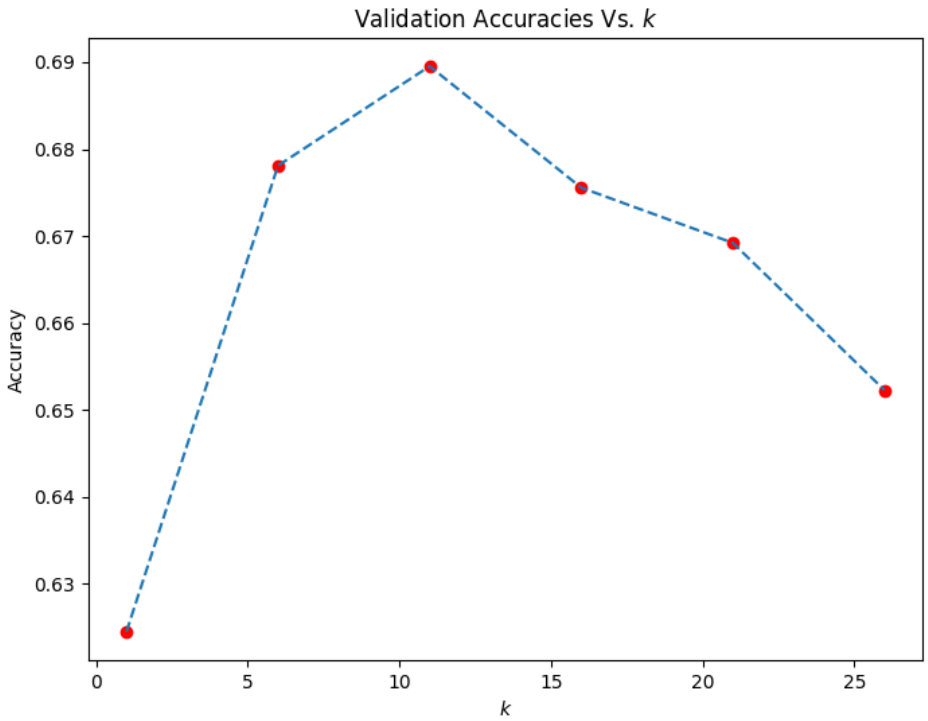
# Part A

## 1

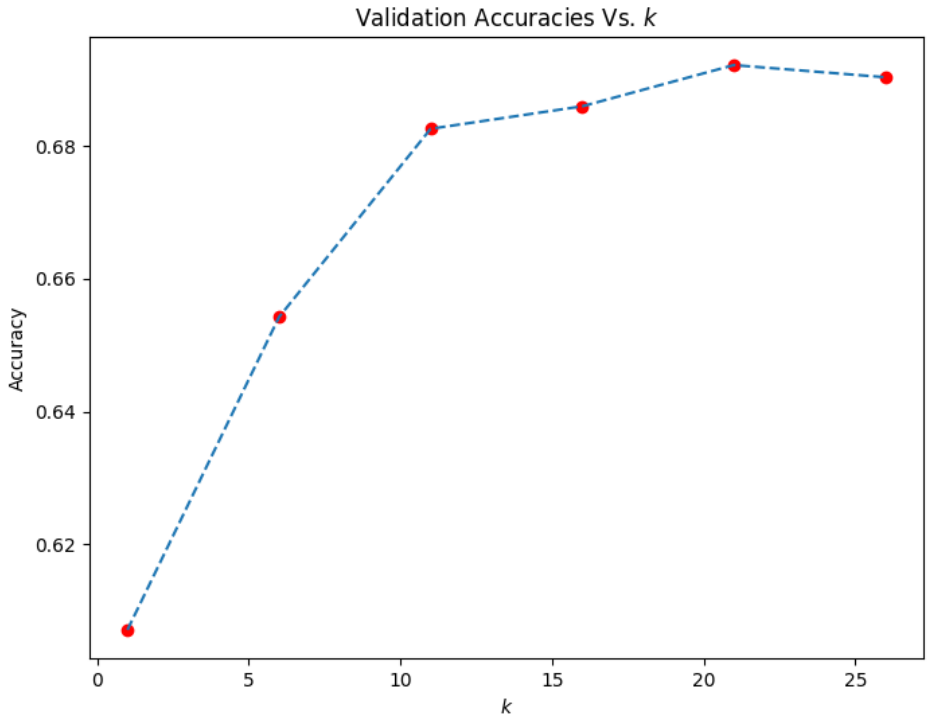
### a)




### b)

- $k^* = 11$
- User-based filtering test accuracy: 0.6841659610499576

## c)




- $k^* = 21$
- Item-based filtering test accuracy: 0.6683601467682755

The underlying assumption is for two questions 1 and 2, if question 1 has the same accuracy pattern across all users as question 2, question 1's accuracy pattern matches that of question 2.

