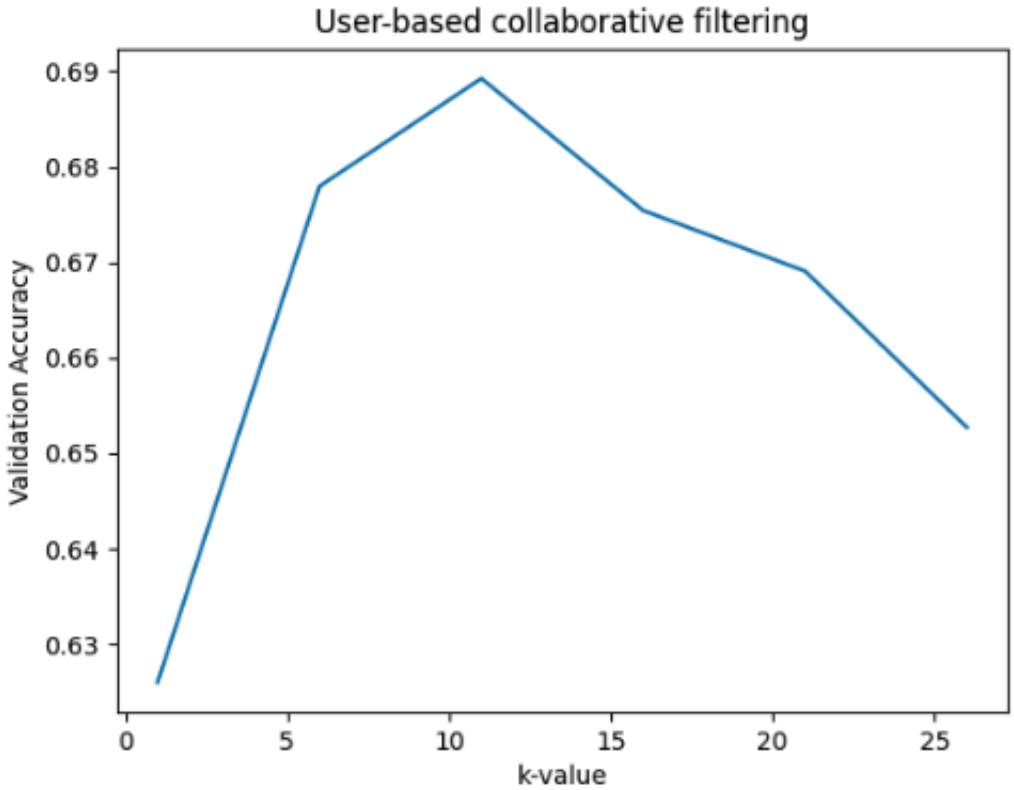
Final Project Writeup

Aabha Roy, Avi Walia, Sukhjeet Nar

November 30, 2024

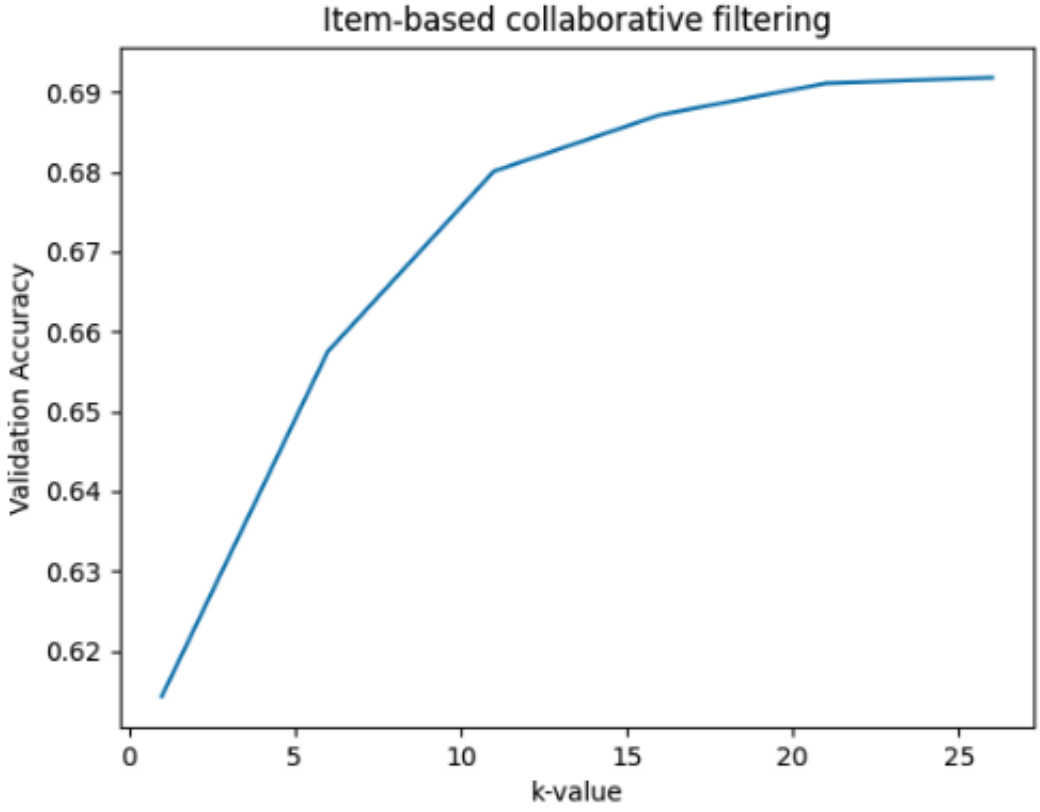
Part A

1. (a) User-based collaborative filtering:
    - Impute By User: k-value: 1, Validation Accuracy: 0.6260231442280553
    - Impute By User: k-value: 6, Validation Accuracy: 0.6779565340107254
    - Impute By User: k-value: 11, Validation Accuracy: 0.6892464013547841
    - Impute By User: k-value: 16, Validation Accuracy: 0.6754163138583121
    - Impute By User: k-value: 21, Validation Accuracy: 0.6690657634772792
    - Impute By User: k-value: 26, Validation Accuracy: 0.652695455828394



(b) The k* that has the highest performance on the validation data is k* = 11
Test Accuracy (user-based, k=11): 0.6821902342647473

(c) The underlying assumption on item-based collaborative filtering is that given two questions A and B, if they are similar (i.e. they are answered correctly/incorrectly by similar groups of students), then the correctness of a student's answer can be predicted based on their answers to the similar questions. This method relies on the idea that questions with similar patterns of responses have comparable levels of difficulty or shared concepts.



Item-based collaborative filtering:

- Impute By Item: k-value: 1, Validation Accuracy: 0.6143099068585944
- Impute By Item: k-value: 6, Validation Accuracy: 0.657493649449619
- Impute By Item: k-value: 11, Validation Accuracy: 0.6800733841377364
- Impute By Item: k-value: 16, Validation Accuracy: 0.6871295512277731
- Impute By Item: k-value: 21, Validation Accuracy: 0.6910810047981937
- Impute By Item: k-value: 26, Validation Accuracy: 0.6917866215071973

The k* that has the highest performance on the validation data is k* = 26
Test Accuracy (item-based, k=26): 0.6830369743155518

(d) Comparison:
User-based: Best k=11, Test Accuracy = 0.6821902342647473
