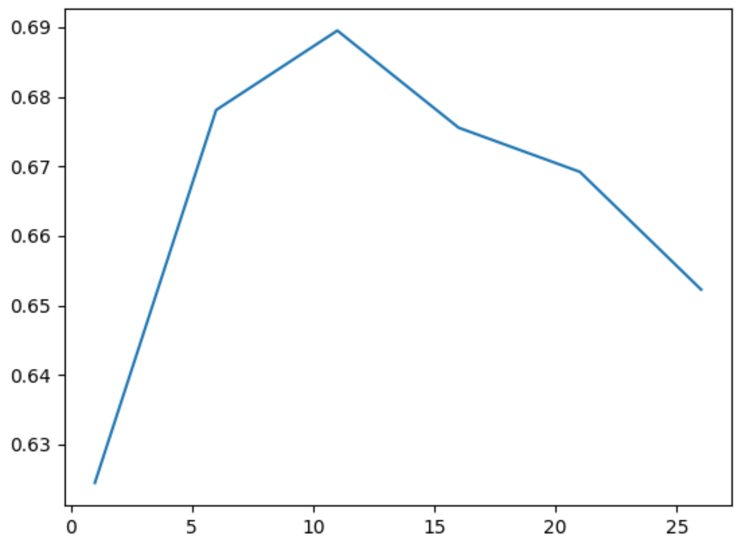
Final Project

Contributions

Juan Deng: 1/3
Xuan Du: 1/3
Maya Shankar: 1/3
We mostly worked together for the entire homework.

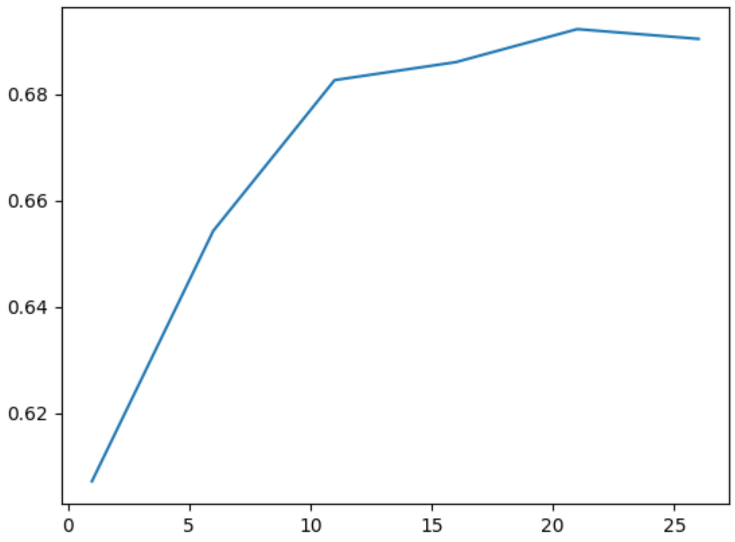
Part A

1. **KNN** (a) Accuracy on validation set by k-value based on user similarity



(b) $k* = 11$ has the highest accuracy on the validation set, with accuracy 0.684165961 on the test set.

(c) The underlying assumption with item-based filtering is that if questions A and B are answered correctly by the same set of students, they have the same correctness. Accuracy on validation set by k-value based on user similarity



$k* = 21$ has the highest accuracy on the validation set, with accuracy 0.68162574 on the test set.

(d) Finding similarities by-user has slightly higher accuracy on the test set with the chosen k* (0.684165961 > 0.68162574).

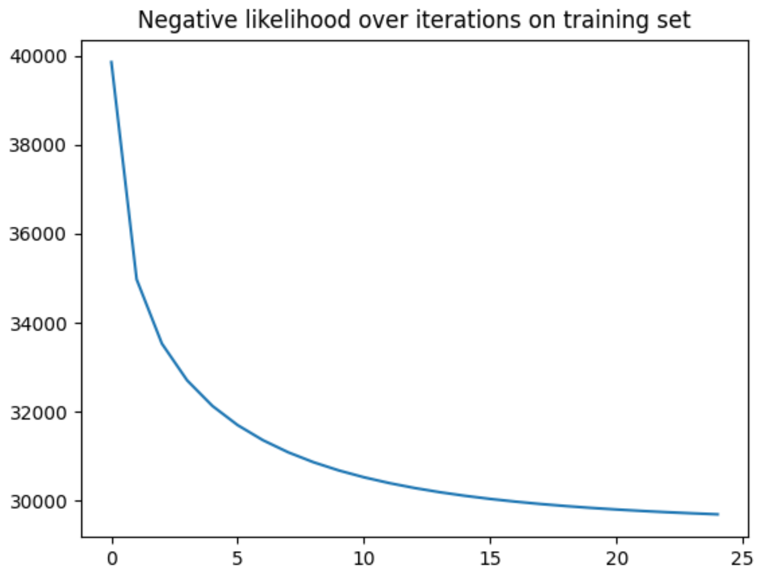
(e) A knn model is good for predicting on large data sets. However, when it is predicting a large data set, the model will have a very long running time and relatively low accuracy (around 68%).
The quality of data set is not very good, since the data set here contains many missing values. Knn is sensitive to irrelevant features and missing data, so this makes the prediction less stable.