## d)

| **Filtering Type** | **Test Accuracy** |
|---|---|
| User-Based | 0.6841659610499576 |
| Item-Based | 0.6683601467682755 |

User-based collaborative filtering performs better than item-based collaborative filtering.

## e)

1. As the number of questions and/or users increases, the distances between points will be similar—the curse of dimensionality. This makes it harder to distinguish between similar points.
2. kNN has a high test-time computational cost and will require more memory to store data for more users and questions.

# 2

## a)

Let $N$ be the number of students, and $M$ be the number of questions in the matrix $\boldsymbol{C} \in \mathbb{R}^{N \times M}$.

$$p(c_{ij} = 1 \mid \theta_i, \beta_j) = \frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)}$$
$$p(c_{ij} = 0 \mid \theta_i, \beta_j) = 1 - p(c_{ij} = 1 \mid \theta_i, \beta_j) = \frac{1}{1 + \exp(\theta_i - \beta_j)}$$

$$\begin{aligned}
\ell = \ln p(c_{ij} \mid \boldsymbol{\theta}, \boldsymbol{\beta}) &= \ln \left[ \prod_{i=1}^{N} \prod_{j=1}^{M} p(c_{ij} \mid \theta_i, \beta_j) \right] \\
&= \ln \left[ \prod_{i=1}^{N} \prod_{j=1}^{M} \left( \frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)} \right)^{c_{ij}} \left( \frac{1}{1 + \exp(\theta_i - \beta_j)} \right)^{1 - c_{ij}} \right] \\
&= \sum_{i=1}^{N} \sum_{j=1}^{M} \Big[ c_{ij} \left[ \theta_i - \beta_j - \ln(1 + \exp(\theta_i - \beta_j)) \right] + (1 - c_{ij}) \left[ - \ln(1 + \exp(\theta_i - \beta_j)) \right] \Big] \\
&= \sum_{i=1}^{N} \sum_{j=1}^{M} \Big[ c_{ij} (\theta_i - \beta_j) - \ln \left[ 1 + \exp(\theta_i - \beta_j) \right] \Big]
\end{aligned}$$

$$\frac{\partial \ell}{\partial \theta_i} = \sum_{j} \left[ c_{ij} - \frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)} \right]$$

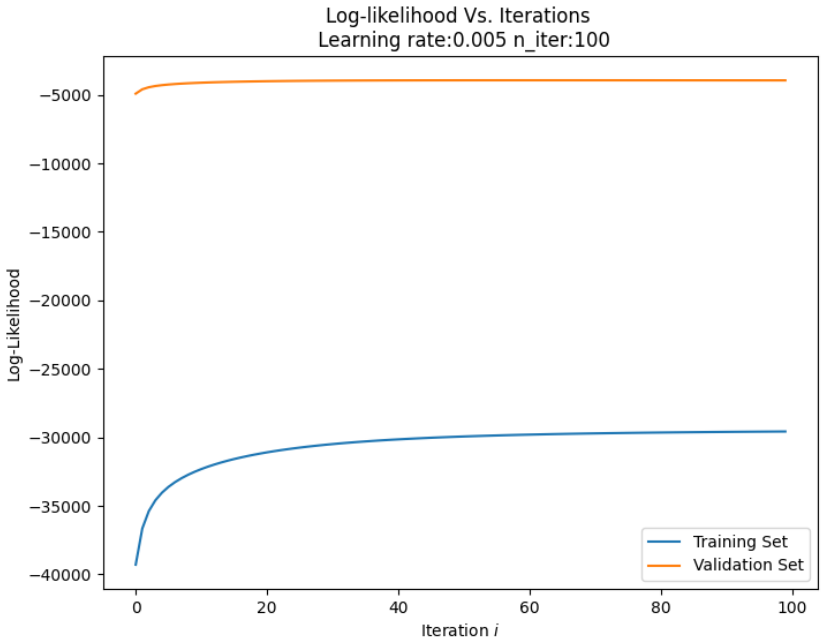
$$\frac{\partial \ell}{\partial \beta_j} = \sum_{i} \left[ -c_{ij} + \frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)} \right]$$

