CSC 311 Winter 2024 Project

Eric, Vishwanath, Shaffaan, Siyang

April 4, 2024

Contents

1 kNN

(a) k* = 11 with test accuracy of 0.6841659610499576 with the plot figure 1.

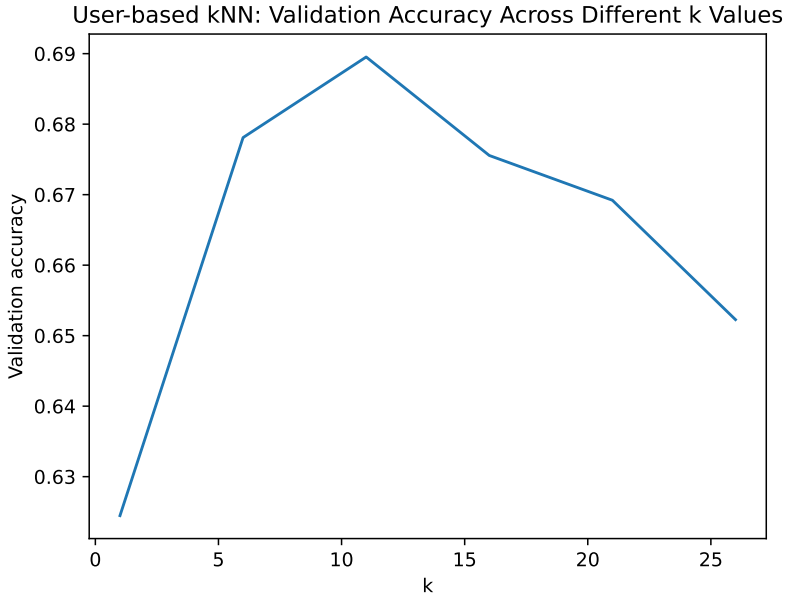

Figure 1: User-based kNN plot

(b) The underlying assumption on which item-based collaborative filtering relies on the likelihood of a new student answering question A correctly is predicted by observing the consistency in answers to question A and another question B across multiple students. If the same students tend to answer both questions correctly or incorrectly, then one can forecast the outcome for question A based on the established response pattern to question B.

(c) k* = 21 with test accuracy of 0.6816257408975445 with the plot figure 2.

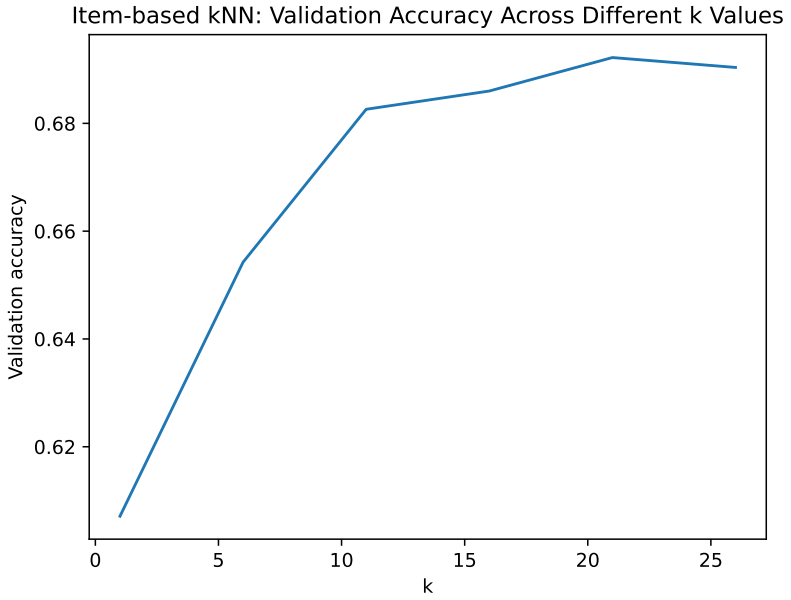

Figure 2: Item-based kNN plot

(d) User-based collaborative filtering marginally outperformed item-based collaborative filtering based on the the test accuracy.

(e)

Potential Limitations:

1. Scalability:
   - As the dataset expands with more users or items or choose to run it on a large dataset, kNN can become computationally intensive. This is due to the necessity of calculating distances across all pairs of users or items, which can lead to inefficiencies for particularly extensive datasets.
2. Sparsity:
   - Another issue could be the lack of sufficient data. Given that approximately 94% (as calculated using np.isnan(sparse_matrix).sum() / sparse_matrix.size) of the values are missing in the sparse matrix used to calculate distances between items or users, this scarcity could potentially diminish the effectiveness of the NaN-Euclidean distance metric deployed in the kNN algorithm.

2 Item Response Theory

(a) Because the students can either answer one diagnostic question only correctly or incorrectly, for student i and question j, we can represent c_{ij} as Bernoulli random variable.

Because $p(c_{ij} = 1|\theta_i, \beta_j) = \frac{exp(\theta_i - \beta j)}{1+exp(\theta_i - \beta_j)}$ and c_{ij} is Bernoulli random variable,

$$\begin{aligned} p(c_{ij}|\theta_i, \beta_j) &= (\frac{exp(\theta_i - \beta j)}{1 + exp(\theta_i - \beta_j)})^{c_{ij}}(1 - \frac{exp(\theta_i - \beta j)}{1 + exp(\theta_i - \beta_j)})^{(1-c_{ij})} \\ &= (\frac{exp(\theta_i - \beta j)}{1 + exp(\theta_i - \beta_j)})^{c_{ij}}(\frac{1}{1 + exp(\theta_i - \beta_j)})^{(1-c_{ij})} \end{aligned}$$