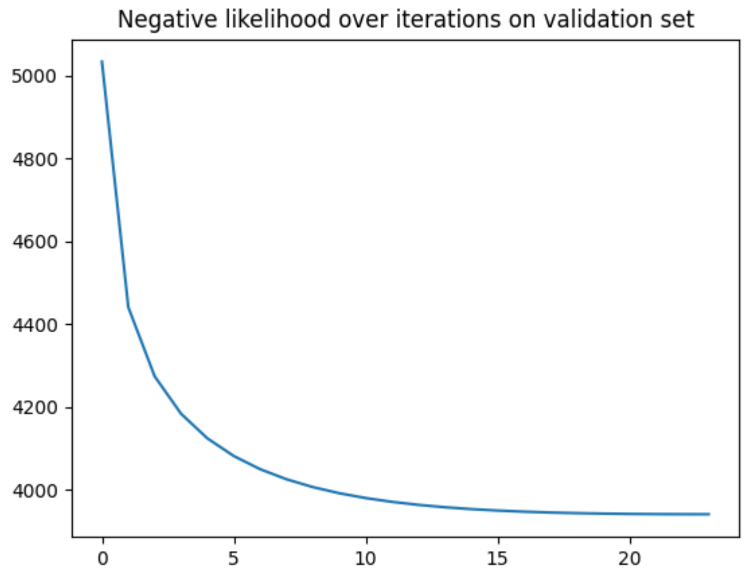
2. **IRT** (a) derive log-likelihood
log p(C | θ, β)
$= \log (\prod_i \prod_j p(c_{ij}=1|\ \theta_i,\beta_j)^{c_{ij}} \cdot (1 - p(c_{ij}=1|\ \theta_i,\beta_j)^{(1-c_{ij})})$
$= \log (\prod_i \prod_j (\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)})^{c_{ij}} \cdot (1 - \frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)})^{(1-c_{ij})}))$
$= \sum_i \sum_j (c_{ij}) \log (\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)}) + (1 - c_{ij}) \log (1 - \frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)})$

Then, taking derivative with respect to θ_i,
$\frac{\partial logp(C|\theta,\beta)}{\partial\theta_i}$
$= \sum_j \frac{\partial}{\partial\theta_i} ((c_{ij})\log(\frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)} + (1 - c_{ij})\log(1 - \frac{exp(\theta_i-\beta_j)}{1+exp(\theta_i-\beta_j)}))$
$= \sum_j (c_{ij}) \frac{exp(\beta_j)}{exp(\beta_j)+exp(\theta_i)} - (1 - c_{ij}) \frac{exp(\theta_i)}{exp(\beta_j)+exp(\theta_i)}$
Then, taking derivative with respect to β_j,
$\frac{\partial logp(C|\theta,\beta)}{\partial\beta_j}$