Item-based: Best k=26, Test Accuracy = 0.6830369743155518
Item-based collaborative filtering performs better.

(e)
- KNN is computationally expensive for large datasets at test time because it involves finding the nearest neighbours for every prediction.
- This method relies heavily on the quality of data points. If the dataset is sparse, meaning many entries in the user-question matrix are missing (NaN), it could lead to reduced prediction

accuracies. This is because KNN relies on sufficient overlap between users or items to compute similarity, and finding reliable neighbours can become challenging with sparse datasets.

2. (a)

$$p(c_{ij}|\theta_i, \beta_j) = \left(\frac{exp(\theta_i, \beta_j)}{1 + exp(\theta_i, \beta_j)}\right)^{c_{ij}} + \left(1 - \frac{exp(\theta_i, \beta_j)}{1 + exp(\theta_i, \beta_j)}\right)^{(1-c_{ij})}$$
$$= \left(\frac{exp(\theta_i, \beta_j)}{1 + exp(\theta_i, \beta_j)}\right)^{c_{ij}} + \left(\frac{1}{1 + exp(\theta_i, \beta_j)}\right)^{(1-c_{ij})}$$

The log-likelihood $log\ p(\mathbf{C}|\boldsymbol{\theta}, \boldsymbol{\beta})$ can be derived by:

$$log\ p(\mathbf{C}|\boldsymbol{\theta}, \boldsymbol{\beta}) = log \sum_{i,j} \left[\left(\frac{exp(\theta_i, \beta_j)}{1 + exp(\theta_i, \beta_j)}\right)^{c_{ij}} + \left(\frac{1}{1 + exp(\theta_i, \beta_j)}\right)^{(1-c_{ij})} \right]$$
$$= \sum_{i,j} \left[c_{ij}\ log \left(\frac{exp(\theta_i, \beta_j)}{1 + exp(\theta_i, \beta_j)}\right) + (1 - c_{ij})\ log \left(\frac{1}{1 + exp(\theta_i, \beta_j)}\right) \right]$$
$$= \sum_{i,j} \left[c_{ij} \Big(\theta_i - \beta_j - log(1 + exp(\theta_i, \beta_j))\Big) + (1 - c_{ij}) \Big(- log(1 + exp(\theta_i - \beta_j))\Big) \right]$$
$$= \sum_{i,j} \left[c_{ij}(\theta - \beta) - c_{ij}\ log(1 + exp(\theta_i - \beta_j)) - log(1 + exp(\theta_i - \beta_j)) + c_{ij}\ log(1 + exp(\theta_i - \beta_j)) \right]$$
$$= \sum_{i,j} \left[c_{ij}(\theta - \beta) - log(1 + exp(\theta_i - \beta_j)) \right]$$

