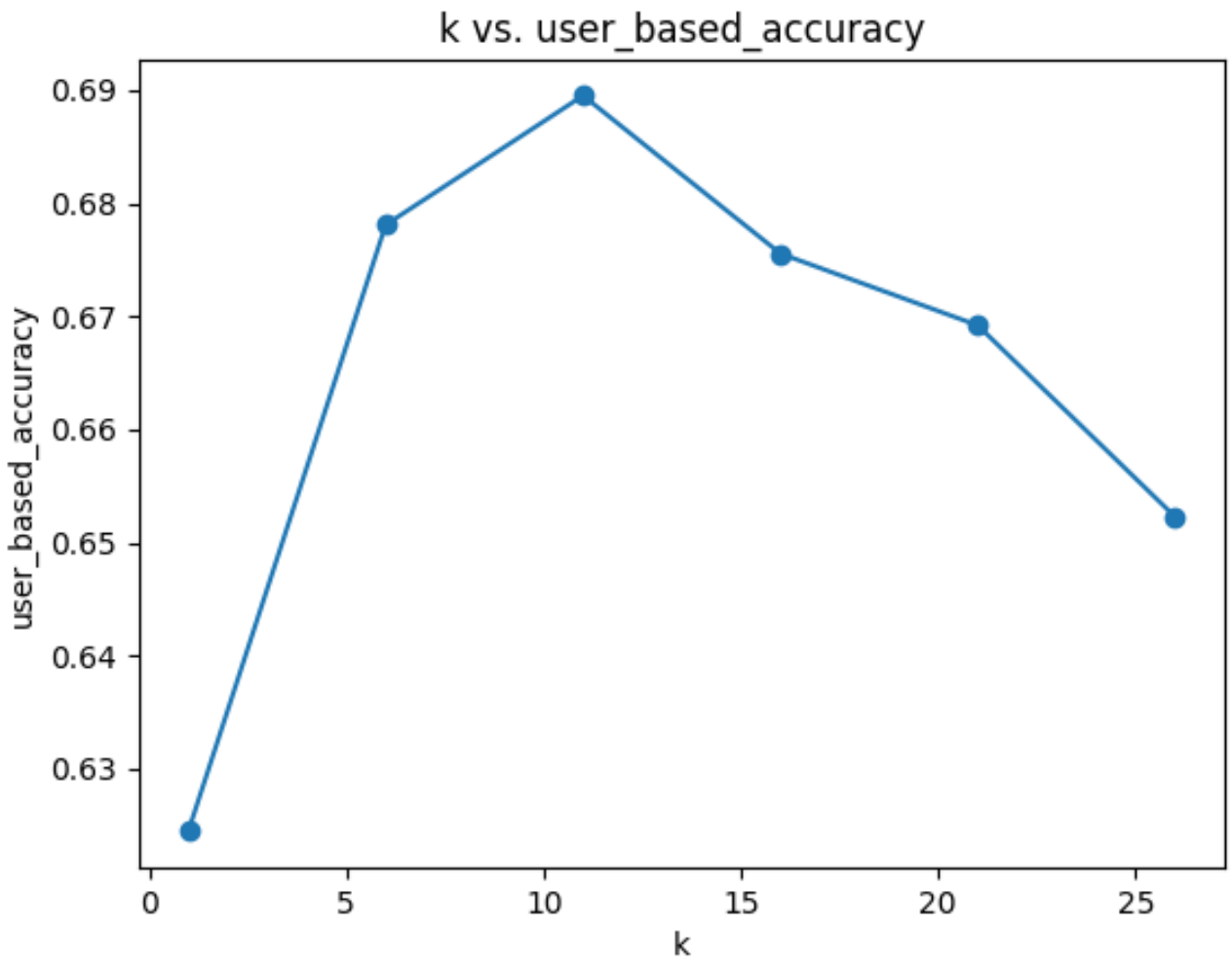
November 2023

1 Part A

1.1 k-Nearest Neighbor

```
----------User-based predication accuracy----------
k = 1, validation accuracy = 0.6244707874682472
k = 6, validation accuracy = 0.6780976573525261
k = 11, validation accuracy = 0.6895286480383855
k = 16, validation accuracy = 0.6755574372001129
k = 21, validation accuracy = 0.6692068868190799
k = 26, validation accuracy = 0.6522720858029918
```

(a)

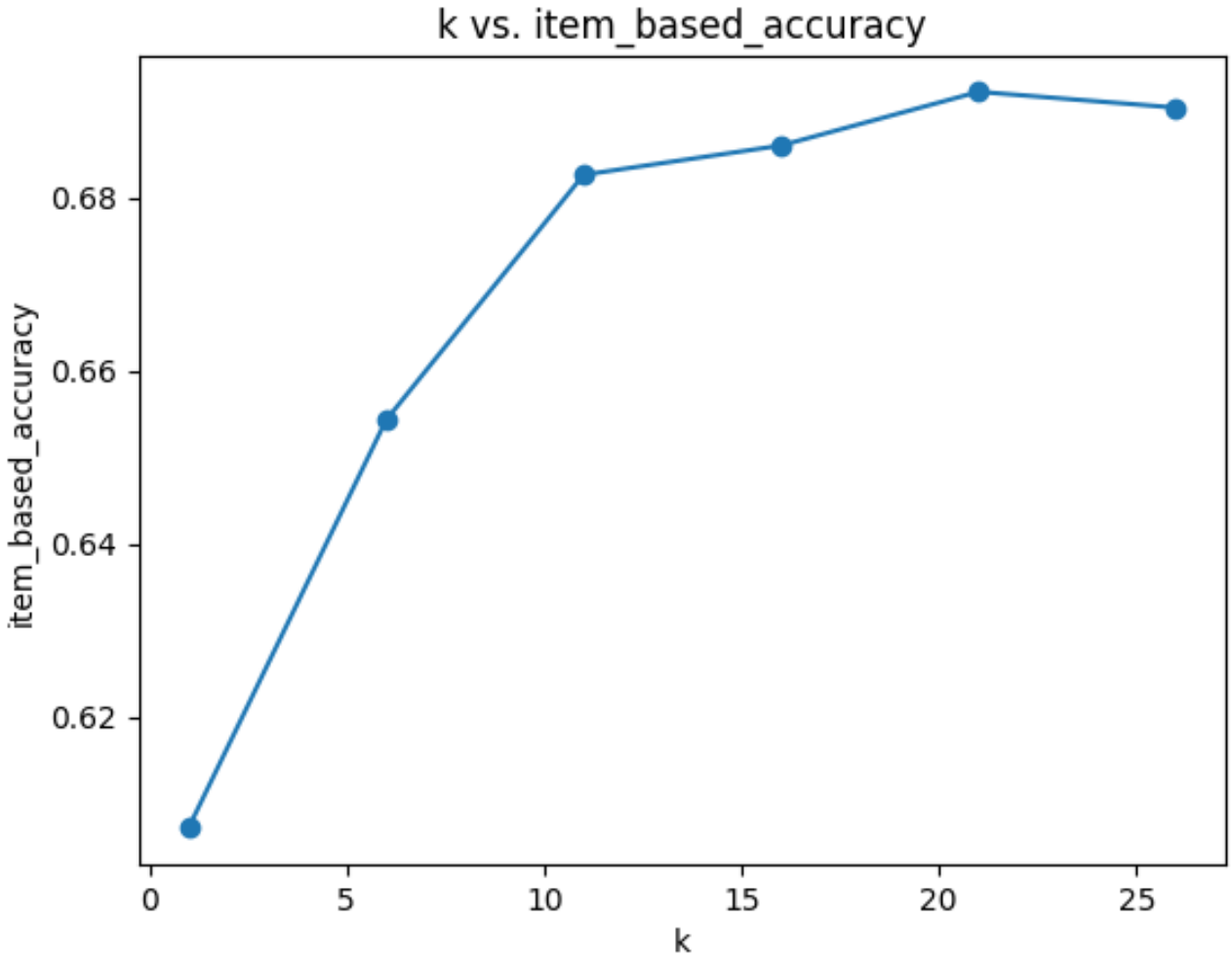


(b) k^* that has the highest performance on validation data is 11.

