

CSC311 - Final Assignment

Zelong Liu, Fizzah Mansoor, Harrison Deng

December 3, 2021

Contents

I	Predicting Student Correctness	3
1	K-Nearest Neighbor	3
a	Complete Main kNN, Plot and Report Accuracy	3
b	Selecting k^*	4
c	Implementing Impute by Item	4
d	Comparing user and item based Collaborative Filtering	4
e	Potential Limitations of kNN in this Context	5
2	Item Response Theory	6
a	Mathematical Derivations for IRT	6
b	Implementation of IRT	7
c	Reporting Validation and Test Accuracies	8
d	Plots of Questions With Respect to θ and β	8
3	Neural Networks	9
a	Differences Between ALS and Neural Networks	9
b	Implementing AutoEncoder	9
c	Tuning and Training NN	9
d	Plotting and Reporting	9
e	Implementing L_2 Regularization	9
4	Ensemble	10
II	Modifying for Higher Accuracy	11
5	Formal Description	11
6	Figure or Diagram	11
7	Comparison or Demonstration	11

Part I

Predicting Student Correctness

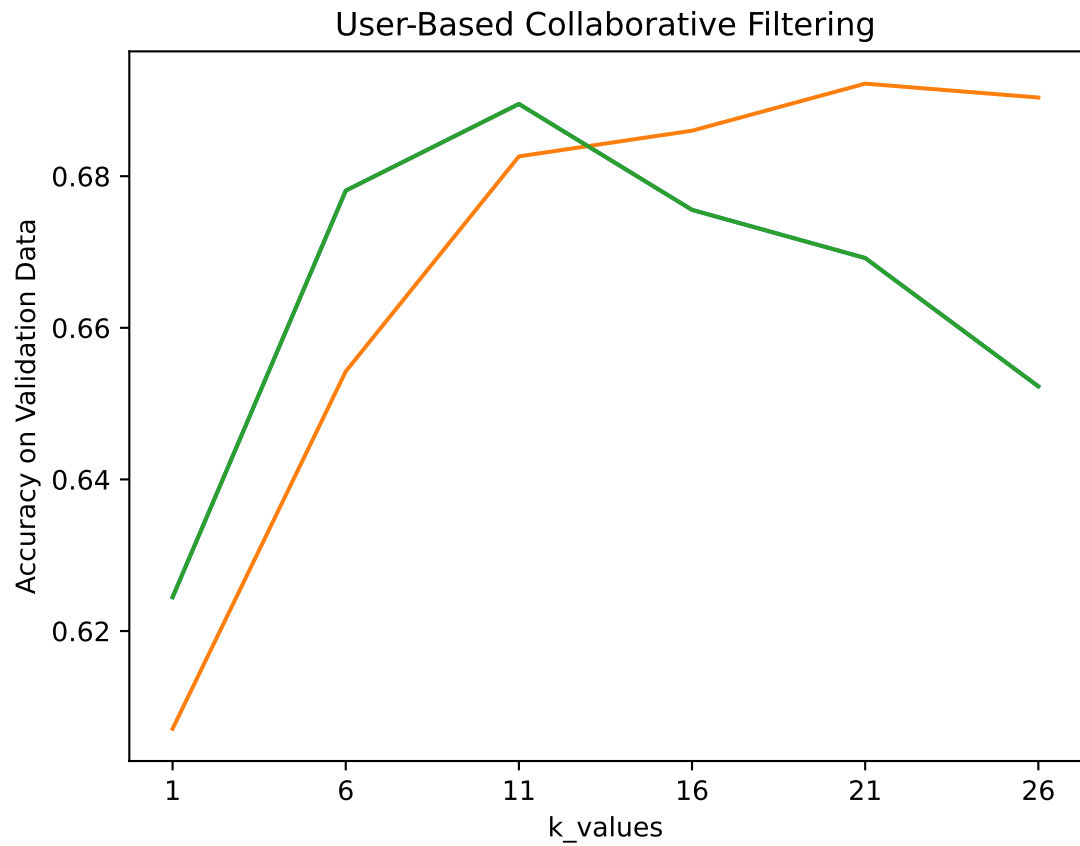
1 K-Nearest Neighbor

a Complete Main kNN, Plot and Report Accuracy

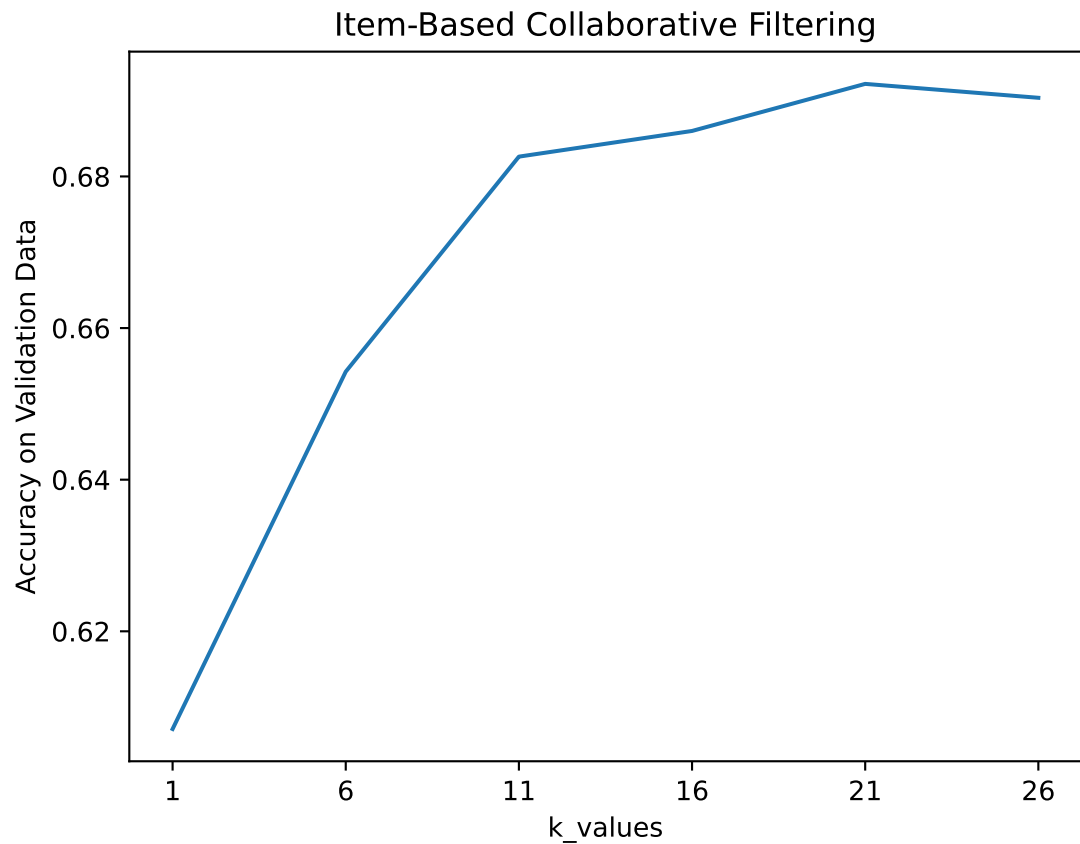
The implementation of all code is in the `part_a/knn.py` file. Following are plots of accuracy on validation data as a function of $k \in \{1, 6, 11, 16, 21, 26\}$:

```
import codebase.part_a.knn as knn
k_vals, val_user_acc, val_item_acc = knn.main("./codebase/data")

knn.accuracy_plot(k_vals, val_user_acc, "User-Based Collaborative
↪ Filtering")
```



```
knn.accuracy_plot(k_vals, val_item_acc, "Item-Based Collaborative  
→ Filtering")
```



b Selecting k^*

c Implementing Impute by Item

Underlying assumption: if answers by certain users to Question A match those of Question B, then As answer correctness corresponding to a specific user matches that of question Y.

d Comparing user and item based Collaborative Filtering

User-Based collaborative filtering performs better on test data. 68.416% accuracy on user-based filtering and 68.162% accuracy on item-based filtering.

e Potential Limitations of kNN in this Context