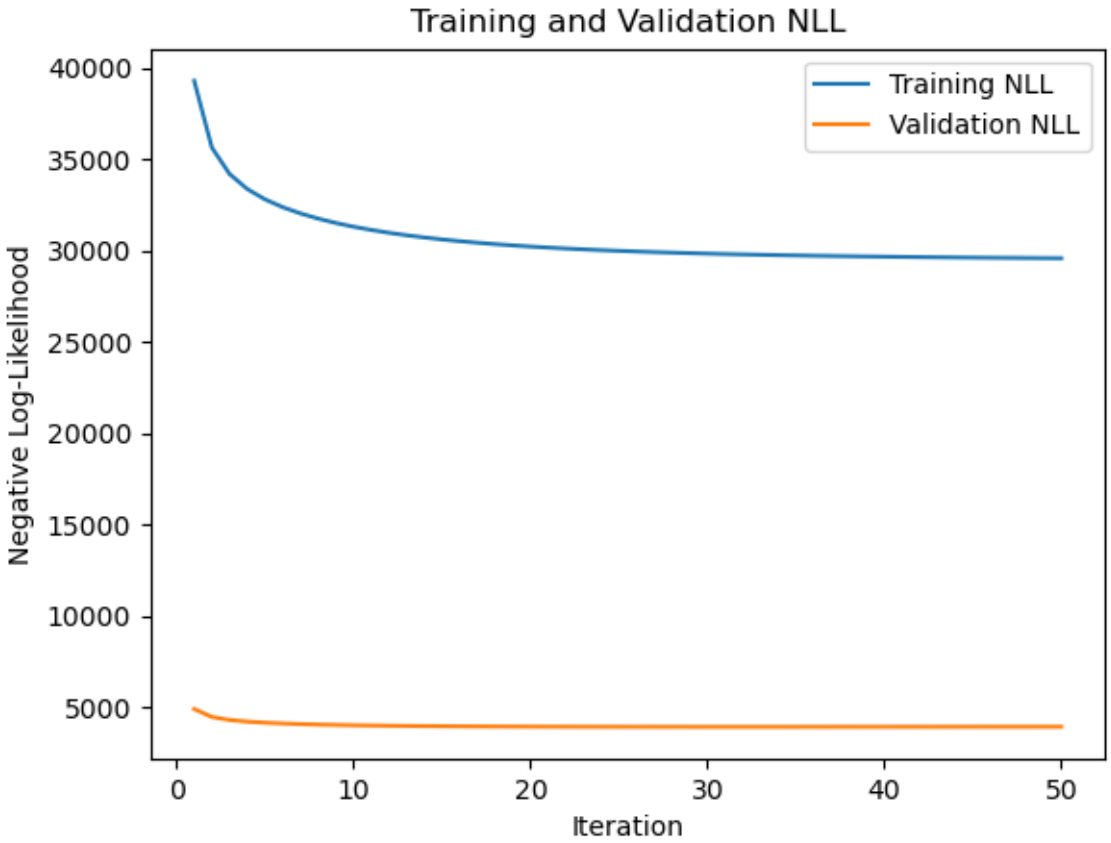
Let p_{ij} be equal to the sigmoid function on $(\theta_i - \beta_j)$.
Then $p_{ij} = \frac{exp(\theta_i, \beta_j)}{1 + exp(\theta_i, \beta_j)}$.
If we take the derivative of $log\ p(\mathbf{C}|\boldsymbol{\theta}, \boldsymbol{\beta})$ with respect to θ_i:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \sum_j (c_{ij} - p_{ij})$$

If we take the derivative of $log\ p(\mathbf{C}|\boldsymbol{\theta}, \boldsymbol{\beta})$ with respect to β_j:

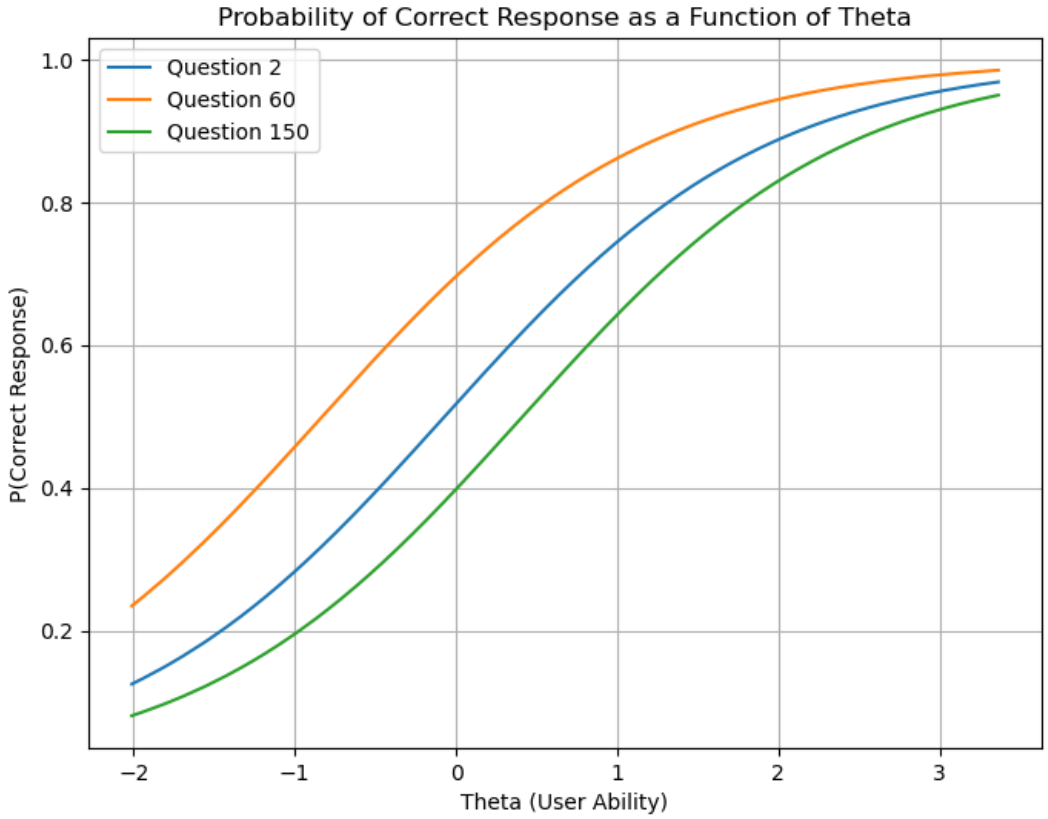
$$\frac{\partial \mathcal{L}}{\partial \beta_j} = \sum_i (p_{ij} - c_{ij})$$