Report: The k which achieve the highest performance is 11, and the corresponding test accuracy is 0.68416596104

(c)

```
----------Item-based predication accuracy----------
k = 1, validation accuracy = 0.607112616426757
k = 6, validation accuracy = 0.6542478125882021
k = 11, validation accuracy = 0.6826136042901496
k = 16, validation accuracy = 0.6860005644933672
k = 21, validation accuracy = 0.6922099915325995
k = 26, validation accuracy = 0.69037538808919
```



The k which achieve the highest performance is 21, and the corresponding test accuracy is 0.6816257408975445

(d) Since the k^* which achieves the highest performance is 11 for user-based and 21 for item-based(the smaller k means computational time lower), $11 < 21$, and the test accuracy for those k^* is 0.6841 for user-based and 0.6816 for item-based(higher accuracy means more accurate), $0.6841 > 0.6816$. Then we know the user-based performs better.

(e) 1.Curse of dimension: The dimension of the input is large, therefore data points have similar distances between each other and will raise problems for knn.
2.Computation time is large, also we need to store the training data set in the training stage, which will raise problems if the dataset is large.

1.2 Item Response Theory

(a)

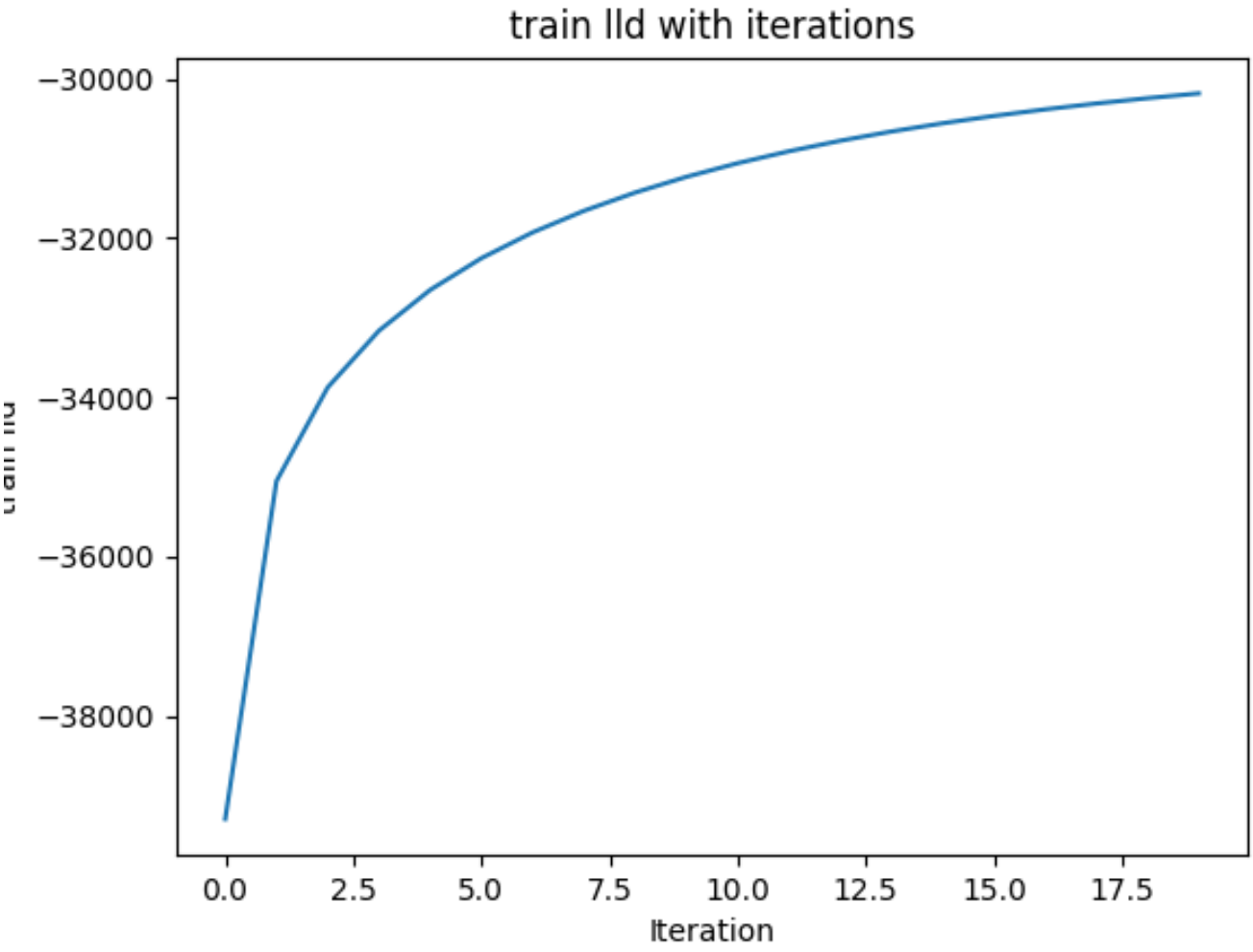
$logp(C|\theta,\beta) = \sum_{i=1}^{n}\sum_{j=1}^{m}(c_{ij}\log(\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)})+(1-c_{ij})\log(\frac{1}{1+exp(\theta_i-\beta_j)})) = \sum_{i=1}^{n}\sum_{j=1}^{m}(c_{ij}(\theta_i-\beta_j)-log(1+exp(\theta_i-\beta_j)))$

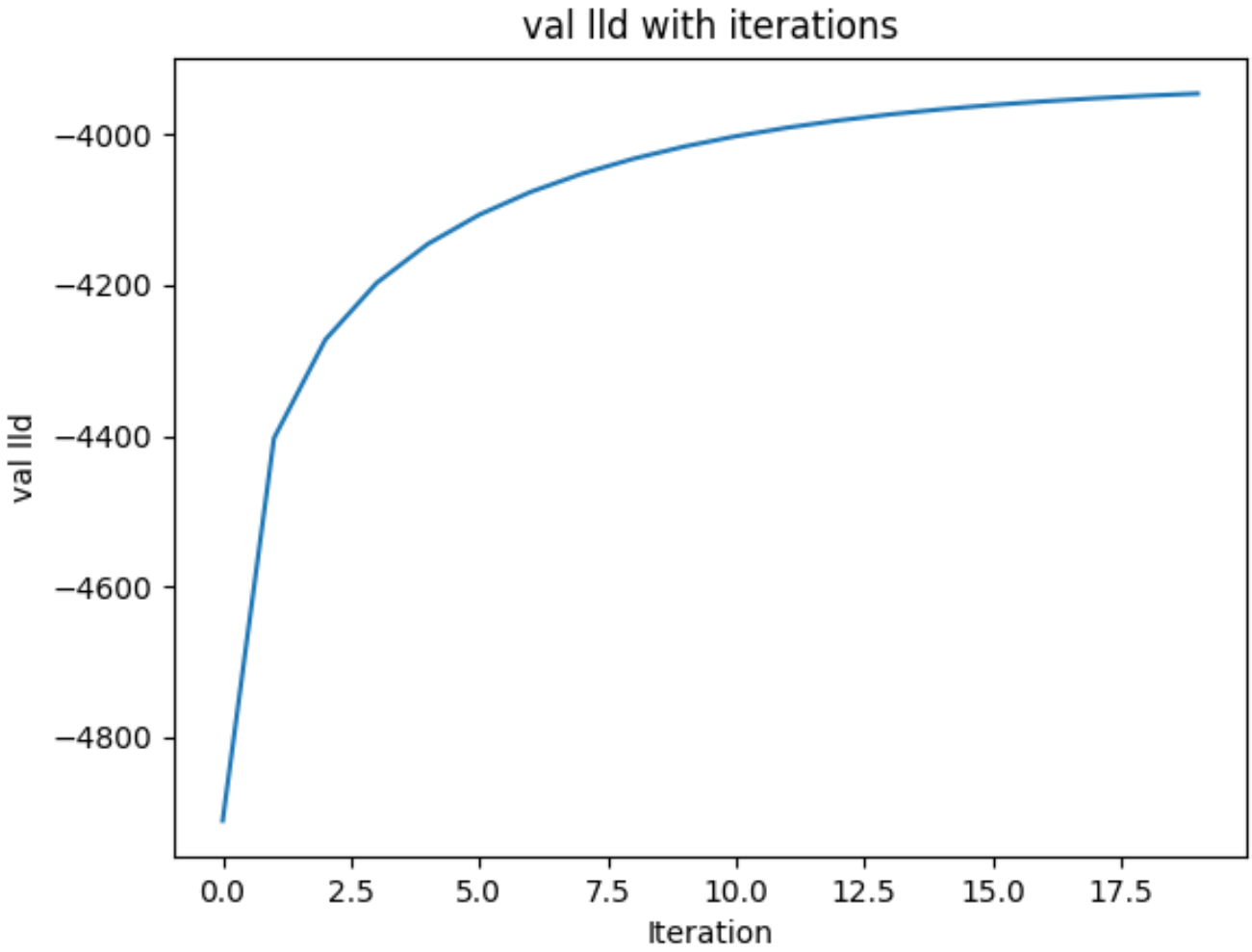
$\frac{\partial logp(C|\theta,\beta)}{\partial\theta_i} = \sum_{j=1}^{m}(c_{ij}-\frac{exp(\theta_i-\beta j)}{1+exp(\theta_i-\beta_j)}) = \sum_{j=1}^{m}(c_{ij}-p(c_{ij}=1|\theta_i,\beta_j))$

