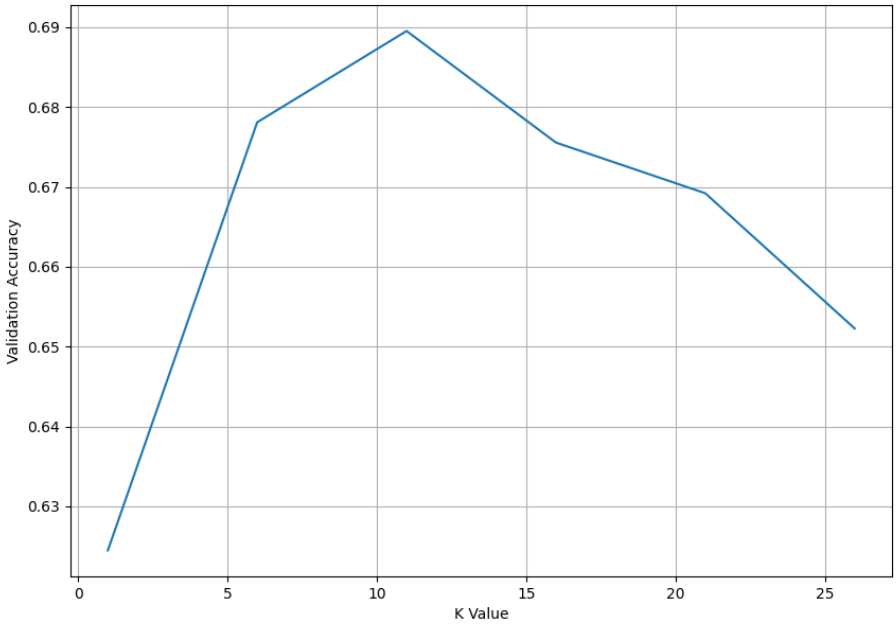
CSC311 Project

Umair Hussain

November 2023

1 k-Nearest Neighbor



(a)

(b) The best k is: 11

With test accuracy of: 0.6841659610499576

(c) The core underlying assumption is that if question A has the same correct and incorrect responses from other students as question B, A's correct response from specific students matches that of question B. Less precisely if two questions seem to be equally difficult the same students who got one of the questions right will get the other question correct

(d) User based filtering with test accuracy of: 0.6841659610499576

Item based filtering with test accuracy of: 0.6816257408975445

They perform basically the same although user based filtering performs slightly better.

(e) If the number of questions is very large our data set will begin to struggle as it will face the curse of dimensionality, that is, the euclidean distances will all be approximately the same. Thus deeply hindering KNN's performance.

Another limitation is since kNN approaches can be sensitive to data sparsity, like in this example. Since not all users have provided an answer to each question. The lack of sufficient overlapping preferences between users can result in poor recommendations or limited accuracy. As the finding the KNN will be less meaningful the less data we have because the distance calculations could be wildly wrong when dealing with NAN.

2 Item Response Theory

(a)

$$L(\theta, \beta|C) = \prod_{i,j} P(c_{ij}|\theta_i, \beta_j)$$

$$log(P(C|\theta, \beta)) = \sum_{i,j} log(P(c_{ij}|\theta_i, \beta_j))$$

$$= \sum_{i,j} log((\frac{e^{\theta_i - \beta_j}}{1 + e^{\theta_i - \beta_j}})^{c_{ij}}(1 - \frac{e^{\theta_i - \beta_j}}{1 + e^{\theta_i - \beta_j}})^{1 - c_{ij}})$$

$$= \sum_{i,j} c_{ij}(\theta_i - \beta_j - log(1 + e^{\theta_i - \beta_j})) + (1 - c_{ij})log(1 - \frac{e^{\theta_i - \beta_j}}{1 + e^{\theta_i - \beta_j}})$$

Derivative with respect to θ_i

$$\frac{\partial \ell}{\partial \theta_i} = c_{ij}(1 - \frac{e^{\theta_i - \beta_j}}{1 + e^{\theta_i - \beta_j}} + (1 - c_{ij})(-\frac{e^{\theta_i - \beta_j}}{1 + e^{\theta_i - \beta_j}})$$