(b) The hyper parameters that I chose were a learning rate of 0.01 and 50 iterations.



(c) The final validation accuracy is 0.7055 and the final test accuracy is 0.7051.

(d) We choose questions 2, 60, and 150.



The graph displays the Item Characteristic Curves from Item Response Theory (IRT), showing the probability of a correct response for each question as a function of user ability (θ). The curves follow a sigmoid shape, which is typical for logistic models, indicating that as user ability increases, the probability of answering correctly also rises. The steepness of the curve reflects the sensitivity of the question to changes in ability. Differences in the curves highlight the relative difficulty of the questions. For instance, the curve for Question 60 (orange) is shifted to the left, suggesting it is easier, as users with lower ability levels (θ) have a higher probability of success. All curves asymptote to 0 for very low abilities and 1 for very high abilities, illustrating that users with extremely low ability are unlikely to answer correctly, while users with high ability are almost certain to do so.

3. **Option 2: Neural Networks**

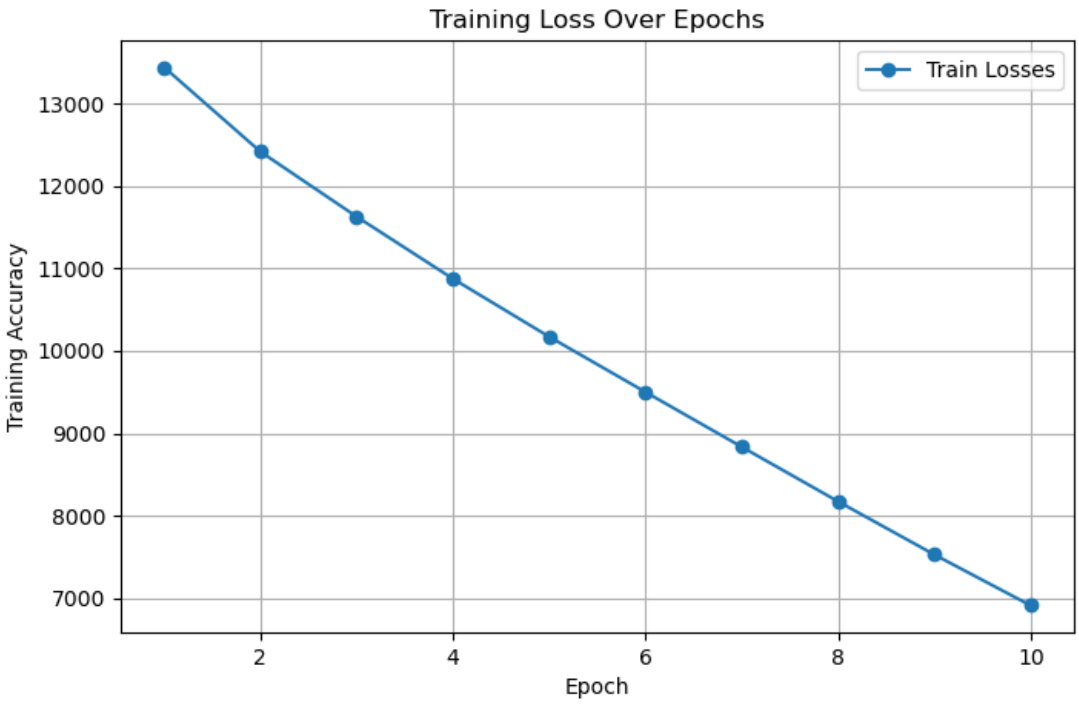
(a)
- **Optimization Approach:**
  - ALS optimizes by alternating between updating different sets of variables and solving a linear least-squares problem for each.
  - Neural networks use iterative gradient descent. The parameters are updated simultaneously based on the gradient of the chosen loss function.
- **Model Architecture:**
  - ALS focuses on a linear model for factorizing a matrix into two low-rank matrices. It assumes a simple relationship between the factors.
  - Neural networks use a layered architecture that allows for non-linear transformations. There are often multiple hidden layers with activation functions.
- **Regularization:**