## b)

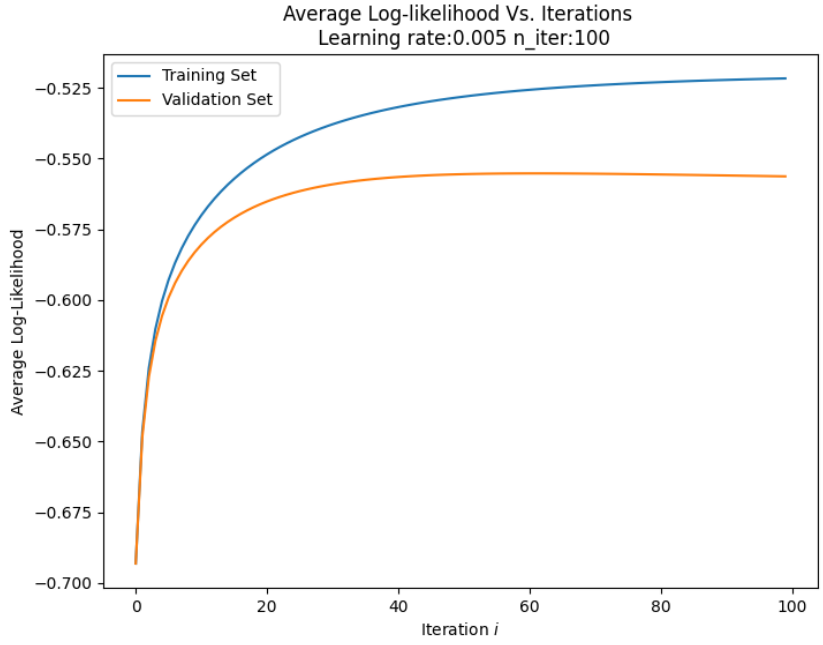
Hyperparameters:

- Learning Rate: 0.005
- Iterations: 100

## Log-Likelihood



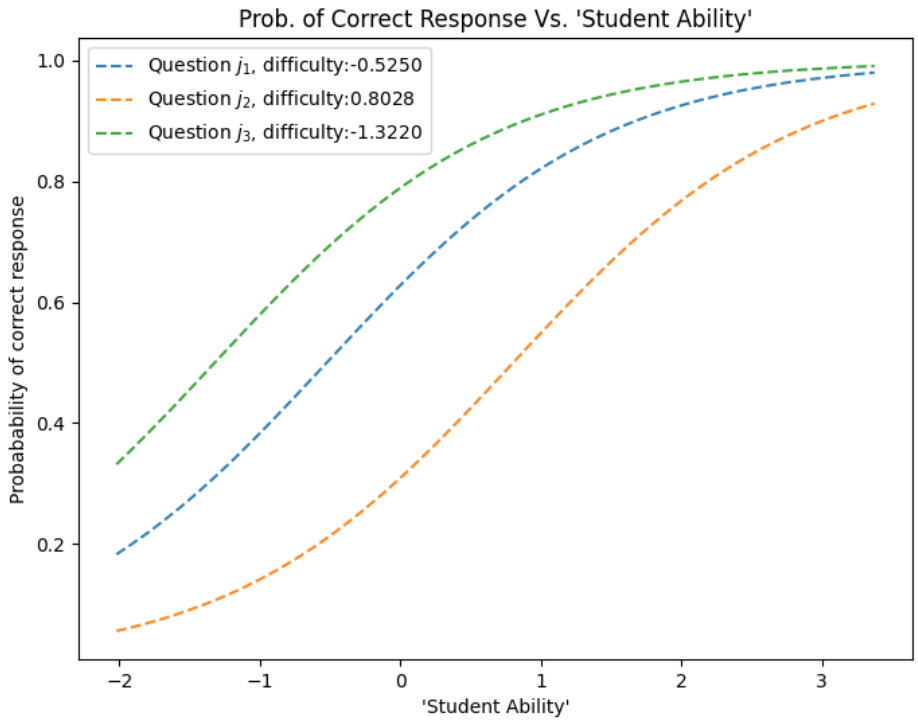

## Average Log-Likelihood



**c)**

| Validation Accuracy | Test Accuracy |
|---|---|
| 0.7396979960485465 | 0.7067456957380751 |

**d)**




The shape of the curve looks similar to a sigmoid (monotonically increasing), where correct response probability increases with increasing student ability and decreases with increasing question difficulty.