Derivative with respect to β_j

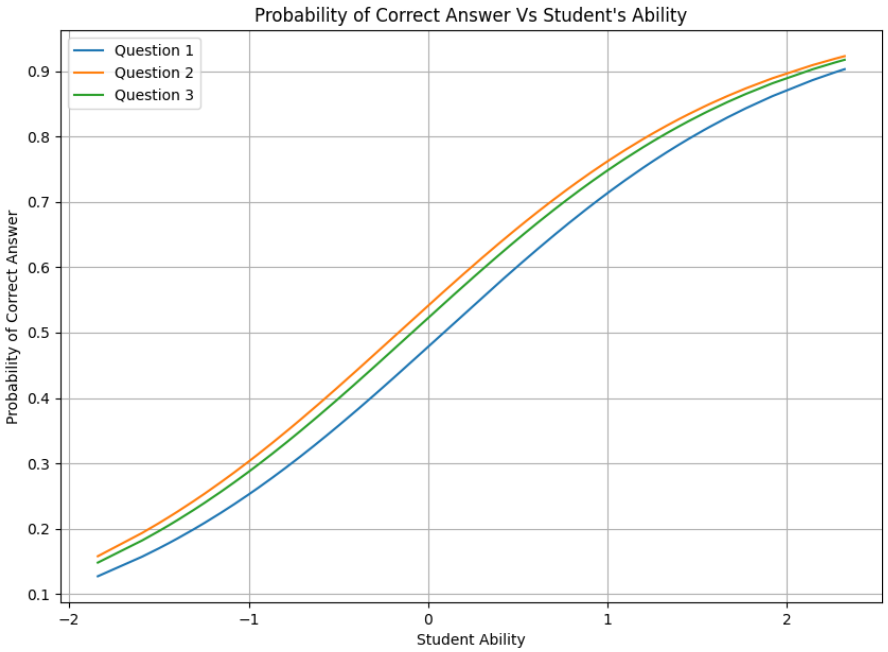
$$\frac{\partial \ell}{\partial \beta_j} = c_{ij}(-\frac{e^{\theta_i - \beta_j}}{1 + e^{\theta_i - \beta_j}})) + (1 - c_{ij})(\frac{e^{\theta_i - \beta_j}}{1 + e^{\theta_i - \beta_j}})$$

(b) Note: When implementing into python cannot loop over all i and j since not all students answered all questions so our sum is more like $\sum_{i,j \in (dataset)}$ meaning we are looking at pairs i and j that appear in the data set when implementing these functions

(c) Validation Accuracy: 0.7011007620660458

Test Accuracy: 0.6951735817104149

(d) The shape of the curves seems to resemble a sigmoid function after sorting theta. These curves the probability a student gets the correct answer for a certain question given the student's ability. More specifically, on our plot looking at the blue line, we can see that this represents the probability a student with ability x will get question 1 correct.



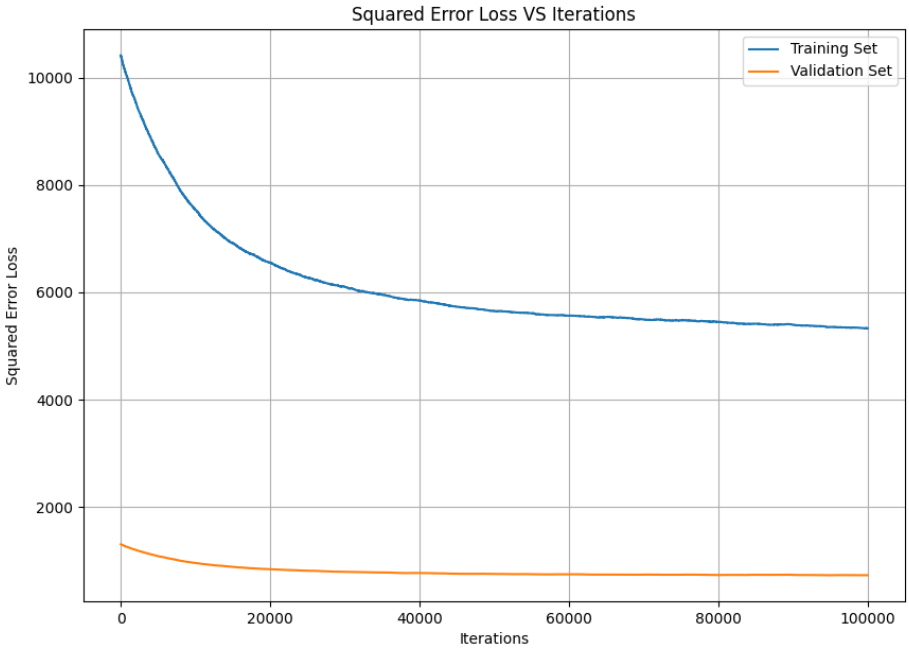
3 Matrix Factorization

(a) Best K value: 9
Validation Accuracy: 0.6613039796782387
Test Accuracy: 0.6587637595258256

(b) One limitation of SVD in this context is that it fills in the NAN values with the average of the current row. This could limit SVD's performance as filling in the data with the average could be an inn accurate representation of a student getting a question right, given the average response of the question. That is, if a certain question does not have a lot of data on it and the data we have is just students getting the question wrong. The mean matrix would treat the question as being answered wrong by everyone.