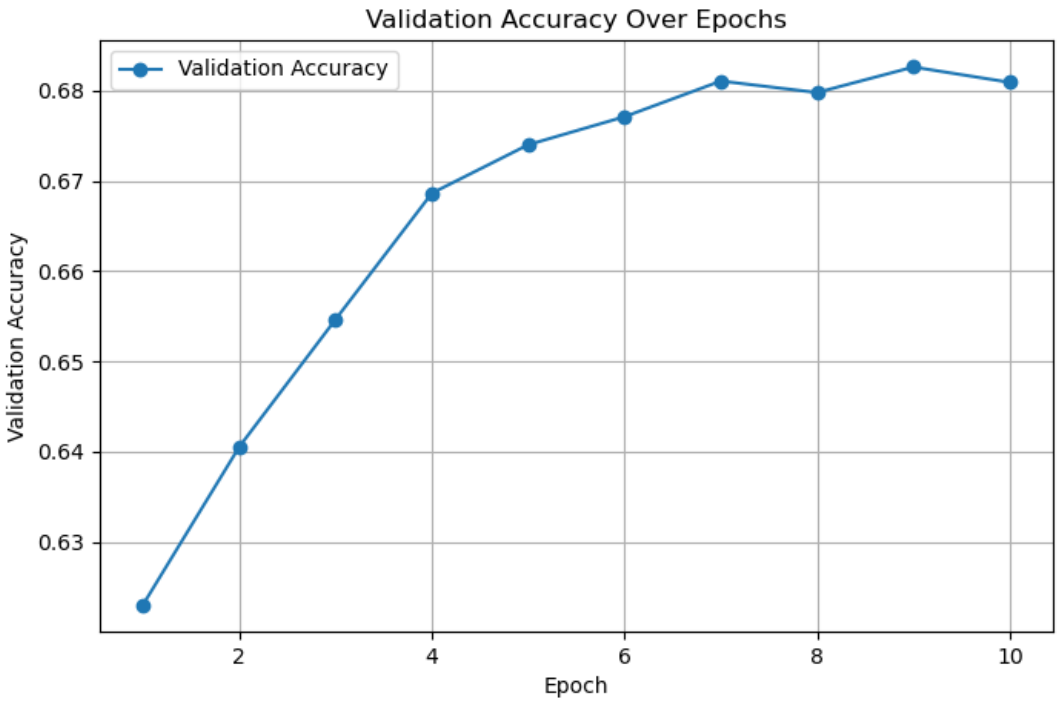
- ALS often includes explicit regularization terms (e.g. L2 regularization) in the least-squares formulation to prevent over-fitting.
- Neural networks use a wide range of regularization techniques such as weight decay and batch normalization.

(b) Code in python file

(c) The hyper-parameters that I found had the highest validation accuracy was {k=100, learning_rate=0.05, epoch=10}.

(d) The final test accuracy is 0.6805.





(e) The lambda we chose was $\lambda = 0.001$. The final validation accuracy is 0.6828 and the final test accuracy is 0.6819. Our model performs marginally better with the regularization penalty.

4. **Ensemble**.

Chosen Base Models: Chosen Base Models: KNN, Item Response Theory (IRT), and Neural Networks (NN). Each model was implemented by using code that was written for Part A.

Final Accuracies:

- Final Validation Accuracy $\rightarrow$ 0.707169
- Final Test Accuracy $\rightarrow$ 0.708439

Implementation Notes: Ensemble was implemented in three core steps: (1) the primary data given for this assignment was randomly sampled (with replacement) to generate three bootstrapped data samples, (2) each model was trained using the three new datasets, (3) the predictions generated by each model were first averaged (mean of the three prediction lists by the same model) and then, by weighted aggregation, the final accuracy was computed.

The combination of these three steps resulted in a better performance using ensemble.

Why it worked: In lecture, we learned that unlike the theoretical space of P_{sample}, when we sample with replacement for each bootstrapped dataset, each produced sample is not independent of each other. For bagging to truly have its benefits, artificial independence must be introduced. This is what motivated weighted aggregation. Each model's validation accuracy was tested independently. In this case, IRT had the greatest accuracy, then KNN, and followed by NN. Therefore, IRT was given a weight of 0.5, KNN was given 0.3, and NN given 0.2 (weights add to 1). Even though there was a marginal difference between the highest weighted model's accuracy and the final ensemble accuracy, ultimately it was this system that allowed the performance to be better than any single model. By using weighted aggregation, it meant the accuracy was not significantly penalized by a poorer performing model.