Because the probability of different students answering different questions are independent of each other, the likelihood of the sparse matrix given θ and β parameters are:

$$p(C|\theta, \beta) = \prod_{i=0}^{541} \prod_{j=0}^{1773} p(c_{ij})$$

Therefore, the log-likelihood is:

$$\begin{aligned} &\sum_{i=0}^{541} \sum_{j=0}^{1773} log(p(c_{ij})) \\ &= \sum_{i=0}^{541} \sum_{j=0}^{1773} log((\frac{exp(\theta_i - \beta j)}{1 + exp(\theta_i - \beta_j)})^{c_{ij}}(\frac{1}{1 + exp(\theta_i - \beta_j)})^{(1-c_{ij})})) \\ &= \sum_{i=0}^{541} \sum_{j=0}^{1773} (c_{ij}((\theta_i - \beta_j) - log(1 + exp(\theta_i - \beta_j))) + (1 - c_{ij})(log(1) - log(1 + exp(\theta_i - \beta_j)))) \\ &= \sum_{i=0}^{541} \sum_{j=0}^{1773} (c_{ij}(\theta_i - \beta_j) - c_{ij}(log(1 + exp(\theta_i - \beta_j))) + (1 - c_{ij})(-log(1 + exp(\theta_i - \beta_j)))) \\ &= \sum_{i=0}^{541} \sum_{j=0}^{1773} (c_{ij}(\theta_i - \beta_j) + (c_{ij} + 1 - c_{ij})(-log(1 + exp(\theta_i - \beta_j)))) \\ &= \sum_{i=0}^{541} \sum_{j=0}^{1773} (c_{ij}(\theta_i - \beta_j) - log(1 + exp(\theta_i - \beta_j))) \end{aligned}$$

Therefore, the log-likelihood is: $log(p(C|\theta, \beta) = \sum_{i=0}^{541} \sum_{j=0}^{1773} (c_{ij}(\theta_i - \beta_j) - log(1 + exp(\theta_i - \beta_j)))$

Let LL denote that log-likelihood (i.e. $LL = log(p(C|\theta, \beta) = \sum_{i=0}^{541} \sum_{j=0}^{1773} (c_{ij}(\theta_i - \beta_j) - log(1 + exp(\theta_i - \beta_j))))$

The derivative of the log-likelihood LL with respect to θ_i is

$$\begin{aligned}
\frac{\partial LL}{\partial \theta_i} &= \frac{\partial}{\partial \theta_i}(\sum_{k=0}^{541}\sum_{j=0}^{1773}(c_{kj}(\theta_k - \beta_j) - log(1 + exp(\theta_k - \beta_j)))) \\
&= \sum_{j=0}^{1773}(\frac{\partial}{\partial \theta_i}c_{ij}(\theta_i - \beta_j) - \frac{\partial}{\partial \theta_i}log(1 + exp(\theta_i - \beta_j))) \\
&= \sum_{j=0}^{1773}(c_{ij} - \frac{\frac{\partial}{\partial \theta_i}(1 + exp(\theta_i - \beta_j))}{1 + exp(\theta_i - \beta_j)}) \\
&= \sum_{j=0}^{1773}(c_{ij} - \frac{\frac{\partial}{\partial \theta_i}(exp(\theta_i - \beta_j))}{1 + exp(\theta_i - \beta_j)}) \\
&= \sum_{j=0}^{1773}(c_{ij} - \frac{exp(\theta_i - \beta_j)}{1 + exp(\theta_i - \beta_j)}) \\
&= \sum_{j=0}^{1773}(c_{ij} - p(c_{ij} = 1|\theta_i, \beta_j))
\end{aligned}$$

The derivative of the log-likelihood LL with respect to β_j is

$$\begin{aligned}
\frac{\partial LL}{\partial \beta_j} &= \frac{\partial}{\partial \beta_j}(\sum_{i=0}^{541}\sum_{k=0}^{1773}(c_{ik}(\theta_i - \beta_k) - log(1 + exp(\theta_i - \beta_k)))) \\
&= \sum_{i=0}^{541}(\frac{\partial}{\partial \beta_j}c_{ij}(\theta_i - \beta_j) - \frac{\partial}{\partial \beta_j}log(1 + exp(\theta_i - \beta_j))) \\
&= \sum_{i=0}^{541}(-c_{ij} - \frac{\frac{\partial}{\partial \beta_j}(1 + exp(\theta_i - \beta_j))}{1 + exp(\theta_i - \beta_j)}) \\
&= \sum_{i=0}^{541}(-c_{ij} - \frac{\frac{\partial}{\partial \beta_j}(exp(\theta_i - \beta_j))}{1 + exp(\theta_i - \beta_j)}) \\
&= \sum_{i=0}^{541}(-c_{ij} - \frac{-exp(\theta_i - \beta_j)}{1 + exp(\theta_i - \beta_j)}) \\
&= \sum_{i=0}^{541}(-c_{ij} + \frac{exp(\theta_i - \beta_j)}{1 + exp(\theta_i - \beta_j)}) \\
&= \sum_{i=0}^{541}(-c_{ij} + p(c_{ij} = 1|\theta_i, \beta_j))
\end{aligned}$$

(b) See python file

(c) The hyper-parameters we selected as tuning them is:

learning rate: 0.00271

number of iterations: 50

The validation accuracy of our final model is: 0.708580299 (70.86%)

The testing accuracy of our final model is: 0.702511995 (70.25%)