(c) Done in code

(d) Best K value: 25
Learning rate = 0.1
Number of Iterations = 100000



(e)
Validation accuracy : 0.6844482077335591
Test accuracy: 0.6861416878351679

4 Ensemble

To complete this we first implemented functions that would randomly sample from the original dictionary data set with replacement. This function was then called three times to train each of the three models from the previous questions. The models we are using are knn impute by item to compare questions rather than users from part a. The information response theory model from part b and the matrix factorization model from part c. The models are trained the same way from the previous questions, that is, knn returns a matrix filled with values showing the probability of each student getting a question correct. The IRT algorithm is trained by performing alternating gradient descent on the parameters θ and β. Finally, matrix factorization is trained using alternating least squares to optimize the U and Z matrix.

After training, the models are then used to compute predictions on both the validation and test set and output the ensembles predictions by averaging the predictions from the original models. Finally the accuracy of the ensemble is calculated on both the validation and test set. This process is looped to see the variance in different bootstrap samples and their effect on the accuracy.

After multiple runnings of the ensemble the validation accuracy ranges from 0.6933389782670054 - 0.696725938470223 and the test accuracy ranges from 0.6875529212531752 - 0.6968670618120237. On average they seem to both be ∼69% accuracy.

In comparison to the individual models:

KNN:
Validation Accuracy: 0.6816257408975445
Test Accuracy: 0.6816257408975445
IRT:
Validation Accuracy: 0.7011007620660458 Test Accuracy: 0.6951735817104149
Matrix Factorization:
Validation accuracy : 0.6844482077335591
Test accuracy: 0.6861416878351679

The average of the models' validation accuracy is ∼0.689
The average of the models' test accuracy is ∼0.688

The average of the models is slightly below the accuracy of the ensemble. This is expected as bagging does not really change the bias of the model, but it does reduce variance. That is, it reduces the impact of noise on the model so the model is more consistent with its prediction. In summary, the ensemble model in this case is more focused on improving stability and reducing variance rather than explicitly boosting accuracy. So while its performance does not increase its robustness does.

5 Part B Altering Matrix Factorization

1. Motivation and Alteration

   Our study on matrix factorization suggests that it is prone to over fitting especially on the sparse data we are dealing with. To combat this our group came up with a couple alterations to the original model. First we plan to apply L2 regularization to the cost function this should help penalize large values within the matrices and prevent over fitting. In the context of matrix factorization, these elements represent the latent factors that the model is learning. Without regularization, we discovered the model to be assigning some high values to some latent factors to fit the training data, leading to poor generalization to new data. With this addition we hope to see the model keep these values smaller, resulting in a more generalized model. Additionally, since the data is sparse, there are instances where certain students have answered only a few questions, and some questions have been answered by only a few students. This regularization should help prevent assigning excessively low to these latent factors. Next our group decided to also implement biases for both the user and question matrix to allow the model to capture individual preferences and item-specific characteristics. Biases allow the model to capture individual tendencies of students to rate questions higher or lower than their baseline expectations, and vice versa for questions. This is especially important in scenarios where certain students may have a general inclination to answer questions more better or worse, and certain questions may inherently be more challenging or easy. This could be indicative of the students' intellect or the questions' difficulty. Overall, to combat over fitting this combination helps the model better capture the underlying patterns in the data without fitting noise. Finally, we changed the optimization method. Instead of using ALS with SGD we are using alternating updates with Stochastic Mini Batch Gradient Descent. Where we sample without replacement and alternate between updating U and U's bias vector and then Z and Z's bias vector. This should help convergence as we take bigger steps towards convergence.