$\frac{\partial logp(C|\theta,\beta)}{\partial\beta_j} = \sum_{i=1}^{n}(-c_{ij}+\frac{exp(\theta_i-\beta j)}{1+exp(\theta_i-\beta_j)}) = \sum_{j=1}^{m}(-c_{ij}+p(c_{ij}=1|\theta_i,\beta_j))$

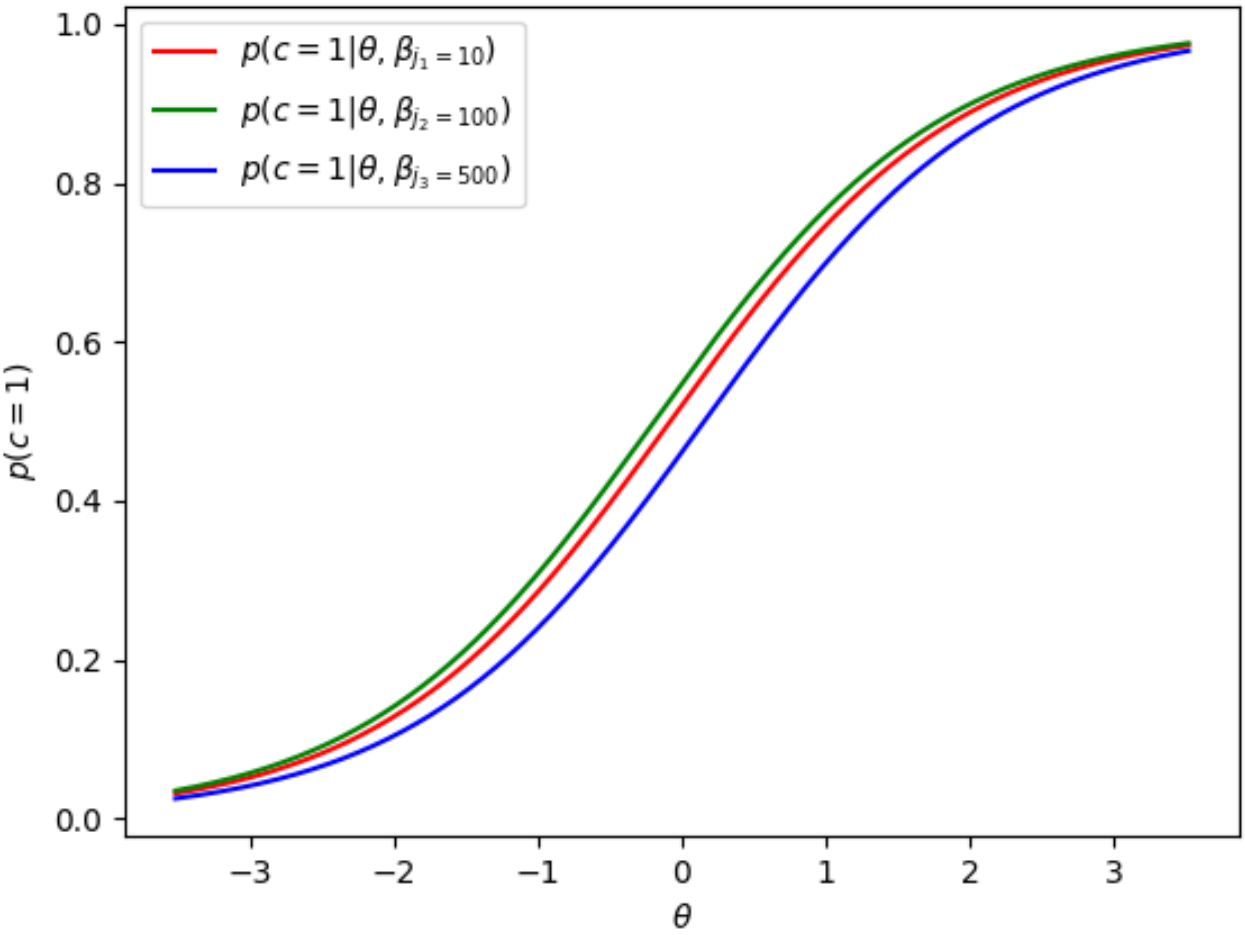
(b)

The learning rate that I set is 0.01, the number of iterations that I set is 20





(c)
The final validation accuracy is 0.7064634490544736, The test accuracy is 0.7050522156364663.
(d)



The shape of curves are similar to the sigmoid function, the curve represents if the student has greater ability given the difficulty of the question, the probability that the student will get the correct answer becomes greater.

1.3 Neural Networks

(a) Below is three differences between ALS and neural networks:

- Method of Training:
  - For ALS, we use an iterative optimization approach by fixing one set of variables in z and u, while optimizing the other. For example, fix z and update u by SGD and then in next iteration fix u and update z by SGD.
  - Neural Networks will update all of the parameter by applying backward propagation without fixing one variable.
- Expressivity:
  - ALS is dealing with linear relationship and cannot express non-linear. In other words, hypothesis spaces of ALS is smaller than neural network
  - Neural Network can use non-linear activation functions, here we use sigmoid function as activation function, its hypothesis spaces is larger than ALS as Neural Network with non-linear activation functions can represent non-linearity.
- Purpose
  - ALS is a collaborative filtering approach that is designed for recommendation system.
  - Neural Networks can solve diverse tasks usually in vision work like convolutional neural networks and natural language processing.