The curves represent the probability of getting the correct response on a question as a function of the question difficulty and student ability.

# 3

## a)

1. Matrix completion projects data onto a linear subspace, but deep nonlinear autoencoders can project data onto a nonlinear manifold.
2. Matrix completion uses ALS to optimize the two parameter matrices, fixing one matrix and optimizing the other.
3. Neural networks on the other hand use backpropagation to update the weights within the layers.

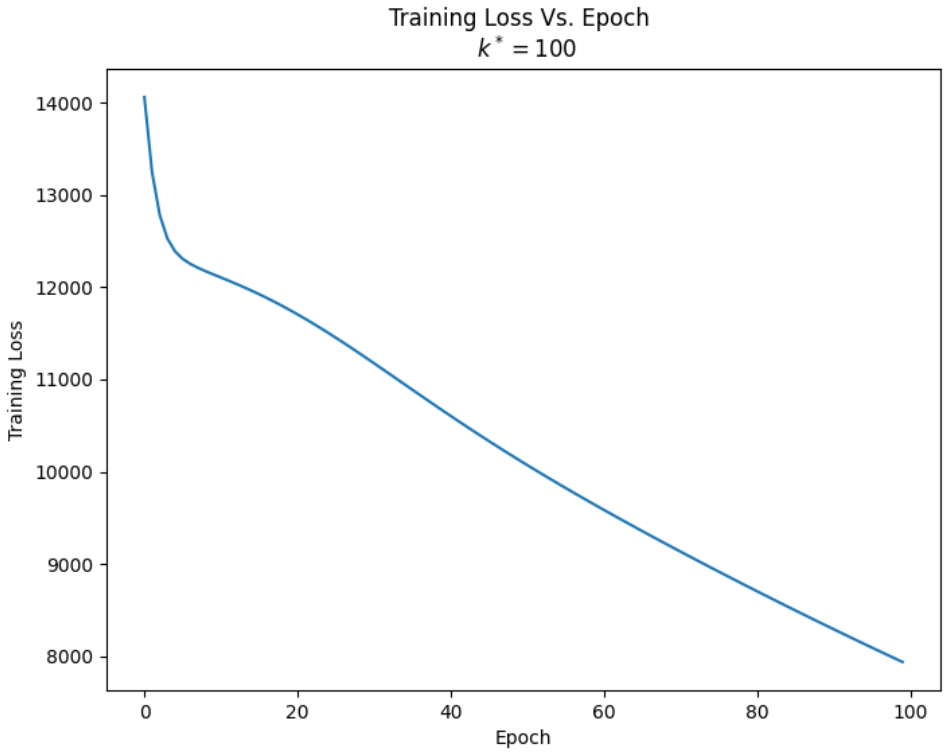
## b)

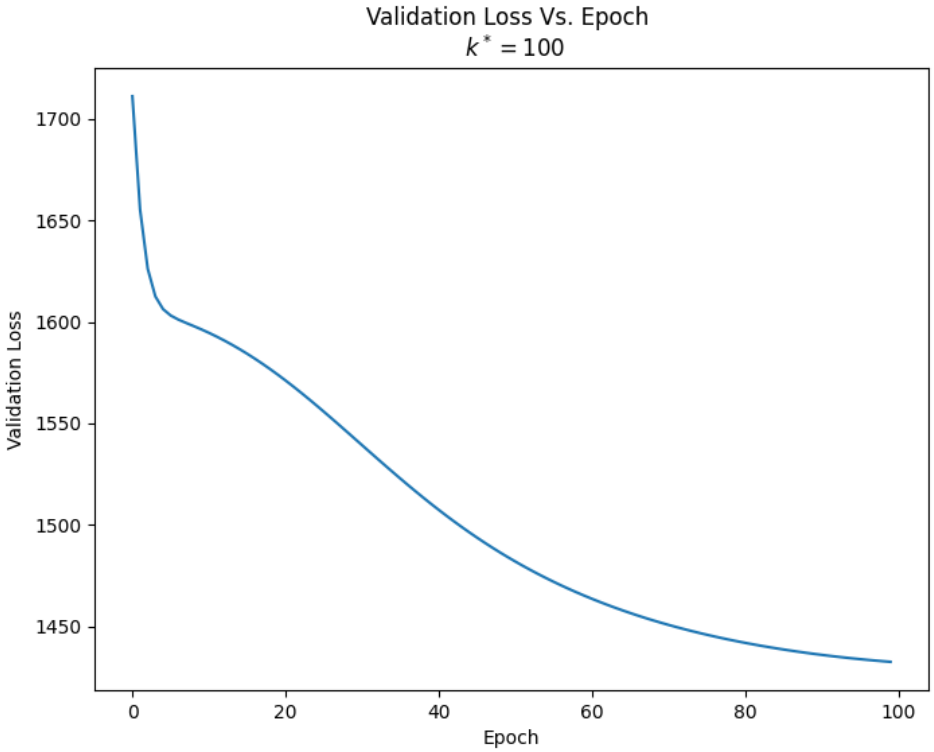
Completed in `neural_network.py`.