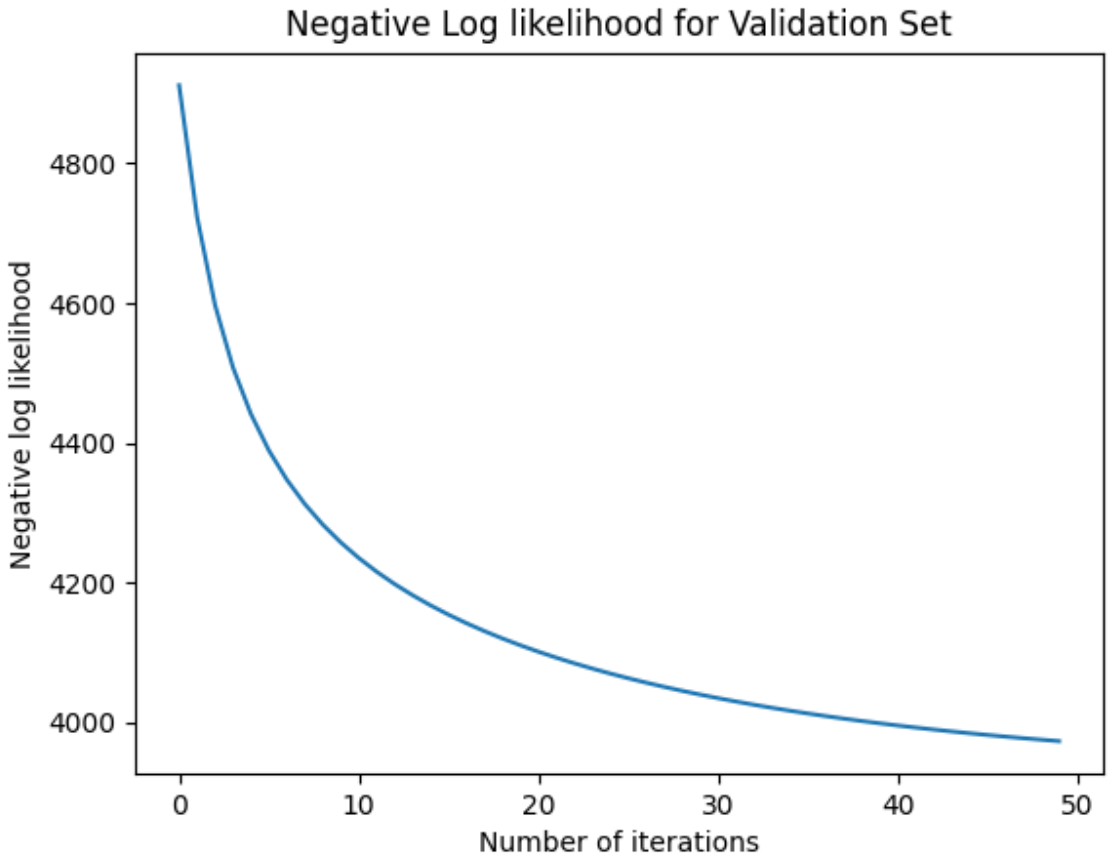

Figure 3: Negative Log-likelihood of validation set

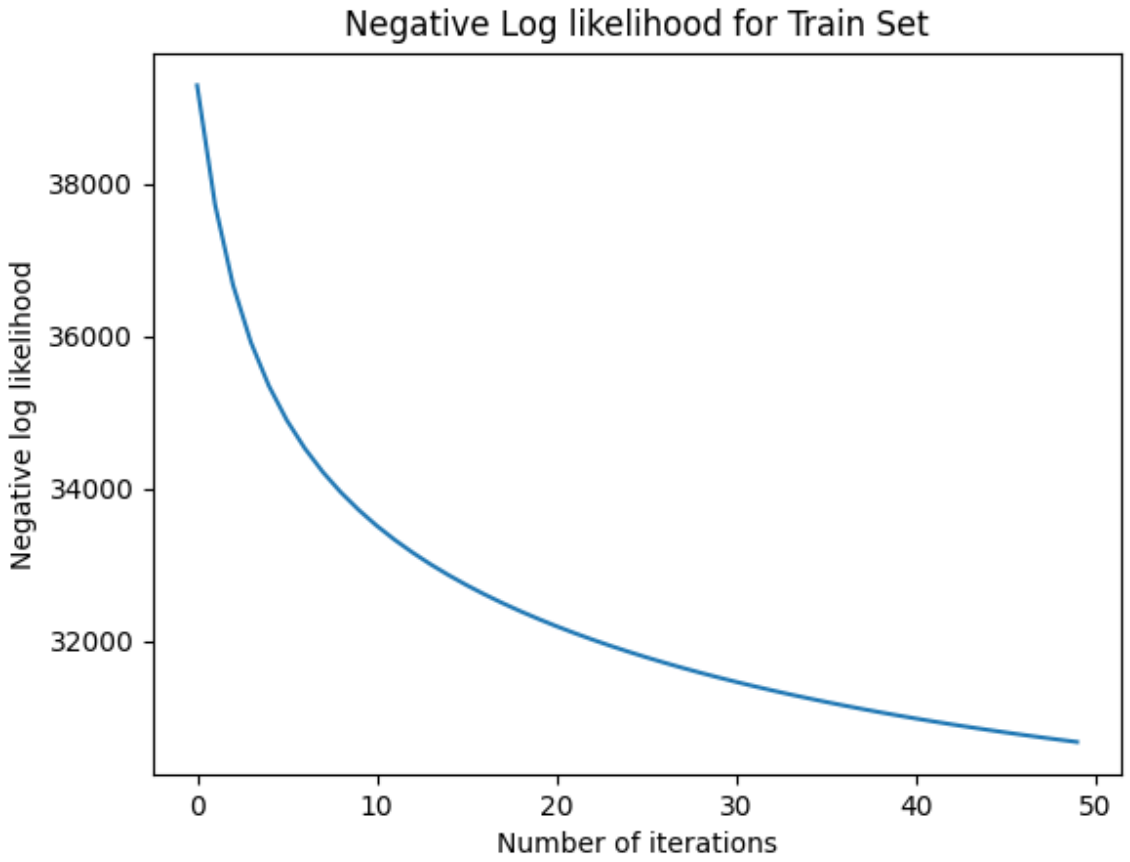

Figure 4: Negative Log-likelihood of training set

(d)