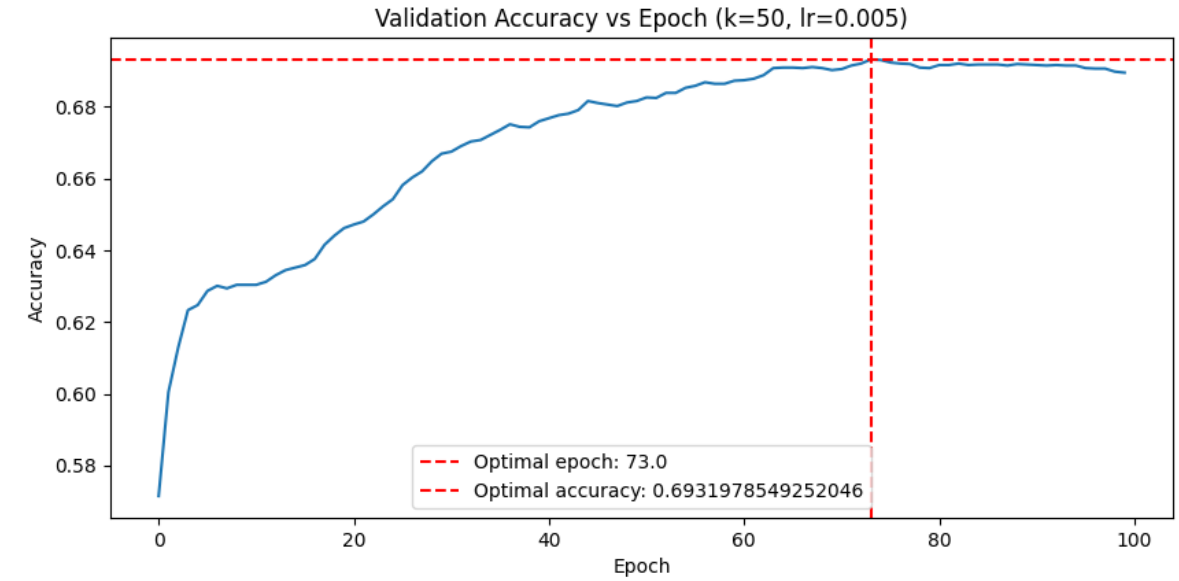
(b) The code is in the file `neural_network.py`.

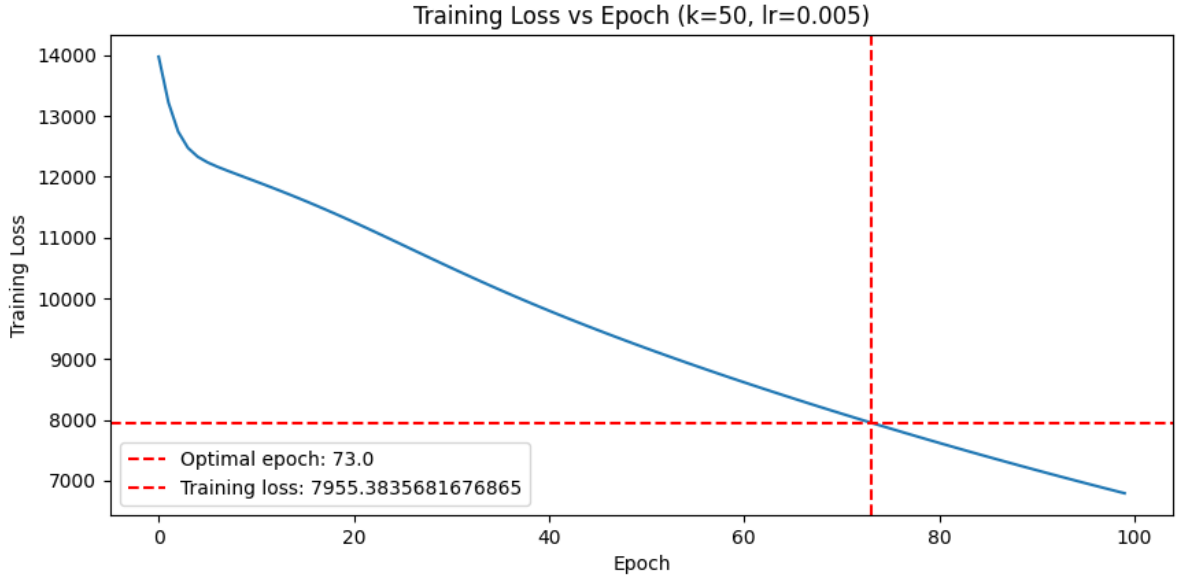
(c) Below is the comparison of different dimensions for autoencoder and different hyperparameters.

	Highest Validation Accuracy	Epoch Index for Accuracies
$k = 10$, learning rate=0.005	0.6760	99
$k = 10$, learning rate=0.01	0.6907	99
$k = 10$, learning rate=0.1	0.6878	18
$k = 50$, learning rate=0.005	0.6932	73
$k = 50$, learning rate=0.01	0.6929	41
$k = 50$, learning rate=0.1	0.6836	5
$k = 100$, learning rate=0.005	0.6842	49
$k = 100$, learning rate=0.01	0.6846	24
$k = 100$, learning rate=0.1	0.6778	4
$k = 200$, learning rate=0.005	0.6819	59
$k = 200$, learning rate=0.01	0.6843	29
$k = 200$, learning rate=0.1	0.6791	7
$k = 500$, learning rate=0.005	0.6774	51
$k = 500$, learning rate=0.01	0.6739	33
$k = 500$, learning rate=0.1	0.6696	8

Here we will choose $k^* = 50$, learning rate= 0.005, total epoch= 75(As epoch idx starts from 0; It is close to 74; It also show a decrease pattern for validation accuracies) with validation accuracy of 0.6932.

(d) Below is how the training and validation objectives changes as a function of epoch in the graph.