2. Formal Description

   Previously the goal of optimization was:

$$min_{\mathbf{U},\,\mathbf{z}} \frac{1}{2} \sum_{(n,m)\in O} (C_{nm} - \mathbf{u}_n^T \mathbf{z}_m)^2$$

We've changed this goal by adding L2 regularization and biases to the matrices. Let p represent the vector of user biases and let q represent the vector of question biases. So the new goal is to minimize this function:

$$min_{\mathbf{U},\ \mathbf{Z},\ \mathbf{p},\ \mathbf{q}} \frac{1}{2} \sum_{(n,m)\in O} (C_{nm} - (\mathbf{u}_n^T \mathbf{z}_m + p_n + q_m))^2 + \lambda(\sum_{n=1}^{N}(||\mathbf{u}_n||^2 + p_n^2) + \sum_{m=1}^{M}(||\mathbf{z}_m||^2 + q_m^2))$$

This function incorporates the biases into the prediction as well as within the regularization. We decided to incorporate the biases with their respective matrix (p with U and q with Z) to try and regularize them together and reduce the functions complexity. We alternatively thought about assigning a different λ to each regularization term, however we thought it would be too difficult to choose the best hyper parameters.

This change to the function aims to try and prevent over fitting the training data. Since we are dealing with a sparse data set matrix factorization tends to over fit our goal with regularization is to combat that. In regards to the biases we want to capture, what we presume to be, users skill and question difficulty in a vector. This way the model takes these extra conditions into account when making predictions.

We tackle this optimization problem slightly differently from before. We use alternating updates on the matrix U and vector p and on the following iteration on matrix Z and vector q. We also made use of mini batches, where we randomly sample without replacement and focus on optimizing the function above. More specifically each batch only updates U and p or Z and q.

These are the update rules:

$$\mathbf{u}_n \leftarrow \mathbf{u}_n(1 - \alpha\lambda) + \alpha((C_{nm} - (\mathbf{u}_n^T \mathbf{z}_m + p_n + q_m))\mathbf{z}_m - \lambda \mathbf{u}_n)$$

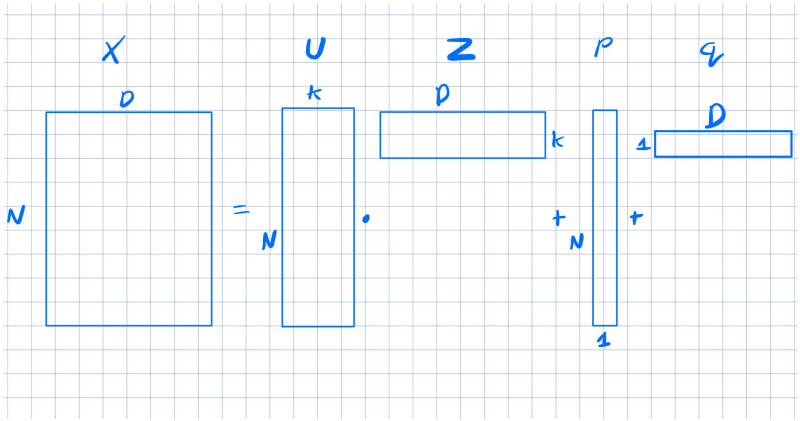
$$\mathbf{z}_m \leftarrow \mathbf{z}_m(1 - \alpha\lambda) + \alpha((C_{nm} - (\mathbf{u}_n^T \mathbf{z}_m + p_n + q_m))\mathbf{u}_n - \lambda \mathbf{z}_m)$$

$$p_n \leftarrow p_n + \alpha((C_{nm} - (\mathbf{u}_n^T \mathbf{z}_m + p_n + q_m)) - \lambda p_n)$$

$$q_m \leftarrow q_m + \alpha((C_{nm} - (\mathbf{u}_n^T \mathbf{z}_m + p_n + q_m)) - \lambda q_m)$$

3. Figure or Diagram

Here is an overview of what we are trying to achieve:



Here the vector p is being added along the row, that is the first element of p is added to the entire first row of X. Similarly the first entry of q is being added to the entire first column of X. This pattern is the same for the rest of the indices.

The goal here is to deconstruct the matrix into 4 matrices. A matrix for user latent factors (U), a matrix for question latent factors (Z), a vector for user biases (p), and a vector for question biases (q).

4. Comparison and Demonstration

Model	Validation Accuracy	Test Accuracy
Original Model	0.6844482077335591	0.6861416878351679
Altered Model	0.7081569291560824	0.7042054755856618

Confusion Matrix for Validation Set on Original Matrix Factorization Model:

	TP: 1766	FN: 1063
	FP: 1121	TN: 3136

Confusion Matrix for Validation Set on Altered Matrix Factorization Model:

	TP: 1674	FN: 1155
	FP: 913	TN: 3344

Confusion Matrix for Test Set on Original Matrix Factorization Model:

	TP: 886	FN: 546
	FP: 569	TN: 1542

Confusion Matrix for Test Set on Altered Matrix Factorization Model:

	TP: 856	FN: 576
	FP: 472	TN: 1639

On the validation set we see that the original model is able to get more True Positives, however the altered model does much better on True Negatives.