Part B: Item Response Modification

1 Formal Description

In the standard Item Response Theory (IRT) model, each question's difficulty is represented by a single parameter, β_j, which may not fully capture the nuanced relationships between questions and the underlying subjects they assess. To address this limitation, we introduce a subject-aware IRT model that incorporates metadata about the subjects associated with each question. Specifically, for a question j associated with a set of subjects $\texttt{Subjects}_j$, its effective difficulty is modified as:

$$\beta_j = \beta_{j,base} + \frac{1}{k} \sum_{s \in \texttt{Subjects}_j} \beta_s$$

where $\beta_{j,base}$ is the base difficulty of the question (the same difficulty we used in the previous version of the algorithm), β_s represents the difficulty adjustment for subject s, and k is the number of subjects that question j is associated with. This extension allows the model to dynamically adjust the difficulty of each question based on the difficulty of the associated subjects, providing a more detailed representation of question difficulty. The log-likelihood function is updated accordingly:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\beta_s}) = \sum_{(i,j) \in \text{data}} \left[c_{ij}(\theta_i - \beta_j) - \log(1 + \exp(\theta_i - \beta_j)) \right]$$

where β_j now includes the subject adjustments. Gradients with respect to β_s are computed and updated alongside θ_i and β_j during training. This is the same log-likelihood function as before, just with a modified β_j term.
We can derive the update rule for β_s similarly to how we derived the update rule for β_j.
If we take the derivative of $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\beta_s})$ with respect to β_s:

$$\frac{\partial \mathcal{L}}{\partial \beta_s} = \sum_i (p_{ij} - c_{ij})$$

where p_{ij} is equal to the sigmoid function on $(\theta_i - \beta_j)$ but β_j, as mentioned, includes the adjustments for subjects.
Then, the update function for β_s would be:

$$\beta_s \leftarrow \beta_s + \eta \cdot \sum_{(i,j) \in \text{data}, s \in \text{S}_j} \left[\frac{1}{k} (p_{ij} - c_{ij}) \right]$$

The decision to multiple the $(p_{ij} - c_{ij})$ term by $\frac{1}{k}$ (where, again, k is the number of subjects) was made to balance the contribution of each subject in determining the effective question difficulty. Without this scaling, subjects specific to a question would each receive the full gradient update, potentially leading to an overestimation of their individual contributions, especially for questions linked to multiple subjects. By dividing the gradient by k, we distribute the influence of the question's difficulty equally across all associated subjects. This adjustment introduces a regularization-like effect, ensuring that no single subject is disproportionately adjusted due to questions shared across subjects.

The proposed modification to the Item Response Theory (IRT) model is designed to improve performance by addressing specific limitations of the standard approach.

1. **Capturing Additional Structure in the Data:**

- The standard IRT model assumes that the difficulty of a question β_j is independent of its underlying subject. This simplification ignores potential patterns or relationships between subjects and the difficulty of associated questions. By incorporating subject-level difficulty parameters (β_s), the model can account for these dependencies, allowing it to represent question difficulty more accurately.

2. **Reduce Underfitting**

   - In traditional IRT, a single parameter (β_j) represents the entire difficulty of a question, which might oversimplify the complexity of real-world educational assessments. Some questions may derive their difficulty primarily from the challenging nature of the subjects they cover. By augmenting the model with β_s, we add granularity, helping the model better align with the observed data and reducing underfitting caused by overly simplistic assumptions.

3. **Enhanced Interpretability:**

   - Beyond prediction accuracy, this extension provides additional interpretability. Subject-specific parameters (β_s) can reveal which subjects are inherently more challenging across the dataset. This insight could guide educators (and online learning tools such as Khan Academy) in identifying and addressing subject-level difficulties that students face.

4. **Optimization and Flexibility:**

   - While the addition of β_s introduces more parameters, it does so in a structured way that leverages the relationships in the data rather than adding complexity arbitrarily. This structured approach avoids overfitting, as the extra parameters are tied directly to interpretable aspects of the data (i.e., subject difficulty). Moreover, the gradient-based optimization naturally integrates the subject parameters without introducing instability.