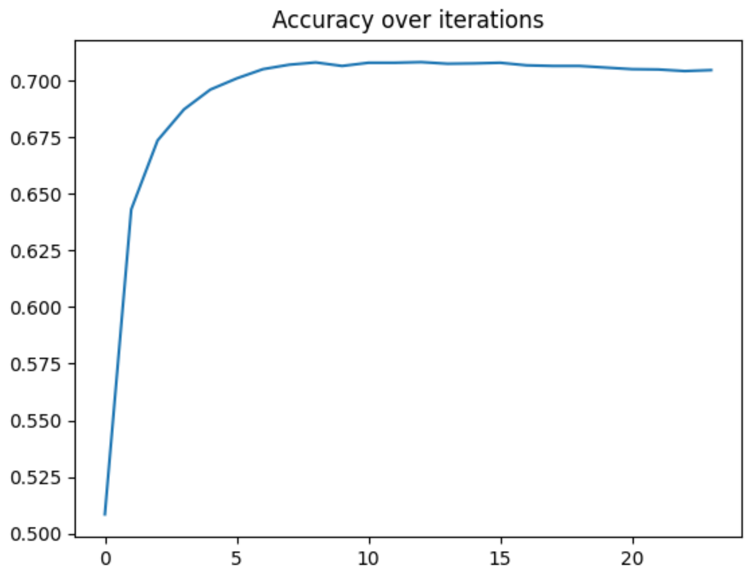
$= \sum_i \frac{\partial}{\partial \beta_j}((\mathrm{c}_{ij})\log(\frac{exp(\theta_i - \beta_j)}{1+exp(\theta_i - \beta_j)} + (1 - \mathrm{c}_{ij})\log(1 - \frac{exp(\theta_i - \beta_j)}{1+exp(\theta_i - \beta_j)}))$
$= \sum_i (\mathrm{c}_{ij})\ (- \frac{exp(\beta_j)}{exp(\beta_j)+exp(\theta_i)}) + (1 - \mathrm{c}_{ij})\ (\frac{exp(\theta_i)}{exp(\beta_j)+exp(\theta_i)})$
$= \sum_i (1 - \mathrm{c}_{ij})\ (\frac{exp(\theta_i)}{exp(\beta_j)+exp(\theta_i)}) - (\mathrm{c}_{ij})\ (\frac{exp(\beta_j)}{exp(\beta_j)+exp(\theta_i)})$

(b) The optimal parameter values were:
Learning rate = 0.005, Number of iterations = 24



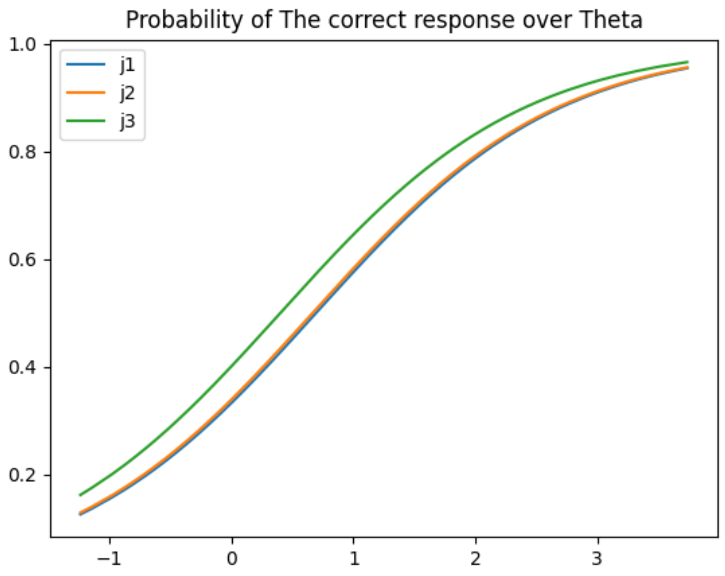


(c) Final Accuracy on Validation set: 0.7046288



Final Accuracy on Test set: 0.7092859

(d) The graph shows the probability of the correct response over students' abilities for three different questions. The probability of answering the question correctly increases as theta (student abilities) increases and the curves show a sigmoid shaped relationship. All three questions have similar shape and values suggest they have similar difficulties.