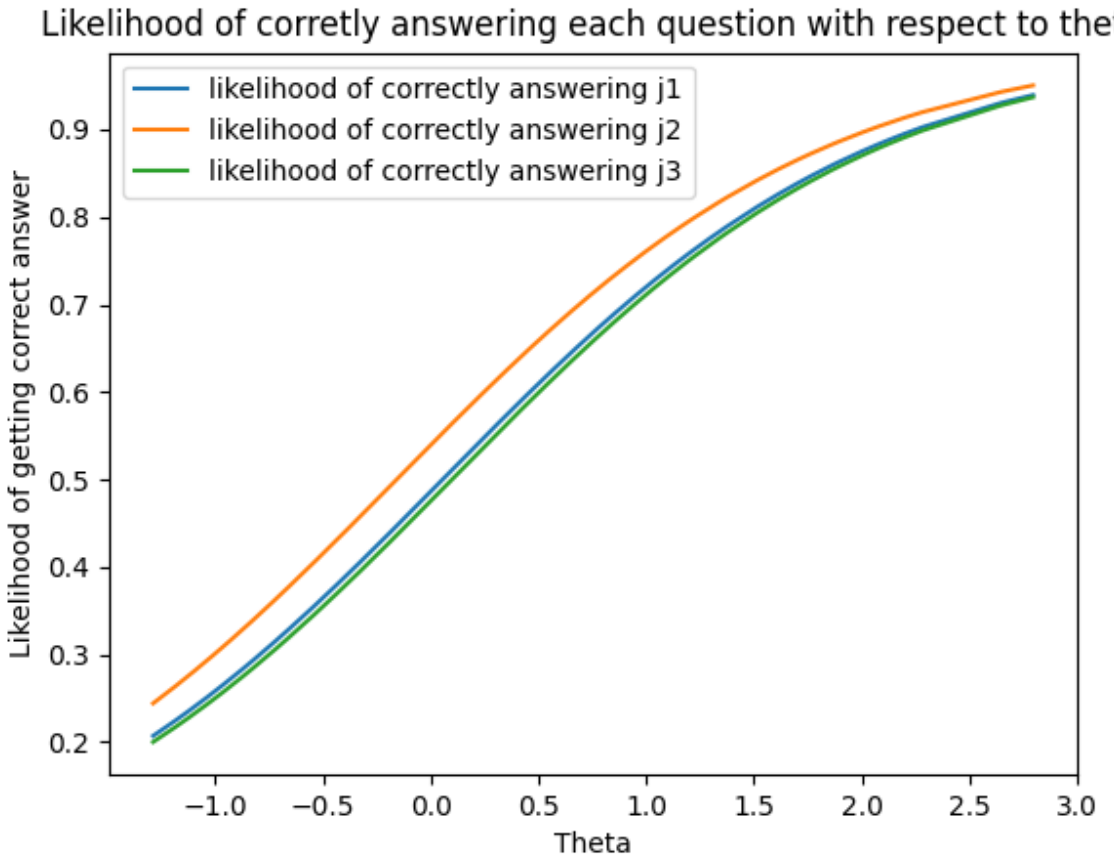

Figure 5: Log-likelihood of each question with respect to Theta

With increasing theta value, the curves generally all go upward in S shape (with relatively lower slope on the 2 ends and higher slope in the middle). These curves represent how the students' abilities affect their chances of answering each question correctly (In general, as students' abilities improves, they are doing better on every question, despite how hard each question is, as the curves are all going upwards). The curves are reflecting the difficulty level of each question as well because the ones that are steeper in average would have relatively lower difficulty because students can largely improve their chances of correctly answering them with slight increase of their abilities, while those with lower slope values will requires higher improvement on students' abilities to increase students' chances of correctly answering them.

3 Neural Networks

(a) The forward pass is completed in the 'forward' method of the AutoEncoder class in the python file 'neural_network.py'

(b) The unregularized AutoEncoder was trained with the hyperparameters k (hidden unit dimension), lr (learning rate), and num_epoch (number of epochs when performing SGD) iterated over the following ranges

$$k \in \{10, 50, 100, 200, 500\}$$
$$\text{lr} \in \{0.01, 0.05, 0.005\}$$
$$\text{num\_epoch} \in \{10, 20, 40, 60\}$$

The combination $(k, \text{lr}, \text{num\_epoch}) = (50, 0.05, 10)$ was found to be most performant.

(c) The optimal hyperparameters from part (b) yielded the following training curves with a final validation and test accuracy of 69.28% and 68.28% respectively.