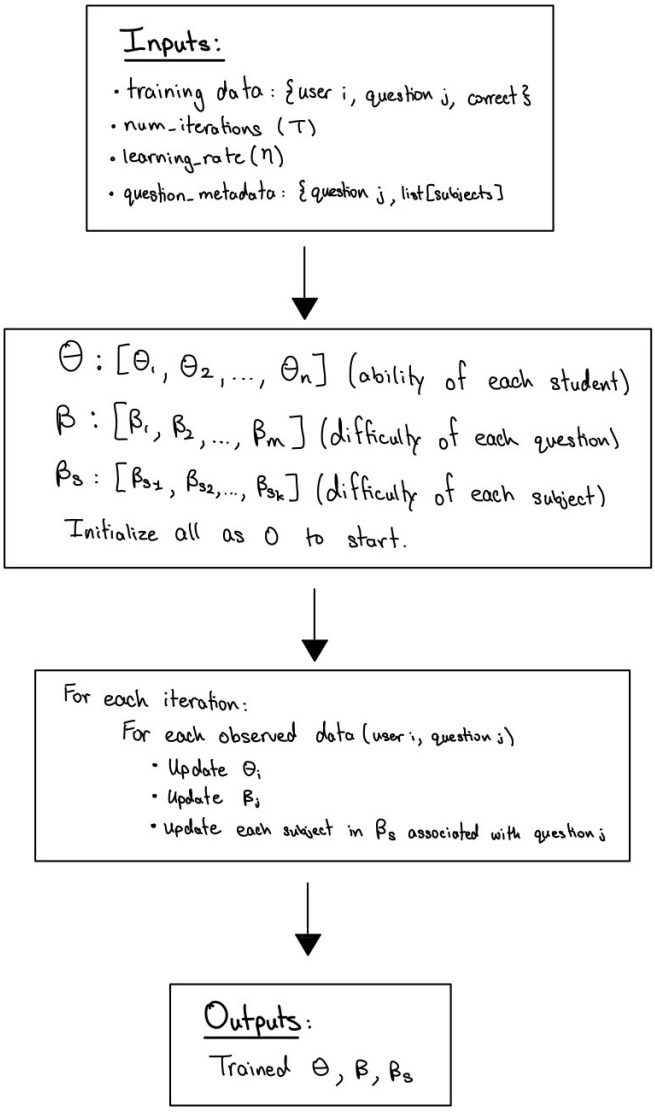
One important feature that we removed from the question metadata was the "Maths" subject. There are several subjects that are members of "Maths" to begin with (e.g. Decimals, BIDMAS, Algebra, etc.). Thus, we removed this subject since it would not help us differentiate each question any better.

The algorithm for the modified IRT was implemented as follows:

Algorithm 1: Subject-Aware Item Response Theory (IRT) Model

Input: Training Data: $\{(u_i, q_j, c_{ij})\}$. u_i is the user, q_j is the question, and c_{ij} is the correctness
Question Metadata: $\{q_j : list[subjects]\}$ (mapping from question q_j to its associated subjects)
Learning Rate: η
Number of Iterations: T
Output: Learned parameters: θ, β, β_s
Initialize parameters $\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\beta_s}$ to 0;
for $t = 1, 2, \ldots, T$ **do**
  foreach *data point* (u_i, q_j, c_{ij}) **do**
    Retrieve the subjects of the question: $S_j = \text{Subjects}_j$;
    Compute the effective question difficulty: $\beta_j^{\text{eff}} = \beta_j + \sum_{s \in S_j} \frac{1}{|S_j|} \beta_s$;
    Compute the probability of correctness: $p_{ij} = \frac{\exp(\theta_i - \beta_j^{\text{eff}})}{1 + \exp(\theta_i - \beta_j^{\text{eff}})}$;
    Compute the error: $\text{Error} = c_{ij} - p_{ij}$;
    Update user ability: $\theta_u \leftarrow \theta_u + \eta \cdot \text{Error}$;
    Update question base difficulty: $\beta_j \leftarrow \beta_j - \eta \cdot \text{Error}$;
    Update subject difficulties: $\beta_s \leftarrow \beta_s - \eta \cdot \text{Error} \cdot \frac{1}{|S_j|}, \quad \forall s \in S_j$;
return θ, β, β_s;

2 Model Diagram



3 Comparison

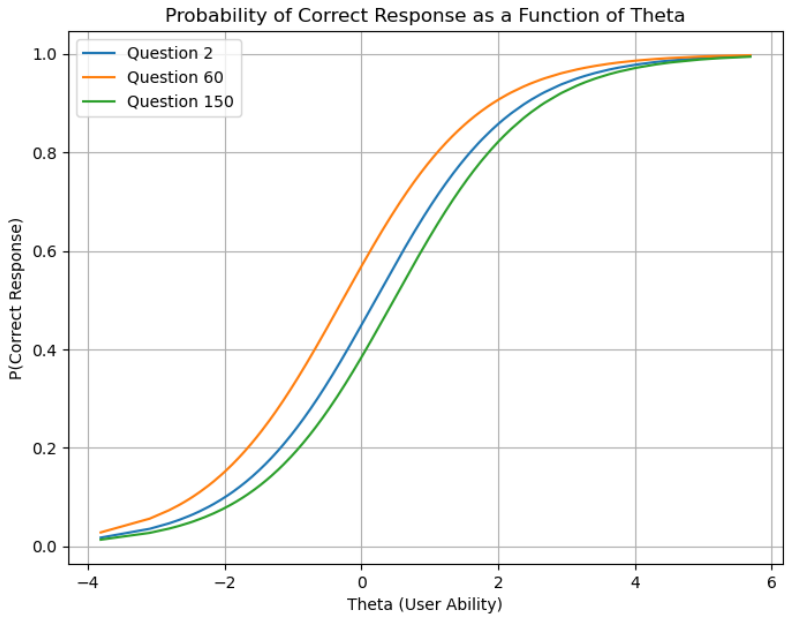
Before Improvement

- Validation Accuracy: 0.7055
- Test Accuracy: 0.7051

After Improvement

- Validation Accuracy: 0.7062
- Test Accuracy: 0.7079

These results show a small but consistent improvement in both validation and test accuracy. This improvement demonstrates that the modified model generalizes better, likely due to its ability to capture subject-specific patterns. The consistency between the validation and test results also indicates that the model is robust and not overfitting to the training data.
We also replotted the ICC curves for the same questions as before (2, 60, and 150):



There is a significant difference in these graphs:
The graph of the newer model has much steeper curves. The steeper slopes in the improved ICCs indicate higher discrimination power with respect to the questions. In the improved model, the probability of a correct response transitions more sharply from low to high as the user ability (θ) increases. This suggests that the improved model can better distinguish between users of varying abilities.