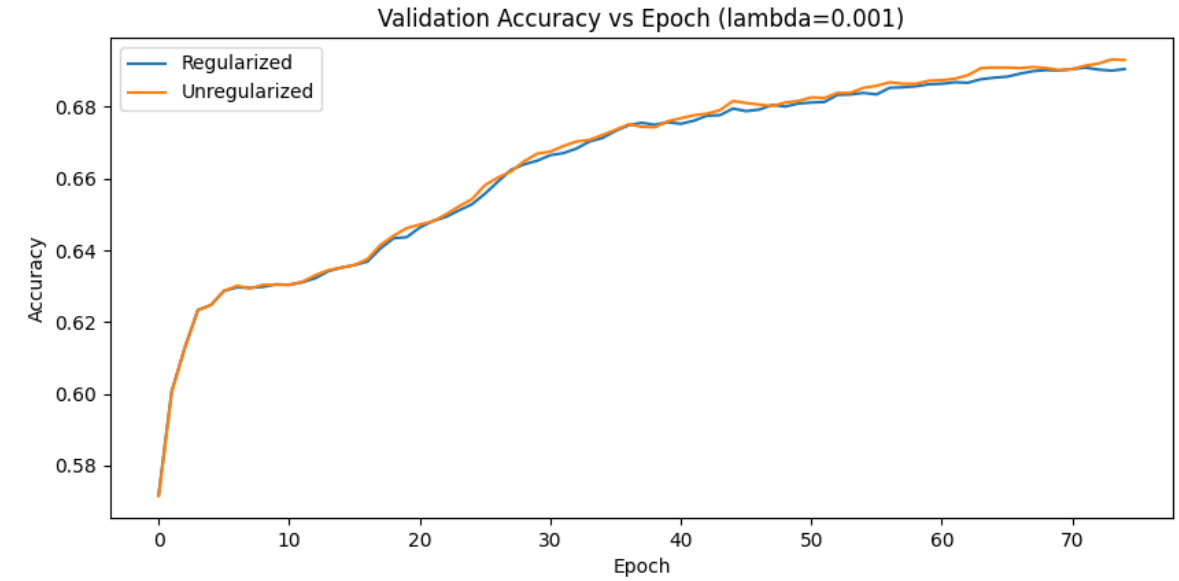


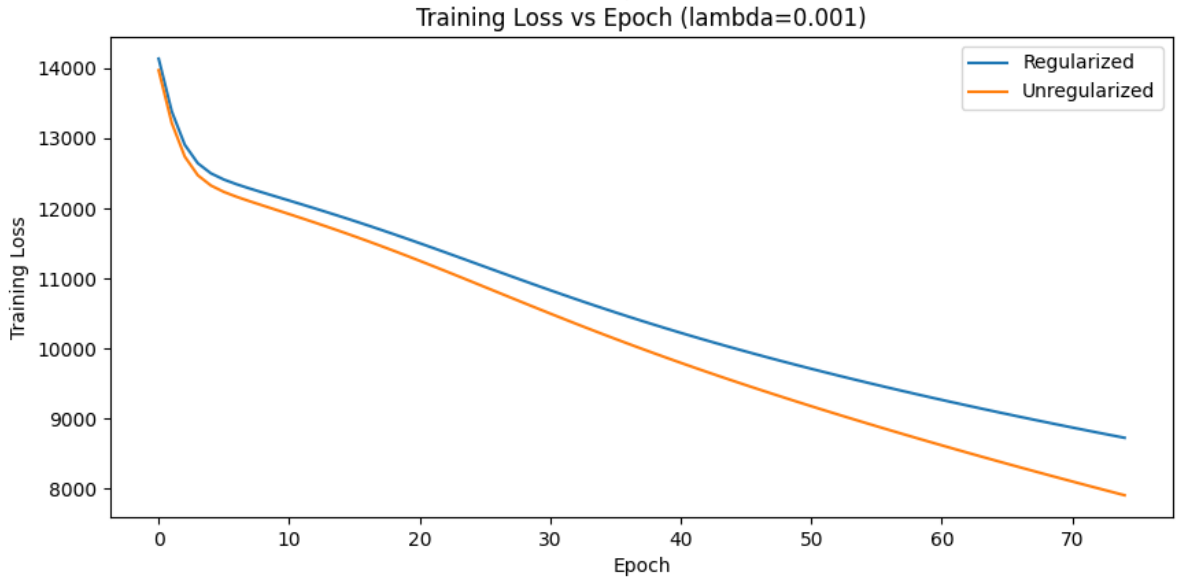


And we have the final test accuracy of 0.6768275472763196.

(e) After finetuning, we will choose $\lambda = 0.001$. The final validation accuracy is 0.6905 and test accuracy is 0.6813. Below is the comparison between each λ (Note: Our comparison is among regularized one and picking λ with highest final validation accuracy):

	Final Validation Accuracy	Test Accuracy
$k = 50$, learning rate=0.005, epoch=75, $\lambda = 0$	0.6932	0.6768
$k = 50$, learning rate=0.005, epoch=75, $\lambda = 0.001$	0.6905	0.6813
$k = 50$, learning rate=0.005, epoch=75, $\lambda = 0.01$	0.6787	0.6751
$k = 50$, learning rate=0.005, epoch=75, $\lambda = 0.1$	0.6246	0.6260
$k = 50$, learning rate=0.005, epoch=75, $\lambda = 1$	0.6242	0.6226





We found that although applying regularization decreased final validation accuracy, but we can see a noticeable increase in the test accuracy. The model performs better with the regularization penalty on test dataset.

1.4 Ensemble

Report:
I using IRT for the three models, with learning rate = 0.01 and iterations = 20.
Final validation accuracy is 0.70575783234547, Final test accuracy is 0.6979960485464296
Process:
First I generate 3 new datasets by resample on the training data, and then I use those three data sets separately to train the model using IRT and get three models. Then I generate 3 predictions by using the three models and average the predicted correctness for both validation data and test data.
Comparison:
By Item Response Theory (c) we know that the individual model(I use the same number of iterations and learning rate) achieves the final validation accuracy as 0.7064634490544736, and the final test accuracy be 0.7050522156364663, which are greater than this ensemble method and hence I don't obtain better performance using the ensemble. This may be because the resample set has some repetitive data and loses some important data during the resampling period.

2 Part B

We modify the neural network in part(a) since it has many potential extensions: it only has two layers, and we can augment the input by providing metadata that hasn't been utilized in part a and the student ability data that we get from the IRT model.

2.1 Binary Cross Entropy Loss

- **Description:** We found that for this question, the output we want from the neural network is binary as $\{0, 1\}$ for each question answered. This is the classification problem. So far as we know, we can use binary cross entropy loss to do the classification problem well. Also our output layer activation function is sigmoid function which has the range of $(0, 1)$, we have met the prerequisite of $f(v; \theta) \geq 0$(Let the original input be vector v, and the student ability be θ). Our loss function has the following equation:

$$\min_{\theta} -\sum_{v \in S} v \log(f(v; \theta)) + (1 - v) \log(1 - f(v; \theta))$$

- **Figure:** Below is the results of finetuning with different hyperparameters for BCE Loss.

	Highest Validation Accuracy	Epoch Index for Accuracies
$k = 10$, learning rate=0.005	0.6852	87
$k = 10$, learning rate=0.01	0.6749	73
$k = 10$, learning rate=0.1	0.6931	13
$k = 50$, learning rate=0.005	0.6946	51
$k = 50$, learning rate=0.01	0.6939	17
$k = 50$, learning rate=0.1	0.6763	4
...	...	...

- **Demonstration:** We normally use L2 Loss for Regression problem, and L2 Loss has great penalty (quadratically) to result with big difference between ground truth and model output. This loss may over-emphasize the effect of outliers. Also by using BCE Loss, our number of epoch for early stopping will be less, compared with our result of L2 Loss in Part A.

2.2 Multi-layer Neural Network and Optimizer Selection

- **Description:** We have the hypothesis that deeper neural network might help to find the pattern of data accurately. Using sigmoid fucntion as activation function, we can have the larger hypothesis space than a normal linear model. We expect more weights helps to fit the data better.

  We then construct the following 4−layer neural network $f_4(v; \theta)$ and use loss function BCE above:

$$f_4(v; \theta) = j(W^{(4)} i(W^{(3)} h(W^{(2)} g(W^{(1)} v + b^{(1)}) + b^{(2)}) + b^{(3)}) + b^{(4)}) \in \mathbb{R}^{N_{\text{questions}}}$$

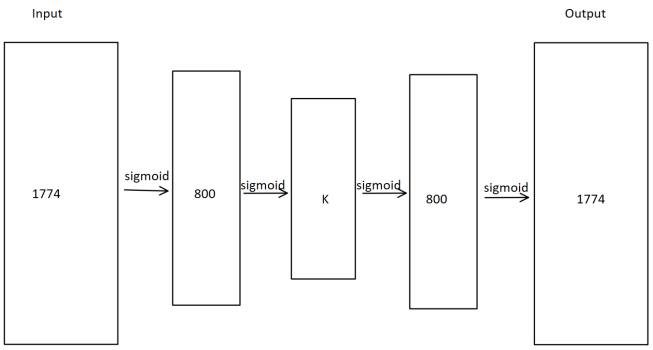
  for some activation functions g to j. Here we are using sigmoid activation functions for all. Here, $W^{(1)} \in \mathbb{R}^{800 \times N_{\text{questions}}}$, $W^{(2)} \in \mathbb{R}^{k \times 800}$, $W^{(3)} \in \mathbb{R}^{800 \times k}$ and $W^{(4)} \in \mathbb{R}^{N_{\text{questions}} \times 800}$ where $k \in \{10, 50, 100, 200, 500\}$ be the weight matrices and $b^{(3)}$, $b^{(4)}$ be the bias vector and other variables remain unchanged from the base-line model.