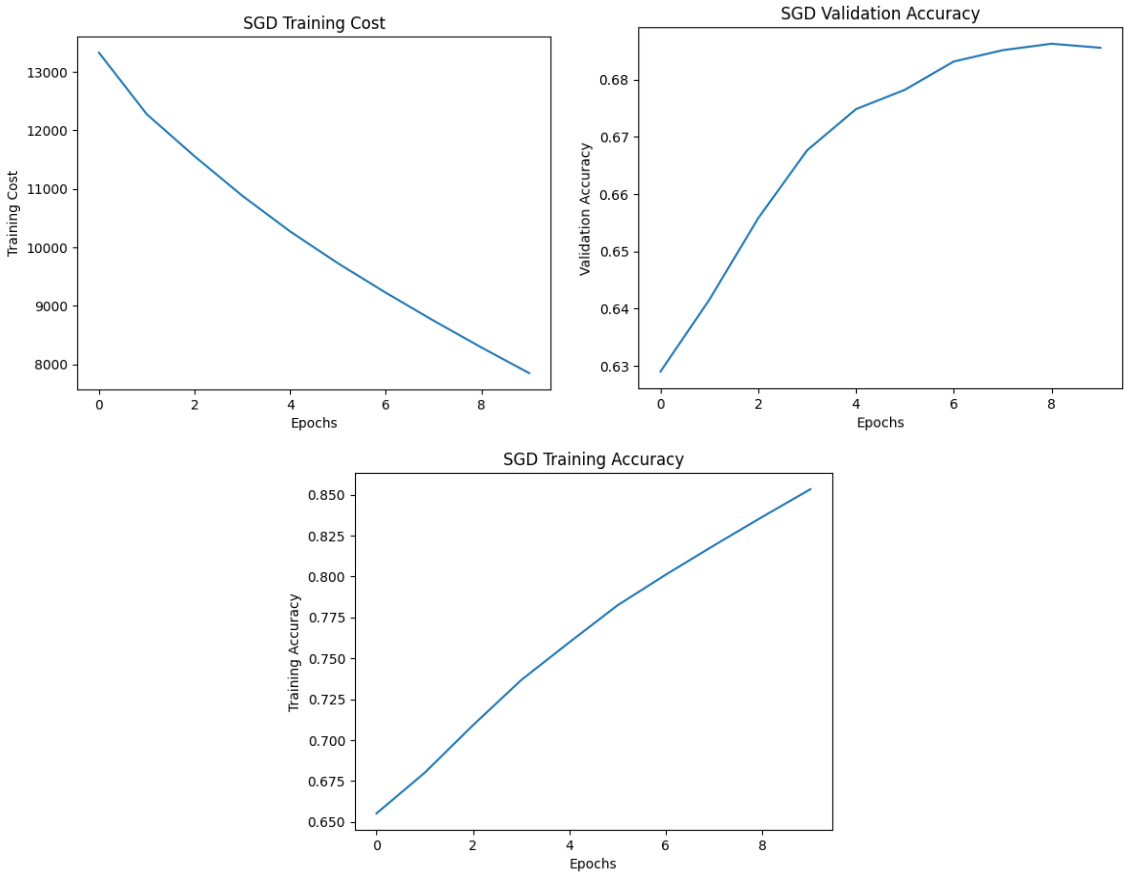

Figure 6: Training curves for unregularized AutoEncoder with hidden unit dimension $k = 50$, learning rate lr $= 0.05$, and number of epochs $= 10$. Training Cost, Validation Accuracy and Training Accuracy plotted against epoch and displayed in top left, top right and bottom respectively.

(d) The L_2 regularizer is implemented in the loss calculation of the train function as seen in the 'neural_network.py' file. When tuning the regularizer weight λ, a value of 0.001 provided the highest validation accuracy, as seen in Figure 7. The final choice of hyperparameters $(k, \text{lr}, \text{num\_epoch}, \lambda) = (50, 0.05, 10, 0.001)$ achieved validation and test accuracies of 68.66% and 68.08% respectively. We note that this is slightly lower than the test accuracy of the unregularized AutoEncoder.

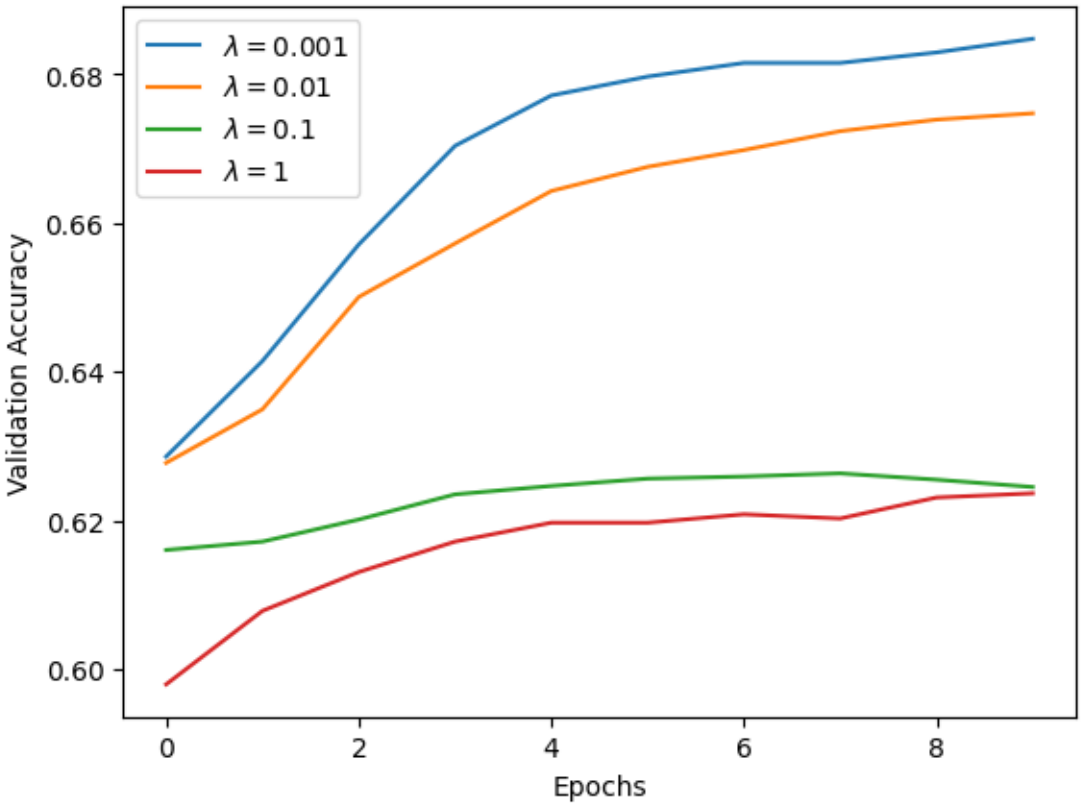

Figure 7: Validation Accuracy Training Curves for L_2 regularized AutoEncoder with regularizer weight $\lambda = 0.001, 0.01, 0.1$ and 1. Hidden unit dimension $k = 50$, learning rate lr = 0.05, and number of epochs = 10

4 Part B - Improving the AutoEncoder

4.1 Limitations of Original AutoEncoder

The AutoEncoder implemented in part (a) appears to underfit the data, as evidenced by the high training cost and relatively low training accuracy in Figure 6. Further, when tuning the regularizer, it was found that the validation accuracy decreased as the regularizer weight, λ, increased as exhibited in Figure 7. The regularizer achieved optimal performance with $\lambda = 0.001$, yielding a still lower validation accuracy than the unregularized AutoEncoder (see section 3). Regularizers seek to penalize finely tuned weights which are associated with overly complex models. As the regularizer diminshed validation accuracy, we can conclude that the AutoEncoder is not complex enough to model the data.