Hypothesis: The improvement in the model's performance (e.g., steeper ICC curves, higher validation and test accuracy) is primarily due to the incorporation of subject-level metadata, which enables a more nuanced representation of question difficulty.
To test this hypothesis and isolate the contribution of the subject-aware adjustment, we can perform the following experiment:
Parameter Sensitivity Analysis: After training the subject-aware IRT model, small random noise was added to each subject parameter (β_s) as follows:

$$\beta_s \leftarrow \beta_s + \epsilon, \epsilon \sim \mathcal{N}(0, 0.5^2)$$

We choose 0.5 as the standard deviation since we wanted a value that would add significant noise to the data without completely changing it.

Evaluation: For the hypothesis to be true, there should be drops in prediction accuracy to indicate that the β_s parameter plays a meaningful role in the improved model performance.

After running this test in python (found in the `hypothesis_eval.py` file). The new test accuracy is 0.7101. This is worse than it was without the subject information added. This indicates that our hypothesis, that the improvement in the model's performance is primarily due to the incorporation of subject-level metadata, is correct.

4 Limitations

The subject-aware IRT model we implemented enhances our standard approach by providing a more nuanced representation of question difficulty. However, it has some limitations in specific scenarios.

- *Outlier questions or users:* This model assumes that the questions are consistent with their pattern of difficulty based on the subjects. However, it is always possible to have questions that deviate from their usual difficulty given their associated subjects. As a consequence of the model's assumptions, questions like this would be poorly modelled. Similarly, it is possible for some students to have inconsistent responses. This could be due to many factors such as resorting to guessing as a consequence of being short on time, or being careless, or tired. Scenarios like this would skew the estimates for θ and β_j, affecting the model's accuracy.
  This limitation exists due to general challenges with educational data that we cannot always accurately account for or predict. Student responses can be influenced by unobserved factors like guessing, fatigue, or motivation, which are not explicitly modelled.
  A proposed solution could be to augment the model to account for behavioural factors like these. We could introduce additional parameters, for example, a slip parameter to model the likelihood of guessing or error for each student. We could also add time-based adjustments to reflect student engagement over the duration of them being active on the platform. This would improve the interpretability and robustness of θ_i and β_j, making the model more reliable in real-world settings.

- *Limitation of the approach:* Incorporating the subject (β_s) into the difficulty calculation is done by finding the average of the parameters. Specifically, if the question covers multiple subjects, the parameter is still averaged. This assumption of independence among subjects is not always accurate in real world settings because there are cases where averaging undermines the true difficulty of the question. For instance, we can agree if a question covers both calculus and linear algebra it, in theory, should be more difficult than any of those subjects alone. Averaging would undercut the difficulty both subjects share. On the contrary, calculating difficulty by using sum of difficulty levels across subjects may equally not be the best approach. This is because if there is a simple question that involves many subjects, and a complex question with very few subjects, our model would produce a higher difficulty scores for even simple questions.
  This limitation exists because our model structure is simple. The linear aggregation of subject difficulty as measured by β_s assumes that the subjects contribute independently and equally to the question difficulty. This oversimplification fails to capture the complex interactions between subjects, especially in multidisciplinary settings.
  Some possible adjustments to improve this metric could be to replace the linear averaging of the subject difficulty (β_s) with weighted or nonlinear techniques. Such as introducing interaction terms to capture how certain combinations of subjects influence question difficulty.

- *Sparse subject metadata:* In cases where the questions are associated with very few subjects, a subject-aware model may not be able to leverage the additional β_s parameter. As aforementioned, this model relies on accurate and granular metadata about the subjects associated with each question. If, for example, there were many questions labeled with the subject, "Math", the model would not be able to precisely gauge the difficulty. Thus poor metadata reduces the interpretability and performance of the model.
  This limitation exists because the model assumes the provided metadata is comprehensive and accurate. It is possible for there to be cases where metadata is incomplete, noisy, or overly generic, limiting the model's ability to leverage subject level adjustments.
  We could account for this better by applying techniques to infer missing metadata or generate more granular subject tags. For example, we could use unsupervised learning to identify latent subject groupings. Ultimately, we could employ other machine learning techniques to find the more hidden relationships among subjects without relying on the predefined labels. We could also refine existing

metadata by analyzing patterns in student responses. As a result, enhanced metadata would allow the subject-aware adjustments to contribute meaningfully even in sparse datasets.

Contributions

Aabha Roy: Implemented and did the write up for KNN.
Avi Walia: Implemented and did the write up for Item Response and Neural Networks.
Sukhjeet Nar: Implemented and did the write up for Ensemble.
We all worked on Part B jointly and have all completed the course evaluation.