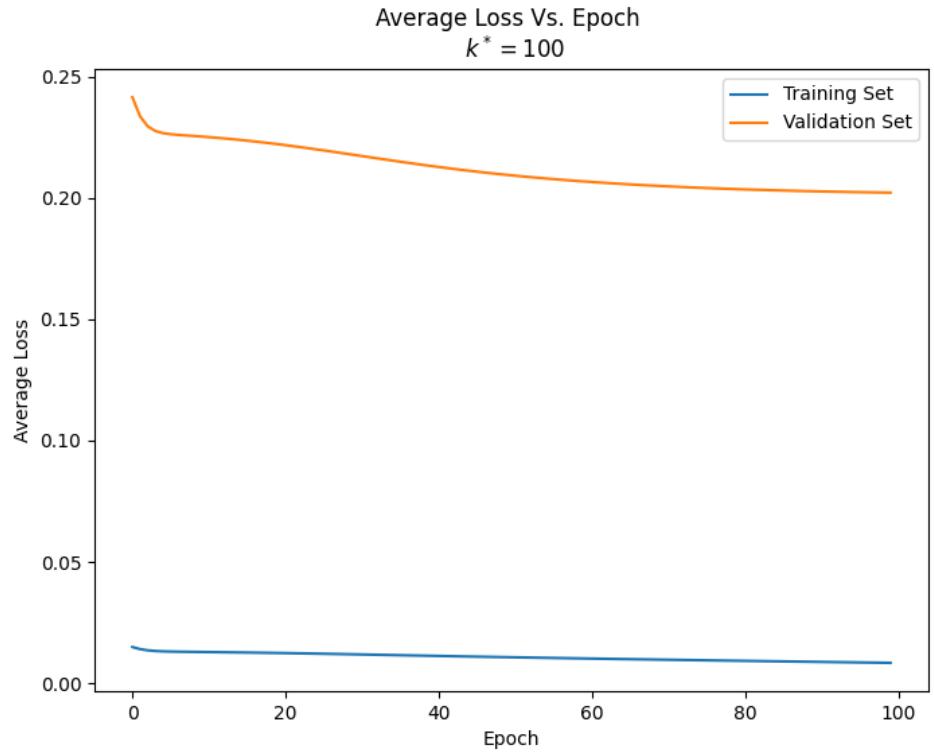
## c)

- $k^* = 100$
- Learning rate = 0.003
- Number of epochs = 100

## d)




Validation Loss Vs. Epoch
$k^* = 100$
Validation Loss
1700
1650
1600
1550
1500
1450
0
20
40
60
80
100
Epoch


Average Loss Vs. Epoch
$k^* = 100$
Training Set
Validation Set
Average Loss
0.25
0.20
0.15
0.10
0.05
0.00
0
20
40
60
80
100
Epoch

As training progresses through epochs, both the training and validation loss decrease.

Final test accuracy: 0.6855771944679651

## e)

- $\lambda^* = 0.001$

| **Validation Accuracy** | **Test Accuracy** |
|---|---|
| 0.687553 | 0.678239 |

Between different values of $\lambda$ there are large differences in validation accuracy. However, the model performs slightly better on validation with the regularization penalty, but does worse on test. However, there is not much difference between the accuracies with and without regularization. Multiple training runs with regularization yield similar validation accuracies.