4.2 Modified Algorithm

To remedy the underfitting issue, we introduced additional complexity in the model by adding two more hidden layers. The capabilities of a neural network with just one layer are somewhat restricted in terms of capturing complex patterns. However, by adding an additional hidden layer, our model gains a more robust set of parameters. This enhancement enables it to identify more complex relationships within the data. This aspect is especially beneficial when the connection between input and output is non-linear. Furthermore, we conducted exhaustive testing with different activation functions like Tanh and ReLU because it is known that different activation functions can alter the models performance [Bro21]. We knew that Sigmoid should be used for the output layer so that it squishes the output between 0 and 1 and after a lot of testing we found that the simplicity of the ReLU combined with the Sigmoid afterwards led to the best validation accuracy's. The forward pass for the model is presented in the equations below.

$$encode = ReLU(W^{(1)} * x + b^{(1)}) \tag{1}$$

$$codevector = Sigmoid(W^{(2)} * encode + b^{(2)}) \tag{2}$$

$$decode = Sigmoid(W^{(3)} * codevector + b^{(3)}) \tag{3}$$

We further introduced a refinement to our sparse matrix structure by incorporating additional information within the columns representing Gender and Premium Pupil. We aimed to integrate this enhancement to better understand student performance. Students who come from wealthier backgrounds and are able to attend better schools may perform better. Similarly we added gender as there might be underlying patterns with certain topics and gender may help the model predict better. The augmented matrix is shown in Figure 9.

Because we added lots of complexity to this new model, we also put different regularizers to this model to modify its complexity to avoid potential overfitting. We implemented 4

different popular regularizers with learning rate of 0.001: (1) Not using regularizer (2) L2 Regularizer (3) L1 Regularizer (4) Dropout Regularizer (with 0.25 dropout rate) [Anw21]. We tried each of them and found that the model performs relatively the same for each of them with enough epoches (150 epoches) as shown in figure 10, with their validation accuracy presented in table 1. Unregularized model performs relatively best among them, and among the regularized models, Dropout regularizer is the best regularizer. Therefore, we would choose unregularized model.

With these modifications in place, we trained the AutoEncoder to arrive at the hyperparameters presented in Table 2.

Table 1: Validation accuracy of different regularizers on the new model

No_Regularizer	L2_Regularizer	L1_Regularizer	Dropout_Regularizer
70.66	70.45	70.41	70.62

Table 2: Hyperparameters for enhanced AutoEncoder

lr	encoder_dim	codevector_dim	num_epoch	λ	Regularizer
0.001	50	10	150	0	None

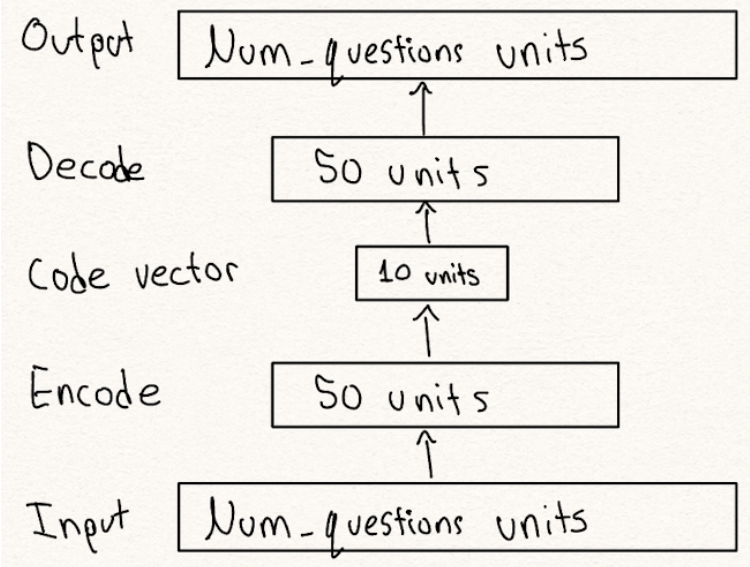

Figure 8: Diagram of modified AutoEncoder Neural Network with 2 more hidden layers