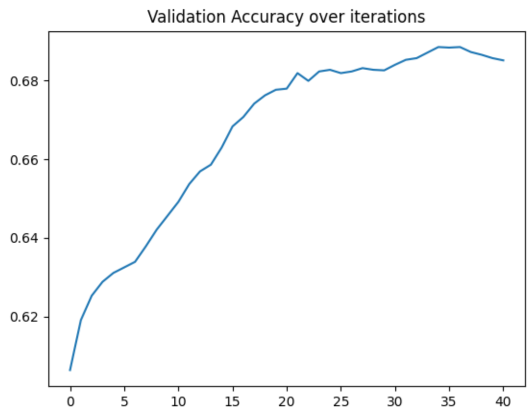
3. **Neural Networks**

   We have chosen Option (ii): Neural Networks

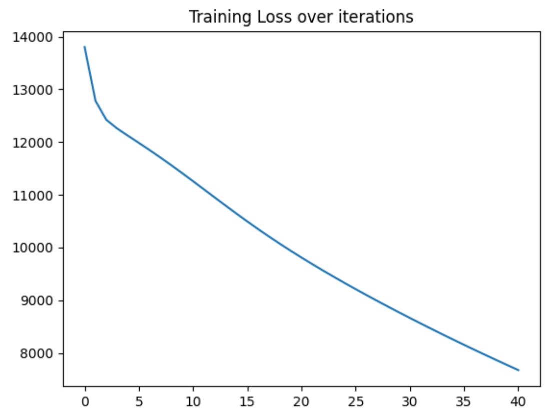
   (a) 1. ALS is used for unsupervised learning data, and neural nets for supervised learning
   2. ALS will decompose the sparse matrix into two smaller matrices while neural nets will use autoencoders and hidden layers.
   3. ALS will factorize the given matrix into two factors and then fix one parameter and alternate with others, while neural nets will update all parameters at the same time and use gradient descent.

   (c) We trained k for the highest validation accuracy, then arrived at these optimal parameters:
   $k* = 50$, lr $= 0.01$, run for 41 iterations
   Max Epoch: 41
   Accuracy on validation set: 0.6910810047981937

   (d) Final accuracy on validation set, with 41 iterations: 0.6841659610499576



   Final loss on training data, with 40 iterations: 7641.957781



Final accuracy on test set: 0.6850127011007621

(e) Our chosen lambda value is 0.001
Final validation accuracy: 0.6910810047981937
Final test accuracy: 0.6861416878351679
The final test accuracy with penalty is slightly higher than in part (d). The model performs better with a regularization penalty.

4. **Ensemble**
Final accuracy on validation set: 0.7019475021168501
Final accuracy on test set: 0.7011007620660458
The random sampling process for bagging:

1 generate a random index from {0, sample size}

2 pick out that sample with index from previous step from the original data-set and put into a new data-set

3 repeat the process for 3 times to generate 3 new data-sets with the same size as the original data set

The ensemble process:

(a) Train the base models (kNN, IRT and neural network) on the new data sets (3 times).

(b) Obtain the predictions (nine lists of probabilities one from each base model).

(c) Average the predictions, and sort the averages over/under the threshold (0.5) to get a list of predictions with value 1 or 0.

(d) Evaluate the accuracy of the processed predictions on the validation and test sets.

We do not see a better performance using the ensemble.
Validation Final Accuracy: 0.6718882303132938

Test Final Accuracy: 0.6711826136042901

We obtained a final accuracy only slightly higher than the Knn and neural network models, but lower than the IRT model. The IRT model has a higher accuracy on its own than te ensemble does.
Since we use the average predicted correctness, the accuracy is also bounded by the the highest accuracy from the base models. The base models have their limitations, so when we average their performance we are trying to reduce the impacts from these limitations.

Part B

Description

In part B, we decided to modify the neural network that we worked on for Question 3. We tried using different activation functions for the layers, adding more layers to extend our neural network model, and splitting our dataset by different features.

Trial Method A: Using different activation functions for input layer and output layer
In this part, we implemented combinations of different activation functions like logSigmoid() and ReLU(). We included both the input layer and output layer when modifying the activation functions.

Trial Method B: Adding more layers to extend our neural network model
In this part, we added more layers to capture more details from the given dataset. We tried out models with two and three hidden layers, as opposed to the base model's one hidden layer.

Trial Method C: Splitting our dataset by different features
In this part, we split our dataset by different features like gender and premium pupil. These were provided as additional metadata.

For a given user v $\in {}^{N_{questions}}$ from a set of users S, we take:

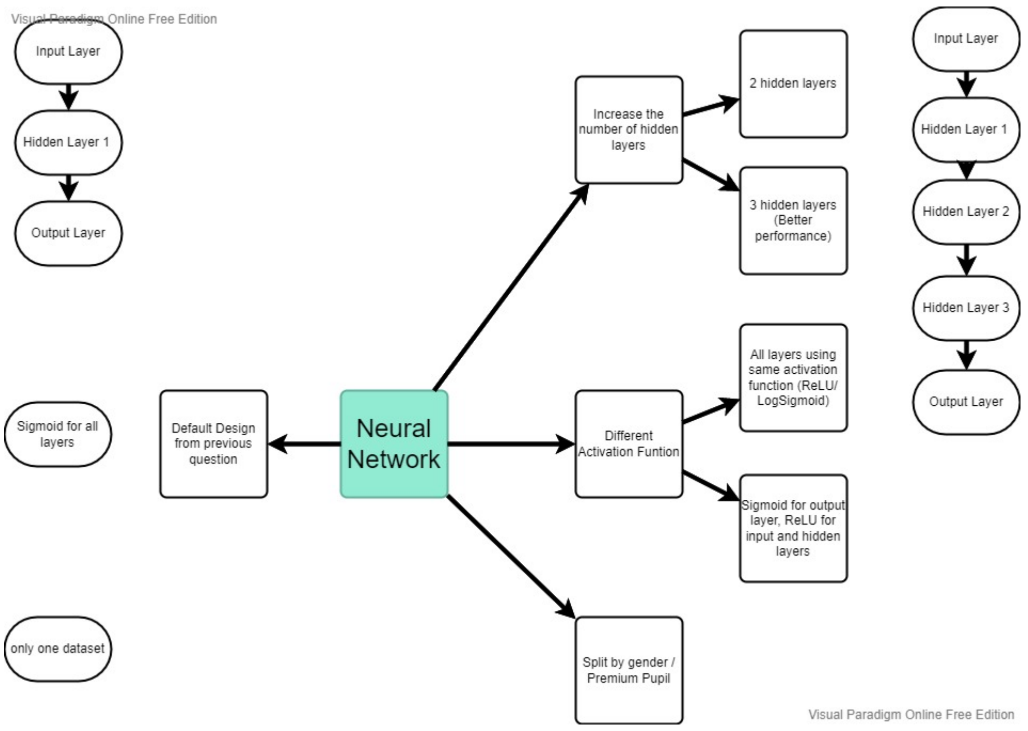
$$\min_{\theta} \sum_{v \in S} \|v - f(v;\theta)\|_2^2 + \frac{\lambda}{2}(\|W^{(1)}\|_F^2 + \|W^{(2)}\|_F^2 + \|W^{(3)}\|_F^2 + \|W^{(4)}\|_F^2)$$

where

$$f(v;\theta) = g(W^{(3)} i(W^{(2)} j(W^{(2)} k(W^{(2)} v + b^{(1)}) + b^{(2)}) + b^{(3)}) + b^{(4)})$$

for ReLU() activation functions i, j, and k and Sigmoid() activation function g. With these two improvements, the final validation accuracy is higher that any other combination of added layers and activation functions.

Visualization


Visual Paradigm Online Free Edition
Input Layer
Hidden Layer 1
Output Layer
Sigmoid for all layers
only one dataset
Default Design from previous question
Neural Network
Increase the number of hidden layers
2 hidden layers
3 hidden layers (Better performance)
Different Activation Funtion
All layers using same activation function (ReLU/ LogSigmoid)
Sigmoid for output layer, ReLU for input and hidden layers
Split by gender / Premium Pupil
Input Layer
Hidden Layer 1
Hidden Layer 2
Hidden Layer 3
Output Layer
Visual Paradigm Online Free Edition

Modifications and Comparisons to the Base Model

Activation Functions

When using the same activation function (ReLU, LogSigmoid) for all layers, we got relatively low accuracy (Fig 1, Fig 2). Then, we have tried to apply different activation functions to the output layer. As shown in Fig. 3, we have used Sigmoid() for the output layer and ReLU() for all other layers. Luckily, this resulted in much higher accuracy.

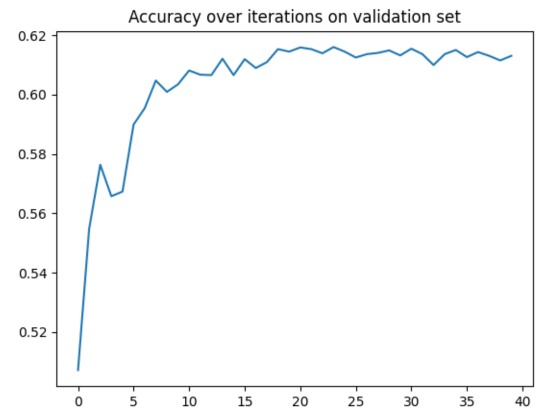


Fig 1. All Layers using ReLU as activation function
Max Accuracy at iteration 7:0.6128986734405871