  And We also test for the 6−layer neural network $f_6(v; \theta)$:

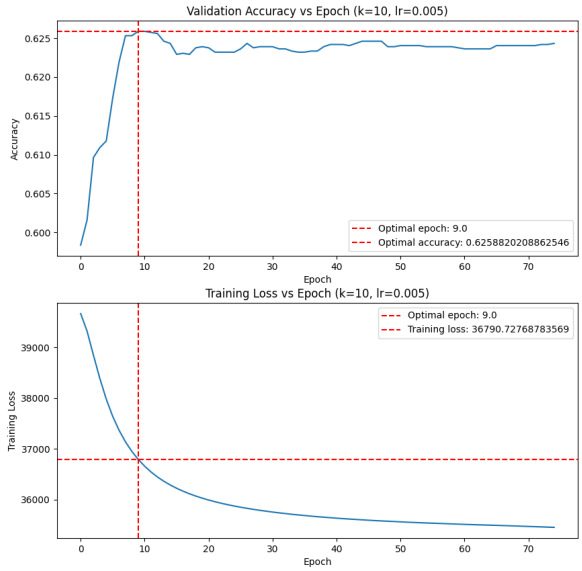
$$f_6(v; \theta) = m(W^{(6)}(l(W^{(5)} j(W^{(4)} i(W^{(3)} h(W^{(2)} g(W^{(1)} v + b^{(1)}) + b^{(2)}) + b^{(3)}) + b^{(4)}) + b^{(5)}) + b^{(6)}) \in \mathbb{R}^{N_{\text{questions}}}$$

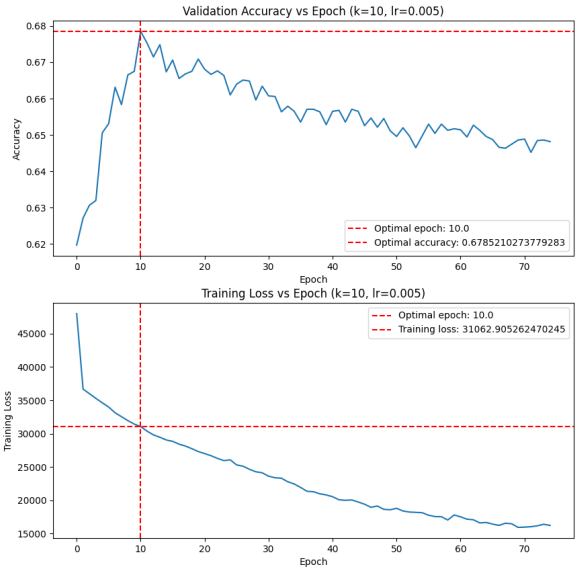
  for some activation functions $\{g, h, i, j, l, m\}$. Here we are using sigmoid activation functions for all. Here, $W^{(1)} \in \mathbb{R}^{1300 \times N_{\text{questions}}}$, $W^{(2)} \in \mathbb{R}^{800 \times 1300}$, $W^{(3)} \in \mathbb{R}^{k \times 800}$, $W^{(4)} \in \mathbb{R}^{800 \times k}$, $W^{(5)} \in \mathbb{R}^{1300 \times 800}$ $W^{(6)} \in \mathbb{R}^{N_{\text{questions}} \times 1300}$ where $k \in \{10, 50, 100, 200, 500\}$ be the weight matrices and $b^{(3)}$, $b^{(4)}$, $b^{(5)}$, $b^{(6)}$ be the bias vector and other variables remain unchanged from the base-line model.

- **Figure:** Below is the Diagram of 4-layer Neural Network:



- **Comparison and Demonstration:** Below is the graph of the model, we found that when the layer of neural network becomes more, it will lead to the problem of slow scholastic gradient descend. This is because gradients can become very small as they propagate backward through many layers during training, espcially for sigmoid function, leading to slow convergence or even stagnation during training as shown in our graph by our experiment shown in bottom left.





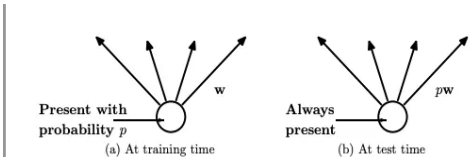
After searching for the solution, we introduce Adam Optimizer instead of SGD. Adam Optimizer has adaptive learning rate and it uses momentum-like behavior, which helps accelerate the optimization process, especially in the presence of flat regions like we saw the slow convergence area.

Above right is the result after applying Adam Optimizer:

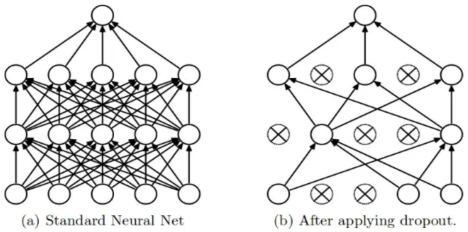
As you can see in the graph, when we use the same hyperparameter, Adam Optimizer decrease training loss faster than SGD while SGD has the effect of stagnation during training. By Adam Optimizer, we can reach highest validation accuracy faster. We can also decrease initial learning rate for Adam Optimizer if we want to find a optimal validation accuracy. As a result, out multi-layer neural networks **all** use Adam Optimizer.

2.3 Dropout

- **Description:**
  - **Define the way:** Dropout is a regulation method which helps to resolve the overfitting problems that arise in neural networks. It randomly selects the neurons and drops them out during training.
  - **Equations/Algorithm box:** Mathematical explanation of dropout: A unit (neuron) during training is present with a probability p and is connected to the next layer with weights 'w' ; (b) A unit during inference/prediction is always present and is connected to the next layer with weights, 'pw'.



  - **Expections:** Since by part (a) we get test accuracy < validation accuracy, therefore the model is a bit overfitting. To solve the overfitting problem, we use dropout, since dropout improves the model's generalization ability by reducing coadaptability between neurons. Therefore, we expect: (1) Reducation in Overfitting: we expect the test accuracy to increase after we apply this method. (2) Generalization: By the property of dropout, dropout helps the model learn more robust features that are not reliant on specific weights, thus enhancing generalization.
  - **Implementation:** We add the dropout between the two layers.
- **Figure/Diagram:**
  - **Figure of Dropout:**



(a) Standard Neural Net (b) After applying dropout.

2.4 Utilizing student ability from IRT

- **Description:**
  - **Define the way:** Enhance the predictive accuracy of neural network models in educational Settings by including student ability (theta) as an additional feature.
  - **Equations/Algorithm box:** Let the original input be vector v, and the student ability be θ, then the new input vector be $v = concatenate(v, \theta)$.
  - **Expectations:** Since by part (a) the irt model achieves good accuracy, and in the neural network we utilize a user-based data, therefore we can make use of the student ability(theta) in part (a), inject it into the input of our new model, therefore get more information. And there may exist a relation between student ability and the correctness of the answers. Therefore, we expect the model will achieve higher accuracy in predictions.
  - **Implementation:** The neural network model is extended to accept two inputs: (1)The original input data(x). (2)The student ability data (Theta), and we concatenate them together.
- **Figure/Diagram:**