# 4

The ensemble process we implemented uses three Item Response Theory (IRT) models. The training data $\mathcal{D}$ is first sampled with replacement to form 3 datasets $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$ such that $|\mathcal{D}| = |\mathcal{D}_1| = |\mathcal{D}_2| = |\mathcal{D}_3|$. We use a set seed to ensure results are reproducible. Then, 3 separate IRT models $M_1, M_2, M_3$ are trained on one of $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$ respectively. The hyperparameters used are a learning rate $\alpha = 0.005$ and a total number of epochs of $N_{\text{epoch}} = 100$. To make a prediction, a majority vote is conducted. The outcome of this majority vote is the bagged prediction. We denote $m_i$ to be the prediction generated by the $i$-th IRT model.

$$y_{\text{bagged}} = \mathbb{I}\left(\frac{1}{3}\sum_{i=1}^{3} m_i \geq 0.5\right)$$

| **Validation Accuracy** | **Test Accuracy** |
|---|---|
| 0.7027942421676545 | 0.7081569291560824 |

The final validation accuracy is about 3% lower than the one achieved by using only a single IRT model, but the final test accuracy is higher by about 0.1%.

The training data is already sufficient in training a model which resembles the population distribution. Further generalization using an ensembling method appears to not be very effective for improving the overall accuracy ability for the model to generalize.

# Part B

## 1

The Symmetric Autoencoder (SAE) created in Part A appeared to be underfitting due to two factors: the lower accuracy on both validation and test data in comparison to the Item Response Theory model, and the decrease in accuracy when regularization was included in the objective function. To address the underfitting, we decided to increase the complexity of the model by adding more layers and units to the autoencoder, as well as incorporating student metadata to improve encoding.

The original function for a user $v \in \mathbb{R}^{N_{\text{Questions}}}$ computed by the network in Part A was

$$f_A(\boldsymbol{v};\boldsymbol{\theta}) = h\left(\boldsymbol{W}^{(2)} g\left(\boldsymbol{W}^{(1)}\boldsymbol{v} + \boldsymbol{b}^{(1)}\right) + \boldsymbol{b}^{(2)}\right) \in \mathbb{R}^{N_{\text{questions}}}$$

where $h$ and $g$ are both the sigmoid activation function, $\boldsymbol{W}^{(1)} \in \mathbb{R}^{k \times N_{\text{questions}}}$, and $\boldsymbol{W}^{(2)} \in \mathbb{R}^{N_{\text{questions}} \times k}$.

Our proposed Asymmetric Autoencoder (AAE) network inspired by Sun et. al [3] includes an additional layer making it asymmetric, and incorporates student metadata into the inputs of the . For a user $v \in \mathbb{R}^{N_{\text{Questions}}+2}$ and their gender and premium pupil status, the network computes the function

$$f_B(\boldsymbol{v};\boldsymbol{\theta}) = h_3\left(\boldsymbol{W}^{(3)} h_2\left(\boldsymbol{W}^{(2)} h_1\left(\boldsymbol{W}^{(1)}\boldsymbol{v} + \boldsymbol{b}^{(1)}\right) + \boldsymbol{b}^{(2)}\right) + \boldsymbol{b}^{(3)}\right) \in \mathbb{R}^{N_{\text{questions}}}$$

where $h_1, h_2, h_3$ are all the sigmoid activation function, $\boldsymbol{W}^{(1)} \in \mathbb{R}^{k+2 \times N_{\text{questions}}+2}$, $\boldsymbol{W}^{(2)} \in \mathbb{R}^{k \times k+2}$, and $\boldsymbol{W}^{(2)} \in \mathbb{R}^{N_{\text{questions}} \times k}$.

Notice that the gender of a user, $r$, and premium pupil status, $p$, information is concatenated directly into the input vector:

$$\boldsymbol{v} = \begin{bmatrix} c_1 & c_2 & \dots & c_{N_{\text{Questions}}} & r & p \end{bmatrix}^\top$$

No changes were made to the objective function, and regularization was not included for reasons outlined in section 3 below. The objective function used was:

$$\min_{\theta} \sum_{\boldsymbol{v} \in \mathcal{S}} ||\boldsymbol{v} - f(\boldsymbol{v};\boldsymbol{\theta})||_2^2$$

where $\boldsymbol{v} \in \mathbb{R}^{N_{\text{questions}}}$ from a set of users $\mathcal{S}$.