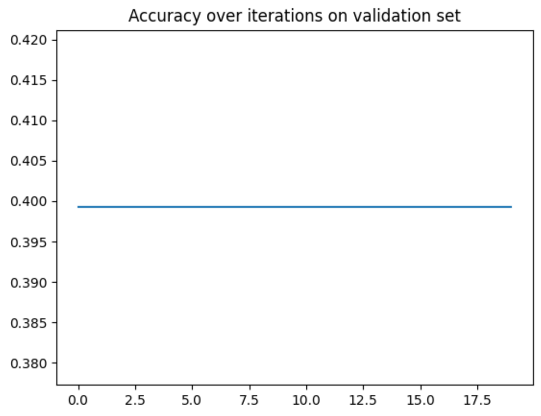


Fig 2. All layers using LogSigmoid as activation function
Max Accuracy at iteration 7:0.39923793395427604

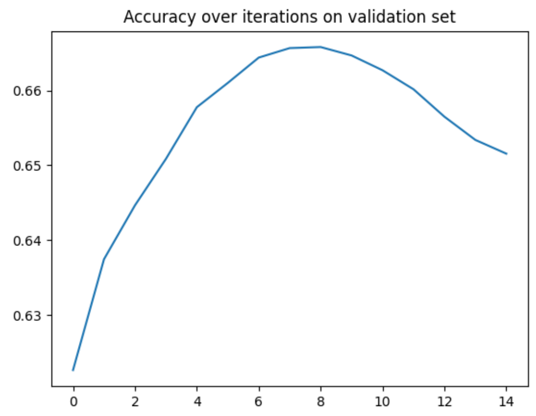


Fig 3. Output layer using Sigmoid, hidden layers using ReLU
Max Accuracy at iteration 8: 0.661162856336438

Adding Layers
Inner layer using ReLU:
When we tried a 4-layer net (two hidden layers), shown in Fig. 4, we had a higher accuracy than the 3-layer (the baseline model). When we tried a 5-layer net (three hidden layers) as shown in Fig. 5, we had slightly higher accuracy than the 4-layer model.

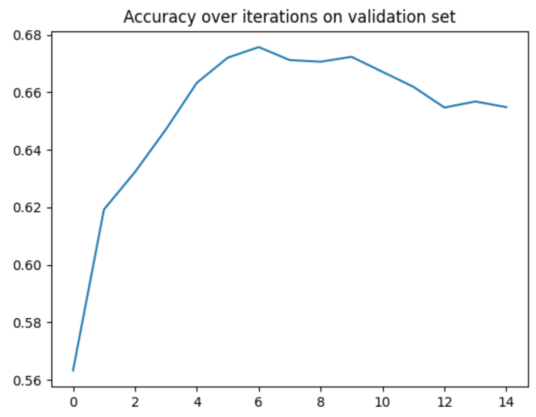


Fig 4. Two hidden layers (ReLU)
Max accuracy at iteration 9: 0.6754163138583121

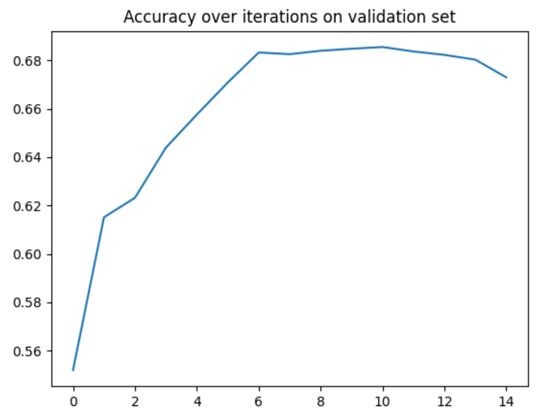


Fig 5. Three hidden layers (ReLU)
Max accuracy at iteration 10: 0.6858594411515665

Figure 6 shows the effect of having three inner layers activated by Sigmoid():

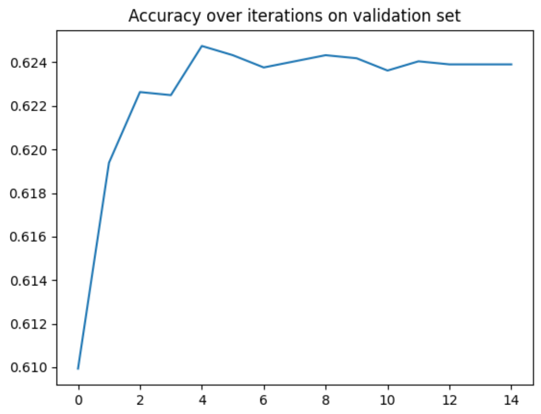


Fig 6. Three hidden layers (Sigmoid)