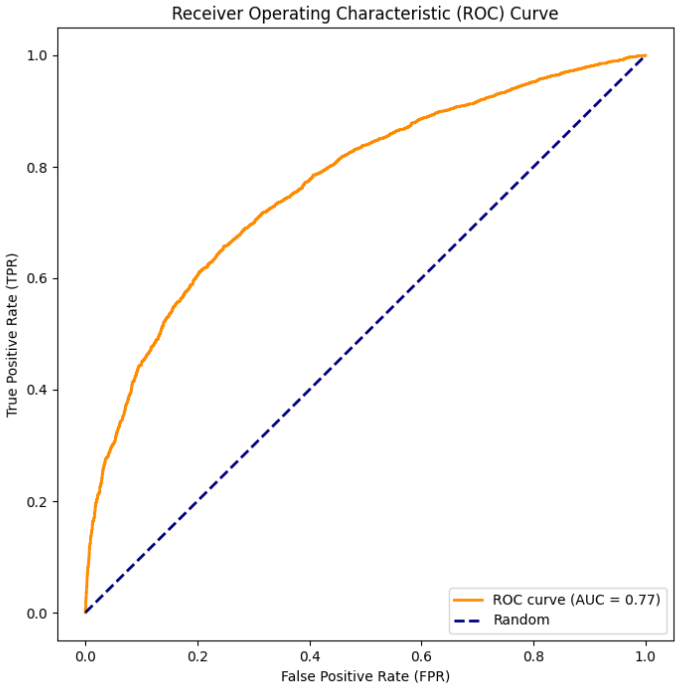
On the test set we see something similar, except that both models seem to be getting the same amount of True Postives.

In terms of the squared error loss, we see from the previous plot the original model has:
Training squared error loss = 5574.156082436201
Validation squared error loss = 703.585661823341

While the altered model has:
Training squared error loss = 4978.281324038328
Validation squared error loss = 678.924730140541

In both these cases we can see the model performs slightly better with a slightly lower squared error loss on both the training and validation sets.

ROC curve for the Validation Set



We can see that the optimal threshold that balances TP and FN is ∼0.6

To disentangle the benefits we tried our altered model with just the mini batch change and just the regularization + bias change. It seems that the benefits are coming from the addition of regularization and bias to the model. As from our testing we got similar results with just regular stochastic gradient descent and the regularization and bias change. Whereas with just the mini batch change our model seemed to perform similarly to the original model. So it seems the regularization and the bias change seems to be improving the models accuracy while the optimization method has little to no effect on the model's performance.

5. Limitations

Due to our method not making use of the metadata it might struggle to learn the biases for the user and question matrices. To extend on this we could try to incorporate the meta data into our solution and see if we can understand a students fluency in a type of question and generalize certain questions' difficulty. This should help learn the biases better as the model could detect patterns better with more user and question data.

This model relies on data collected from students who have completed questions. However, this model will probably not perform well if a new student is added to the set with no or very limited data. Our method and matrix factorization in general could struggle as it would be training its matrices based on the other students neglecting the new which could cause for inaccurate predictions. A solution would be to not add new students without some sort of data on them, or similar to the previous limitation implement metadata so the model could have some sort of inclination on what the students strengths and weaknesses may be.

Our model although has biases it might struggle in capturing the intricacies of a given student's proficiency or what makes a given question difficult. Students exhibit varying levels of proficiency, engagement, and learning styles. A single model that assumes some sort of homogeneity might not effectively adapt to these individual differences. It might not be best to treat all students the same when there could be great deal of diversity between them. This could make the model struggle at generalizing. One extension could be to group students with similar skill and create a model for each of these clusters. This way each model can be used to predict on a certain student's probability while taking account a lot of intricacies that would not be taken into account with a single model.

Finally, from the confusion matrix it seems that the model has a relatively high specificity but it has a lower

sensitivity. The original matrix factorization solution seems to also be exhibiting a similar behaviour. It may be useful to train another model that has a higher sensitivity and use those models in an ensemble. This way we can try to boost both the models sensitivity and specificity.