## 2

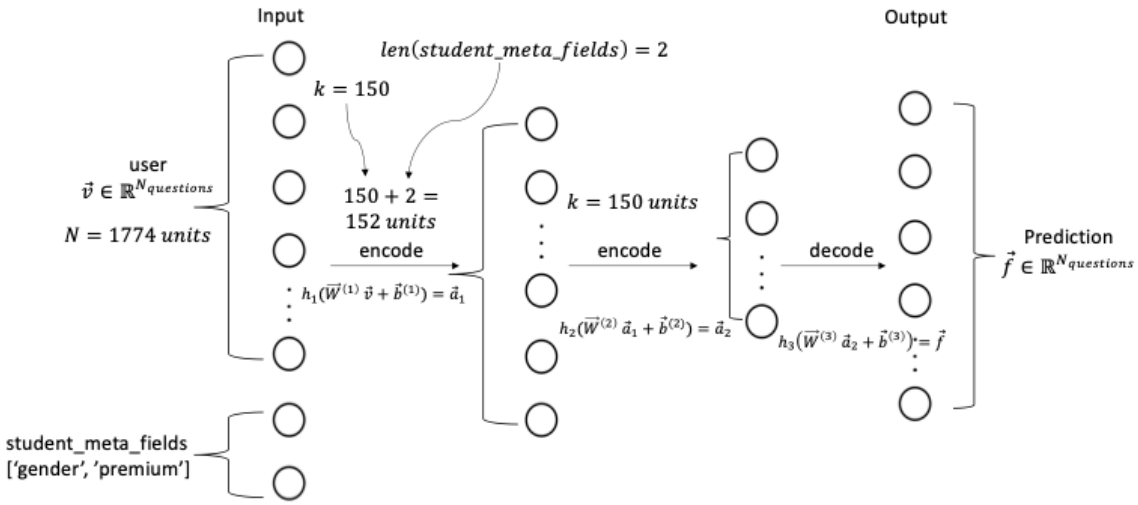



Figure 1: The proposed Asymmetric Autoencoder (AAE) architecture. Note the addition of another hidden layer on the encoder side, and the inclusion of the student metadata fields in the input.

## 3

We decided to extend the autoencoder since we thought it had the most avenues for extensibility and further exploration. When researching other autoencoders for diagnostic question recommendation, we discovered that a common suggestion was to include metadata in the hidden layers [1, 2].

As suggested in Part B Question 1, our AAE model concatenates additional student metadata (being student gender and premium status) to the input vector along with the student ids, aiming for the autoencoder to capture additional insights about the relationship between students and the diagnostic questions.

To test whether these additions would improve our model, we built multiple models with different combinations of the provided student metadata fields of gender, premium status, and age (calculated using date of birth), as well as with different sizes of the additional hidden layer.

Of all the combinations of metadata fields, the one that provided the best results was the combination of gender and premium status. We found that including the age field decreased our accuracy by a few percentage points during our testing.

Of all the additional hidden layer sizes examined (layer 2), the best combination of code vector size and layer 2 size was $k = 150$ and a hidden layer of 152, corresponding to the two metadata fields. With more units in layer 2 ($N_{\text{questions}} + k$), the training time increased significantly, taking about 10 seconds more per epoch, and the validation accuracy increased by 2-3 percentage points.

The benefits of our model are likely due to optimization rather than regularization. Adding more hidden layers and encoding the student metadata optimizes the model by better capturing the relationship between students and diagnostic questions. Furthermore, when we attempted to add $L_2$ regularization to our model ($\lambda = 0.001, 0.01$), the validation accuracy decreased by 2-3 percentage points, so our final model does not include any regularization.

| | Validation Accuracy | Test Accuracy |
|---|---|---|
| **kNN-U** ($k = 11$) | 0.689529 | 0.684166 |
| **kNN-I** ($k = 21$) | 0.692210 | 0.668360 |
| **IRT** | 0.739698 | 0.706746 |
| **E-IRT** ($3 \times$ IRT) | 0.702794 | 0.708157 |
| **SAE** | 0.687553 | 0.678239 |
| **AAE** | 0.697855 | 0.695174 |

Table 1: Validation and test accuracies for the different models from Part A and our proposed model. Models included: k-Nearest Neighbours, User-Based and Item-Based collaborative filtering [kNN-U, kNN-I]; (Ensembled) Item Response Theory [IRT, E-IRT]; Symmetric/Asymmetric Autoencoder [SAE, AAE].

Comparing our autoencoder to the other models implemented in Part A, our proposed model improves on both the validation and test accuracy of the original SAE model by approximately 1% and 2% respectively, indicating there was a successful reduction in underfitting by including another hidden layer and including side information like the student gender and premium status. However, our model was unable to beat the accuracies of both the IRT model and the ensembled IRT model, falling short by 1-3 percentage points.