Split data by features
When we tried to spilt the users by premium pupil (does the user have a premium pupil designation in the metadata). The data set was split into three sets. The first set contains all students state they are not premium pupils. The second group contains all students are premium pupils and the last group contains students don't have information about premium pupil. The validation accuracy fluctuated (Fig 7). When we tried to split the dataset by gender of user, we faced the same problem.

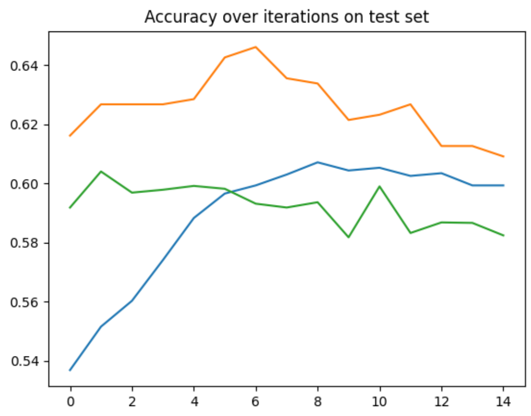
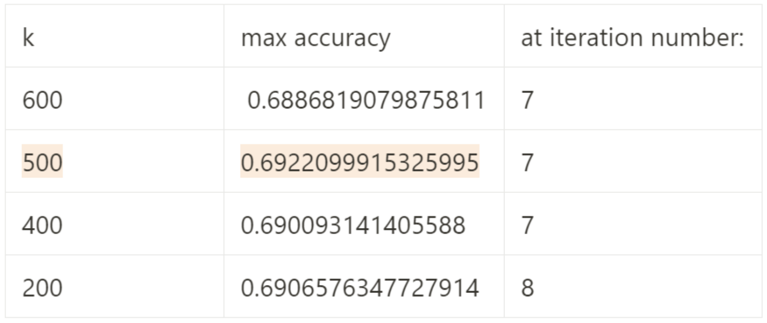


Fig 7. Split by premium pupil. Green represents student without "premium pupil" information. Orange lines represent accuracy for premium pupil input "1" and blue line for input "0".

As a result, we decided to abandon splitting and improve our neural network model by using ReLU as the activation functions for input layer and Sigmoid for output layer with 3 hidden layers (5-layer net).

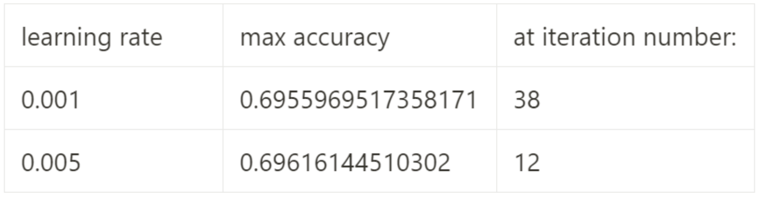
Tuning the hyperparameters
First, we tried to tune the k value with fixed learning rate and lambda.



k	max accuracy	at iteration number:
600	0.6886819079875811	7
500	0.6922099915325995	7
400	0.690093141405588	7
200	0.6906576347727914	8

From the table, we observe that the max accuracy occurs with k = 500, learning rate = 0.01, and lambda = 0.001.

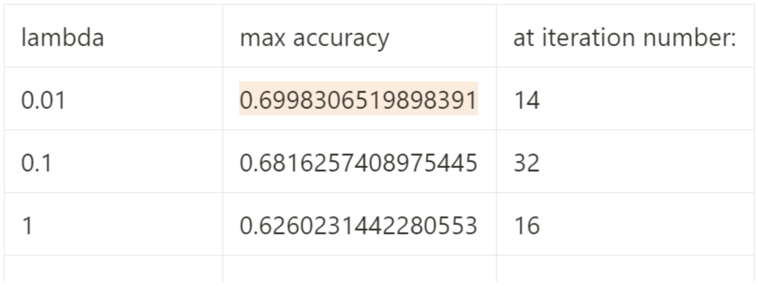
Then, we tried to tune the learning rate with k = 500, lambda = 0.001.



learning rate	max accuracy	at iteration number:
0.001	0.6955969517358171	38
0.005	0.69616144510302	12

Learning rate = 0.005 gave a better result.