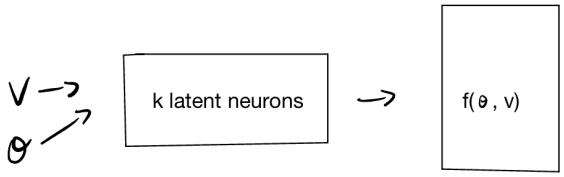


Figure of irt extension

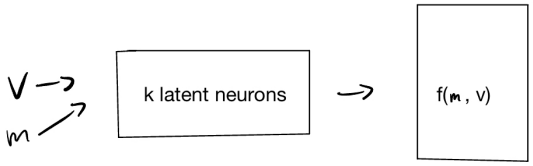


Figure of student meta extension

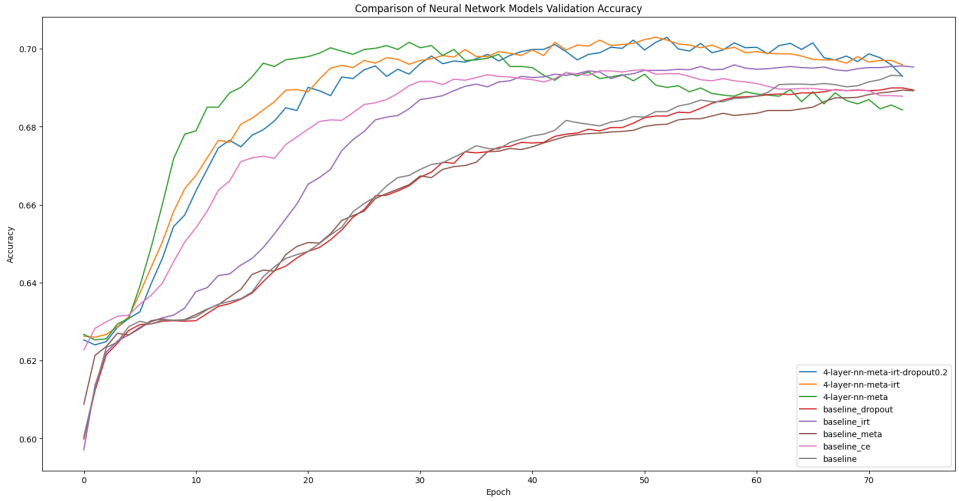
2.5 Utilizing student metadata

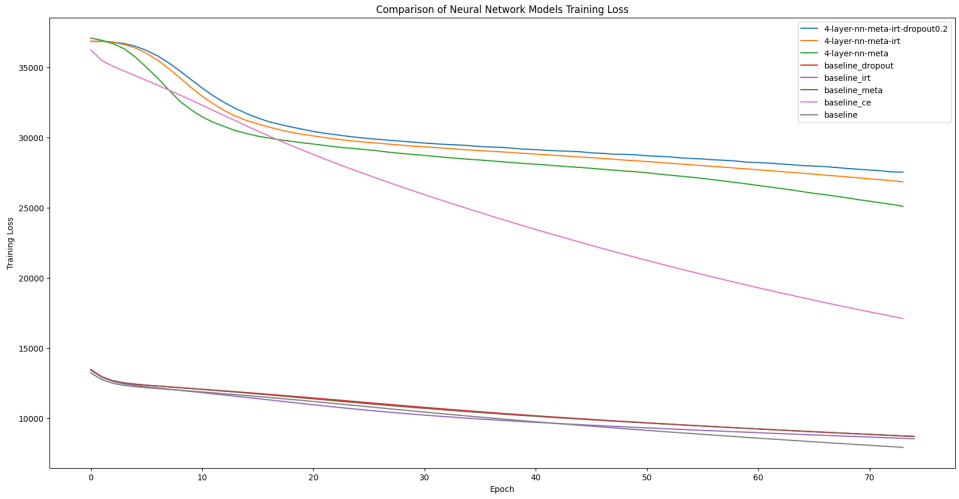
- **Description:**
  - **Define the way:** Enhance the predictive accuracy of neural network models in educational Settings by including student metadata as additional features.
  - **Equations/Algorithm box:** We also modify the data so that it fits our intention. For the *premium_pupil*, we set it to be -1 if there is no data. For the date of birth, we translate it into student age since it best suits our needs. And then we normalize the ages therefore the whole performance of the model won't be sensitive of this particular feature. Let the original input be vector v, and the student metadata be m, then the new input vector be $v = concatenate(v, m)$.
  - **Expectations:** Since we are given the student metadata which includes the student's age, gender, date of birth, and eligibility for free school meals or pupil premium(which shows the financial condition of the student), we can make use of those data since there may exist relations between those conditions and the students' correctness of the questions. For example, older students may be more familiar with some questions and therefore get a correct answer. However, by the observation of the data, we can find out there exist some incorrect data, for example, some student has date of birth which is greater than the current time. Therefore, we expect using these data will decrease the accuracy since the wrong data may cause errors in predictions.
  - **Implementation:** The neural network model is extended to accept two inputs: (1)The original input data(x). (2)The student meta data(m), and we concatenate them together.
- **Figure/Diagram:(see above section 5.4)**

2.6 Experiment Reports

Individual model performance: See appendix section 3.2. **Model Comparison:**

	Valid Accuracy	Test Accuracy
Baseline Model ($k = 50$, $lr = 0.005$, epoch $= 75$)	0.6932	0.6768
Baseline Model with L2 Regularization ($k = 50$, $lr = 0.005$, epoch $= 75$)	0.6905	0.6813
Baseline Model with Dropout ($k = 50$, $lr = 0.005$, epoch $= 98$)	0.6915	0.6808
Baseline Model with IRT ($k = 50$, $lr = 0.005$, epoch $= 86$)	0.6959	0.6980
Baseline Model with BCE Loss ($k = 50$, $lr = 0.005$, epoch $= 52$)	0.6946	0.6853
Baseline Model with BCE Loss, Student Meta ($k = 50$, $lr = 0.01$, epoch $= 27$)	0.6942	0.6844
Baseline Model with BCE Loss, Student Meta, Adam ($k = 100$, $lr = 0.0001$, epoch $= 61$)	0.6909	0.6898
4-Layer Sigmoid Model with BCE Loss, Student Meta (At First Layer) ($k = 50$, $lr = 0.0001$, epoch $= 40$)	0.6997	0.7011
6-Layer Sigmoid Model with BCE Loss, Student Meta (At First Layer) ($k = 100$, $lr = 0.0001$, epoch $= 29$)	0.6977	0.6955
4-Layer Sigmoid Model with BCE Loss, Student Meta (At Third Layer) ($k = 200$, $lr = 0.0001$, epoch $= 28$)	0.6988	0.6986
4-Layer Sigmoid Model with BCE Loss, Student Meta, IRT ($k = 10$, $lr = 0.0001$, epoch $= 53$)	0.7029	0.7042
4-Layer Sigmoid Model with BCE Loss, Student Meta, IRT,Dropout ($k = 10$, $lr = 0.0001$, epoch $= 54$)	0.7029	0.7005





- **Comparison:** Since for the baseline model, the validation accuracy is 0.6932, the test accuracy is 0.6768.

Then we can know that the validation accuracy of dropout version is lesser than the baseline, and the test accuracy of the dropout version is greater than the baseline model, then we can know it prevents overfitting and shows the generalization ability by regularization and fits our expectation.

For the irt extension on the neural network, both test and validation accuracy is greater than the original version, this meets our expectation by optimization.

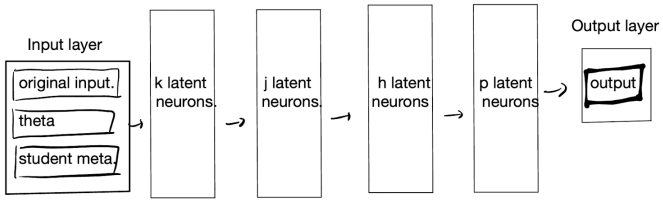
For binary cross entropy loss, our validation accuracy is 0.6946 for $k = 50$ and learning rate is 0.005. This validation accuracy is higher than 0.6932 for baseline. Also, in part A, we have the highest test accuracy of 0.6813 with L2 regularization. Our BCE loss improvement reach a test accuracy of 0.6853 which is higher than baseline. We early stop our training at epoch 52 to decrease the effect of over-fitting on training dataset.

For the student meta extension, the validation accuracy is less than the original which fits our exception. This doesn't optimize well may be due to the data quality.

For multi-layers neural network, we are testing on several setup. First is that we are adding Student Meta and Adam for optimization for all multi-layers neural network. We compared 4-layers and 6-layers setup, 4-layers neural network has higher validation accuracy than 6-layer, so we follow 4-layer neural network as multi-layer neural network baseline. We then compare the place that we concatenate student meta to the neural network and find that concatenate at first layer has validation accuracy of 0.6997 which is higher than 0.6988 of concatenating at third layer. We also experiment adding IRT to our model, it follows IRT is the optimization to the neural newtwork. Finally, we add dropout layer, it has lower test accuracy than validation accuracy which is expected.