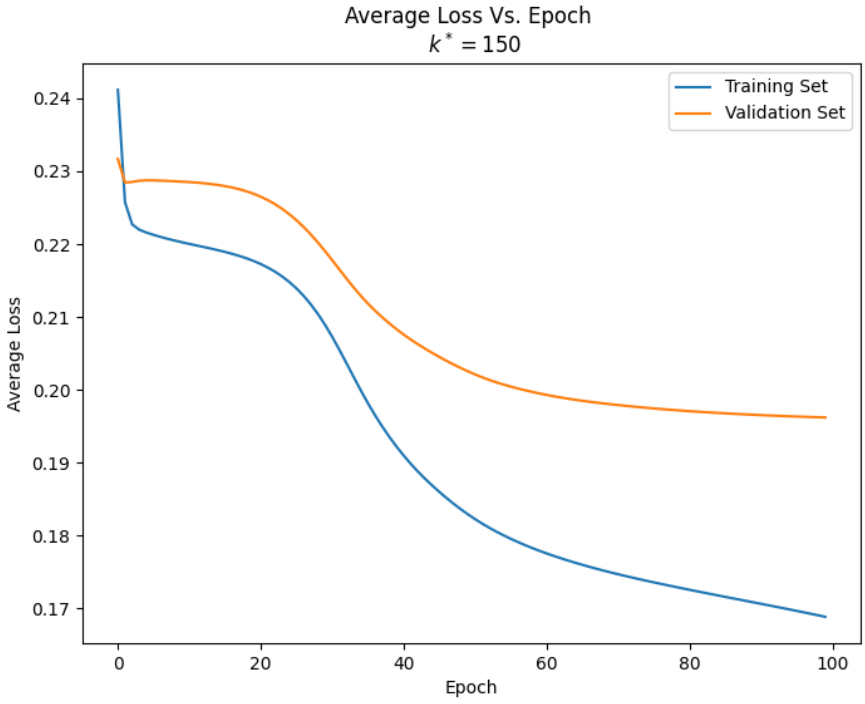



Figure 2: Average loss as a function of epoch during training for the AAE model.

Additionally, based on Figure 2, our model does not appear to be overfitting the data since the training ends at about the same time the validation loss flattens out.

## 4

We expect our model to perform poorly when we incorporate metadata that may not have meaningful correlation to the performance of the student. During the experimentation and tuning parts of developing our AAE, we struggled to see any improvement in the validation accuracy with the inclusion of the 'age' metadata. It took much experimentation to determine which combination of metadata fields could be used for better training and we eventually concluded that not all of the metadata was contributing to improving the accuracy of our model.

We believe this limitation is due to the intrinsic nature of the 'age' feature not having any relevant impact on the performance of the student and therefore our model starts to overfit what it almost interprets as noise. It is also possible that our model is not capable of catching the subtle effects that the 'age' feature has and we need further extension of our neural network to make better usage of the metadata.

Another limitation could be lack of modelling for the relationships between questions using the question metadata. In our model, we easily added a few more units in each layer to include the student features, but the dimension of the question metadata is incomparably larger so we could not find a way to take advantage of this data.

One final limitation may be the way NaN values were handled. We decided to make NaN values take a value of 0, with non-NaN values taking some value strictly greater than 0. This treats the NaN values as useful information for the model, and incorrectly find some erroneous relationship between students (with NaN gender or premium status metadata) and diagnostic questions as a result.

We think it might be possible to address these limitations by somehow combining the question metadata with the student age to form a low-dimensional feature. We suggest this extension because while 'age' may not have a significant correlation with the student performances, we think some groups of ages might be better at dealing with certain types of questions, thus making their combination a useful feature.

## References

[1] Bank, D., Koenigstein, N., and Giryes, R. Autoencoders. *CoRR abs/2003.05991* (2020).

[2] Strub, F., Mary, J., and Gaudel, R. Hybrid recommender system based on autoencoders. *CoRR abs/1606.07659* (2016).

[3] Sun, Y., Mao, H., Guo, Q., and Yi, Z. Learning a good representation with unsymmetrical auto-encoder. *Neural Computing and Applications 27*, 5 (July 2016), 1361–1367.

## Team Contributions

All parts and questions were worked on collaboratively by all group members, for some questions some members focused on certain aspects more. These aspects are outlined below:

- **Andrew C**: A.1 (plots), A.2d, A.4 (resample), B.2, B.4
- **Andrew X**: A.1 (item-based filtering), A.3e, B (metadata experimentation), B.3
- **Raymond L**: Overall write-up, A.1 (main), A.4 (main), B.1, B.3