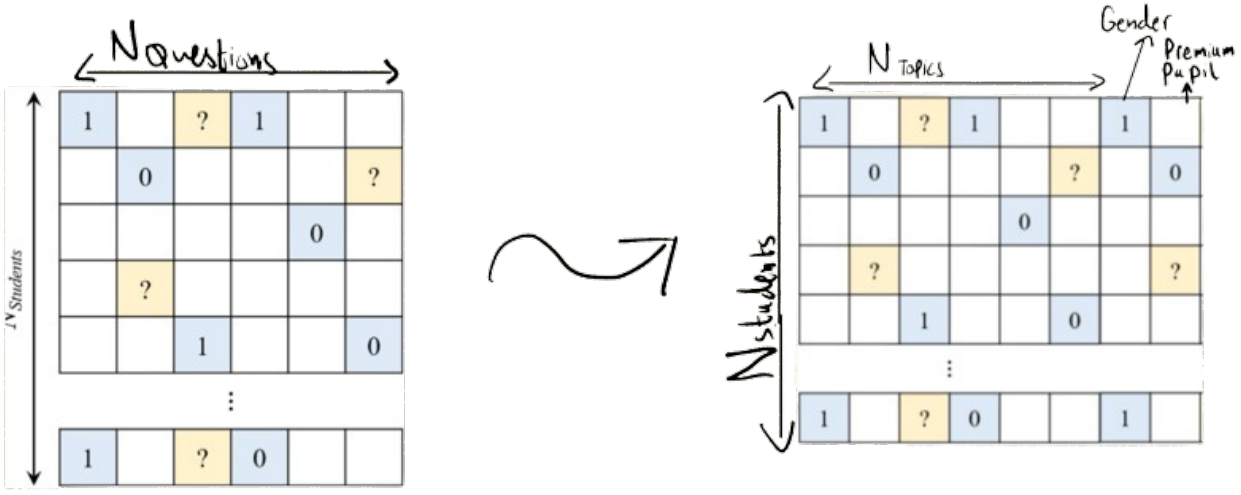

Figure 9: Augmented Sparse Matrix with Gender and Premium Pupil

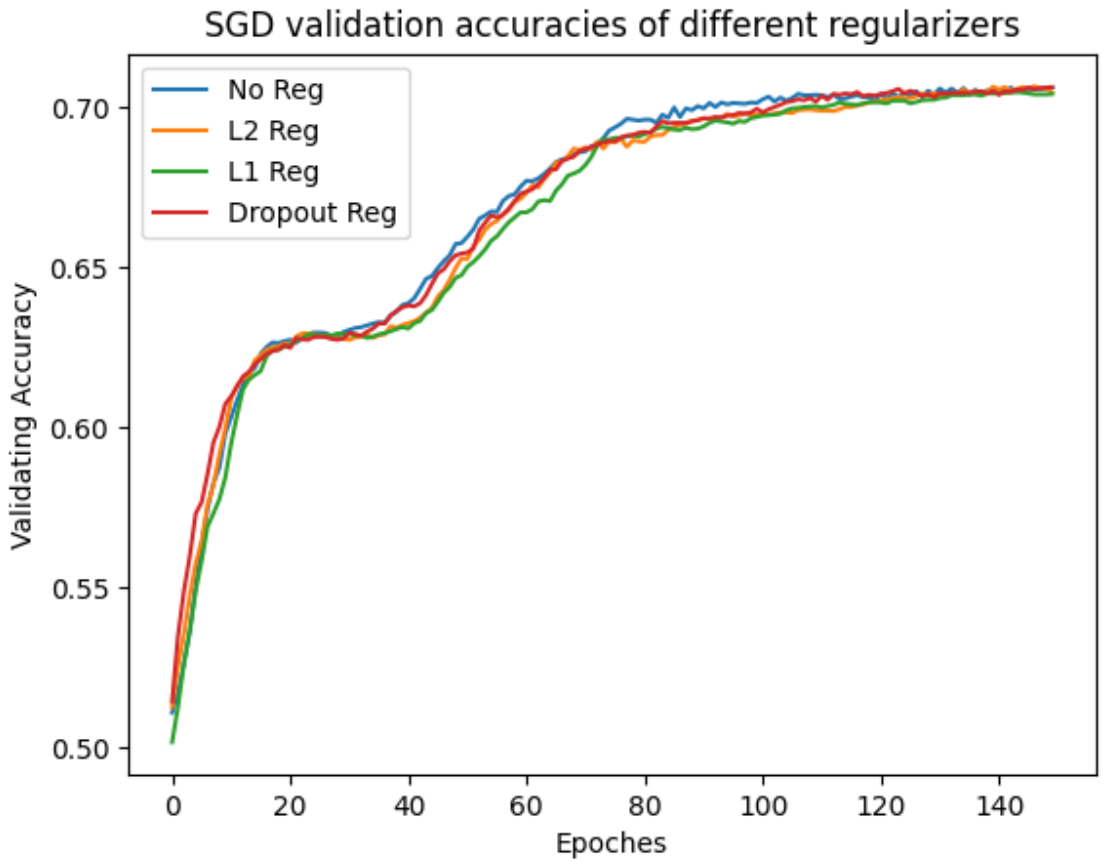

Figure 10: The validation accuracy of different regularizers on the modified model

4.3 Failed Attempts

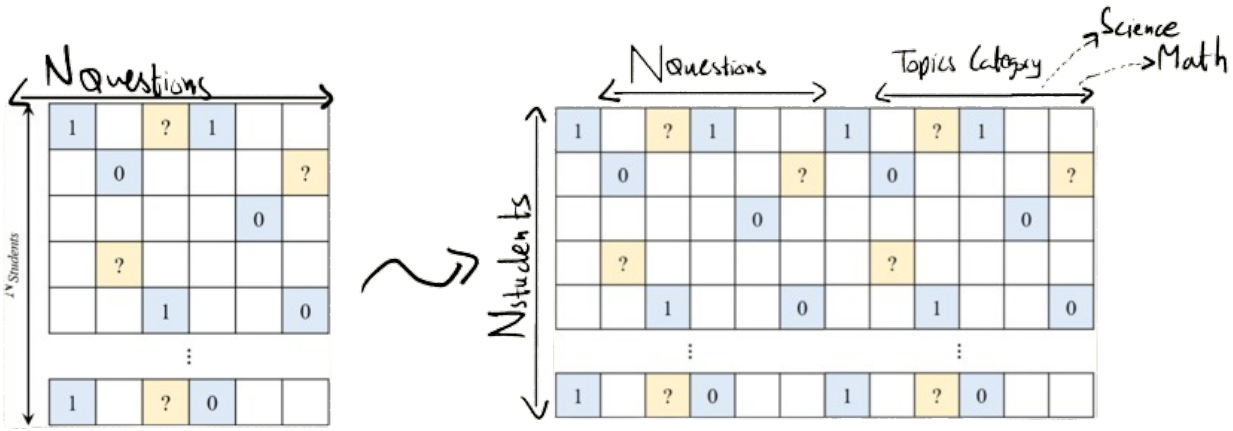

Figure 11: Sparse Matrix with Topic Categories added

We introduced a refinement to our sparse matrix structure by incorporating additional information within the columns representing each subject topic as shown in Figure 11. Specifically, within these columns, each cell now corresponds to the number of questions a student answered correctly that included the respective topic. We aimed to integrate this enhancement to foster a deeper understanding of student performance, anticipating that when a student excels in a topic, they are likely to excel again, hopefully enhancing the accuracy of our model.

However, when implemented, validation accuracy decreased to around 0.55. The data preprocessing process involves traversing the entire matrix for each student, which incurs significant computational overhead. Despite efforts to accelerate the process using tools like NumPy and matrix operations, the computational expense remains substantial. Additionally, as accuracy decreases with this approach, we've chosen to streamline our modified algorithm by omitting it altogether.

4.4 Baseline Comparison

Model	Validation Accuracy (%)	Test Accuracy (%)
kNN	68.95	68.42
IRT	70.86	70.25
Part (a) Unregularized AutoEncoder	69.28	68.28
Part (a) L_2 Regularized AutoEncoder	68.66	68.08
Part (b) Dropout Regularized AutoEncoder	70.62	70.39
Part (b) Unregularized AutoEncoder	**70.66**	**70.45**

It is evident that our model performs better than the baseline neural network in both validation and testing, however, it falls short of the IRT model in validation. The validation accuracy training curve for this final model is given in Figure 12.