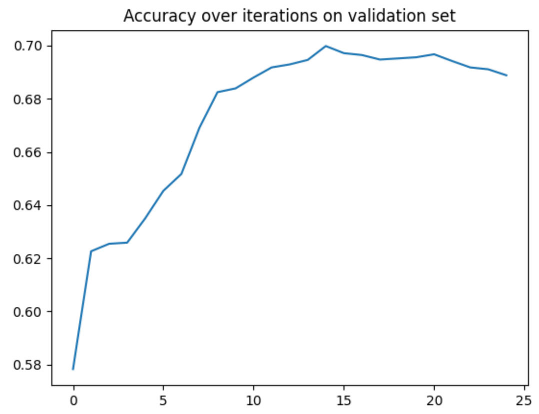
Last, we tuned the lambda with k = 500, learning rate = 0.005:

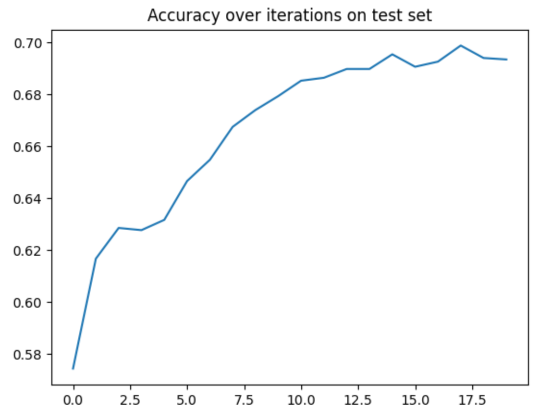


lambda	max accuracy	at iteration number:
0.01	0.6998306519898391	14
0.1	0.6816257408975445	32
1	0.6260231442280553	16

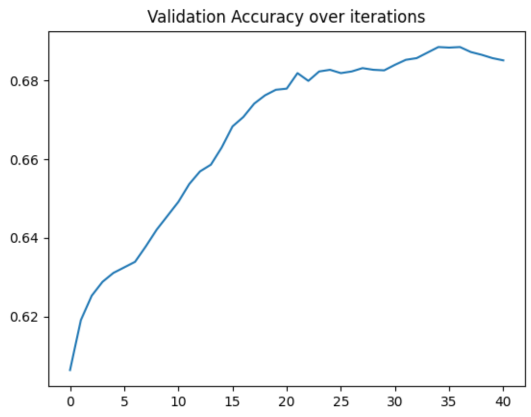
Lambda = 0.01 gave the highest accuracy.

Our final model uses ReLU for the inner layers and Sigmoid for the output layer with 3 hidden layers. Our hyperparameters were:
K = 500, learning rate = 0.005, lambda = 0.01.
We got a validation accuracy of **0.6998306519898391** and test accuracy of **0.698842788597234**





This gives a slightly higher test accuracy compared with the baseline model (graph below).



Base model: All layers using Sigmoid, one hidden layer (the same as Part A, Q3):
Final accuracy on validation set, with 41 iterations: 0.6841659610499576.
Final Test accuracy: 0.6861416878351679

Limitations

Since there are many features in our dataset besides user id and question id, it is hard to combine all information to make the predictions with a 2D sparse matrix and a neural network. We tried to split our dataset by features to improve the final accuracy, but the accuracy did not improve.

Another limitation is the regularization process. In our neural network model, we have used L2 regularization. This is the same as the base model since we did not find any alternatives that might work better.

Finally, we only applied different activation functions for input layers and output layers, and we did not try to apply different hyperparameters to different layers. For example, we could have used different k-values for each hidden layer. Since the layers use nn.linear() to connect to each other, we can also try applying different linking methods in the future.

References

1.PyTorch Activation Functions – ReLU, Leaky ReLU, Sigmoid, Tanh and Softmax, by Palash Sharma, March 10, 2021
https://machinelearningknowledge.ai/pytorch-activation-functions-relu-leaky-relu-sigmoid-tanh-and-softmax/
How to improve performance of Neural Networks
https://d4datascience.in/2016/09/29/fbf/

We used Visual Paradigm Online to generate the graph.