From the graphs, we also can see the pattern, for baseline model, they are using SGD and they converge slowly for training loss as you can see in the second graph. Even when we replace with binary cross entropy loss, base pipeline training loss converges slowly compared with deeper neural network with Adam Optimizer. Indeed, Adam Optimizer is the optimization process.

2.7 Model Architecture (which performs the best)



Following is our final 4−layer neural network $f_4(v;\theta)$:

$$\min_{\theta} -\sum_{v \in S} v \log(f_4(v;\theta)) + (1-v)\log(1-f_4(v;\theta))$$

$$f_4(v;\theta) = j(W^{(4)}i(W^{(3)}h(W^{(2)}g(W^{(1)}concatenate(concatenate(v,m),\theta) + b^{(1)}) + b^{(2)}) + b^{(3)}) + b^{(4)}) \in \mathbb{R}^{N_{\texttt{questions}}}$$

for some activation functions g to j. Here we are using sigmoid activation functions for all. Here, $W^{(1)} \in \mathbb{R}^{800 \times (N_{\texttt{questions}} + |m| + |v|)}$, $W^{(2)} \in \mathbb{R}^{k \times 800}$, $W^{(3)} \in \mathbb{R}^{800 \times k}$ and $W^{(4)} \in \mathbb{R}^{N_{\texttt{questions}} \times 800}$ where $k \in \{10, 50, 100, 200, 500\}$ be the weight matrices and $b^{(3)}$, $b^{(4)}$ be the bias vector and other variables remain unchanged from the base-line model. Also, $concatenate()$ function is defined in section 2.4, meaning adding entries right after last entry of vector.

2.8 Limitations

1. dropout extension: The validation accuracy performs worse than the baseline mode, this may because the drop of the neurons is random which may drop some important neurons. Solution: Adjust the dropout rate to find some lower one that also attains optimal accuracy.
2. irt extension: If the performance of the irt model is bad, then utlizing theta in our model may perform bad result since bias of the theta trained by the irt model may increase the bias of our extension model. Solution: Train the irt model better to get good performance.
3. student metadata extension: The error in the metadata may increase the bias of our output since inaccurate or incomplete metadata can lead to misleading conclusions. Solution: Improve the data quality.
4. binary cross entropy extension :If the dataset has a class imbalance (for example, if a majority of the answers are correct or incorrect), BCE could lead to a model that is biased towards the majority class since there is less penalty on minority class. Solution: increasing the number of minority class in the training set.
5. Multilayer extension:
If the data set is small, then may increase the probability of overfitting since a larger number of parameters in a multilayer model can lead to learning noise rather than the underlying pattern. Solution: Increase the quantity of data.

3 Appendix

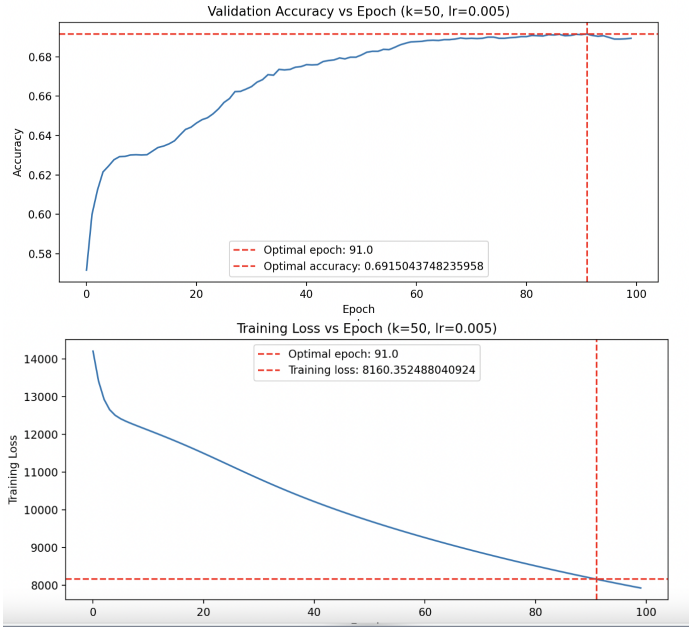
3.1 Contribution:

- Part A Chutong Li: Q1, Q2, Q4; Hanrui Fan: Q3
- Part B Chutong Li: dropout extension, irt extension, student meta extension; Hanrui Fan: Binary Cross Entropy Loss, Multi-layer Neural Network
- Confirmation: All members of the team have submitted the course evaluations.

3.2 Figures:

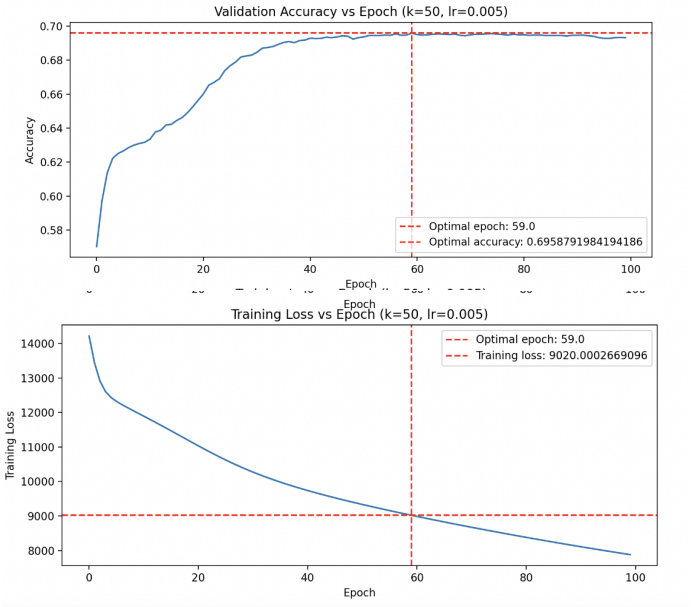
Dropout

- The validation accuracy is 0.6915043748235958, the test accuracy is 0.68077900084674



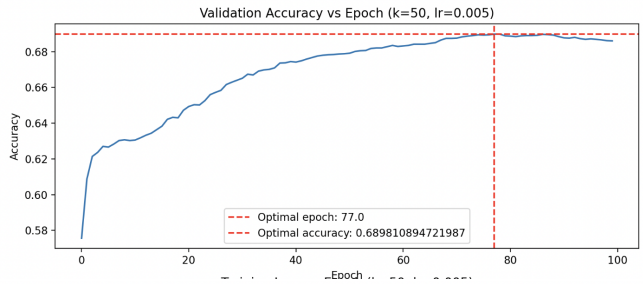
Student ability

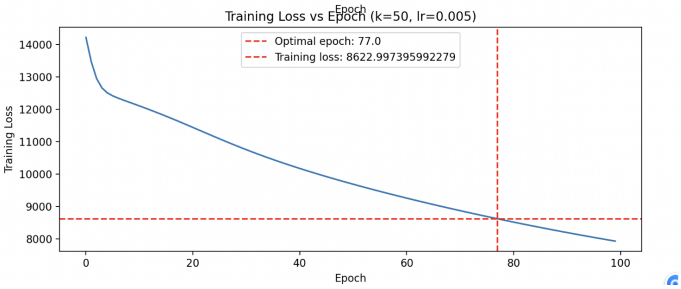
- The validation accuracy is 0.6958791984194186, the test accuracy is 0.6979960485464296



Student metadata

- The validation accuracy is 0.689810894721987, the test accuracy is 0.6833192209991532





Multi-layer Neural Network and Optimizer

- The result figure is shown above under the same title of section 2.2 and the report result is in the model comparison section.