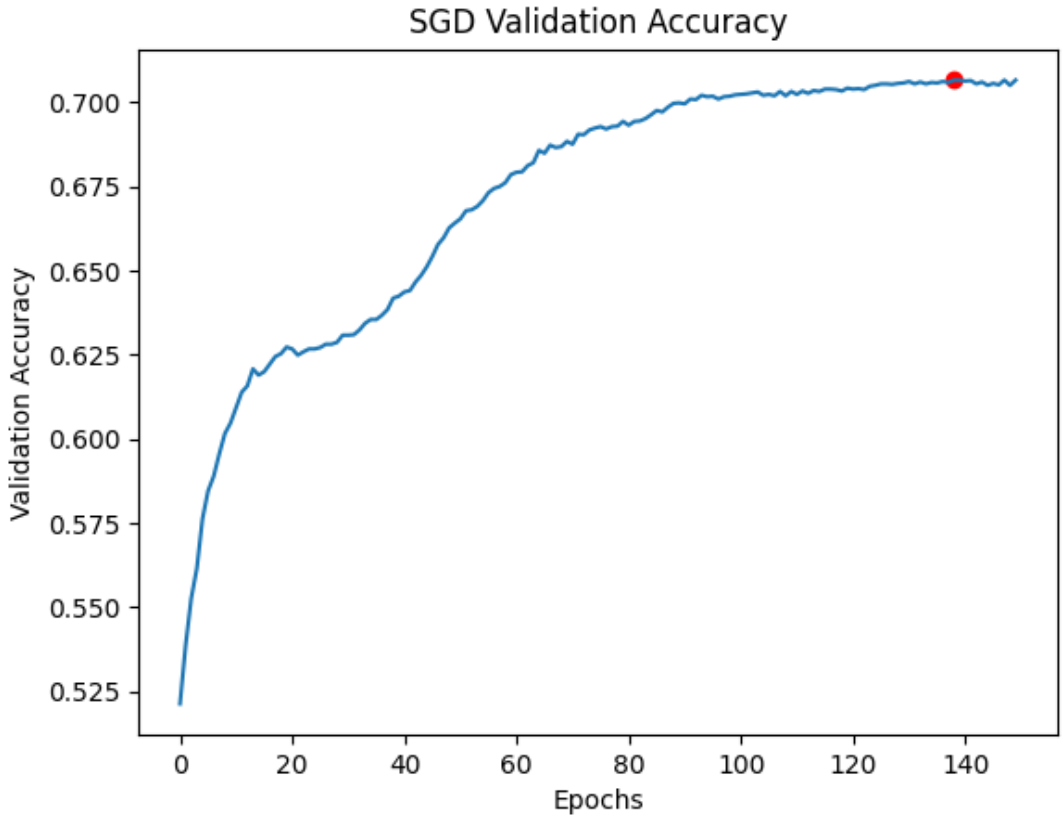

Figure 12: Validation Accuracy of our Unregularized Model

4.5 Model Performance

Our final model is the Part (b) Unregularized AutoEncoder. The increased accuracy of the AutoEncoder is likely linked to the overhaul of the network structure with the addition of 2 more hidden layers and various activation functions. The added complexity sought to compensate for the underfitting seen in the original model. Further, the inclusion of gender and premium pupil student data in the sparse matrix may have allowed the network to develop connections between previously unseen features.

As explained in section 4.2, the regularization of the AutoEncoder did not significantly improve the accuracy of the model. That is likely because the limited number of data still has the issue of underfitting due to the limited number of data points. The number of data is generally suggested to be 10 times more than the number of features [Stu24]. In our model

we have $1774 + 2 = 1776$ features, so we would require 17760 data points for the model to perform well, while the number of data point is only 542. Therefore, even though we have increased complexity of the model, our model is still underfitting because we can not increase the number of data points. Therefore, the regularizers had no obvious effects on our model because our model is still underfitting rather than overfitting. One improvement we can do to address that is to improve the size of data to avoid underfitting.

Hence, the benefits of the enhanced AutoEncoder are attributed to optimization rather than regularization.

4.6 Limitations

Education constantly changes as new information emerges. Schools regularly update their courses to keep up with these changes. For example, what students learned in science a decade ago might be outdated now due to new discoveries. Also, unexpected events like the COVID-19 pandemic can affect how well students learn. For instance, a math problem that was easy to solve before the pandemic might now be challenging because students missed important classroom time. Thus we get to the limitation: Our model doesn't consider these changes and doesn't adjust when new information becomes available or when outside events impact learning; meaning that we keep all their scores as one constant. This means it might not accurately predict the correctness of students' answers to unseen diagnostic questions [Bar+22]. Considering these factors, the AutoEnconder should accommodate changes in curricula, account for external influences on learning, and recognize the fluid nature of knowledge acquisition. Retraining the AutoEncoder periodically becomes essential to ensure its efficacy in predicting the students' answers to unseen questions. We can further fix this by putting dates on the data collected so that the data is only used for a certain time period - for example up to 3 years. This way, the model stays relevant and reliable for assessing student performance.

References

[Anw21] Aqeel Anwar. *Types of regularization in machine learning*. Apr. 2021. URL: https://towardsdatascience.com/types-of-regularization-in-machine-learning-eb5ce5f9bf50.

[Bro21] Jason Brownlee. *How to choose an activation function for deep learning*. Jan. 2021. URL: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/.

[Bar+22] Georgia Barbayannis et al. "Academic stress and mental well-being in college students: Correlations, affected groups, and covid-19". In: *Frontiers in Psychology* 13 (May 2022). DOI: 10.3389/fpsyg.2022.886344.

[Stu24] Thompson Stupak. *How much data is required to train ML models in 2024?* Jan. 2024. URL: https://www.akkio.com/post/how-much-data-is-required-to-train-ml#:~:text=The%20amount%20of%20training%20data,or%20more%20examples%20than%20features..

5 Contributions

Part A
(1) Eric
(2) Siyang and Shaffaan
(3) Vishwanath

Part B: All of us.