We can safely assume that there is a high correlation between both question difficulty and student ability on whether or not the question was answered correctly. But, feature importance is not possible for the KNN algorithm (there is no way to define the features which are responsible for the classification), so it will not be able to make accurate inferences based on these two parameters. In the algorithm used in this question, either one of the two parameters (user ability or question difficulty) is focused on, so it has lower validation and test accuracy scores than other algorithms in Part A of this project.

KNN runs slowly. Finding the optimal k-value from the given list of possible k values (1, 6, 11, 21, 26) takes several minutes for each function.

2 Item Response Theory

a Mathematical Derivations for IRT

We are given that $p(c_{ij} = 1|\boldsymbol{\theta}, \boldsymbol{\beta})$. We will assume c_{ij} is a value in \mathbf{C} where i and j as coordinates are in set O as defined:

$$O = \{(i, j) : \text{Entry } (i, j) \text{ of matrix } \mathbf{C} \text{ and is observed}\}$$

Since this c_{ij} is a binary value, we can describe $P(\mathbf{C}|\boldsymbol{\theta}, \boldsymbol{\beta})$ with a bernoulli distribution:

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

Therefore, our Likelihood function is:

$$L(\boldsymbol{\theta}, \boldsymbol{\beta}) = \prod_{ij} \left[\frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)} \right]^{c_{ij}} \left[\frac{1}{1 + \exp(\theta_i - \beta_j)} \right]^{(1-c_{ij})}$$

Then, apply log to obtain the log-likelihood where N and M are the number of users and questions respectively:

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

Then, we solve for the partial derivative with respect to θ_i and β_j respectively:

$$\begin{aligned} \frac{\delta}{\delta \theta_i} &= \sum_{j=1}^M \left[c_{ij} - \frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)} \right] \\ \frac{\delta}{\delta \beta_j} &= \sum_{i=1}^N \left[-c_{ij} + \frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)} \right] \end{aligned}$$

b Implementation of IRT

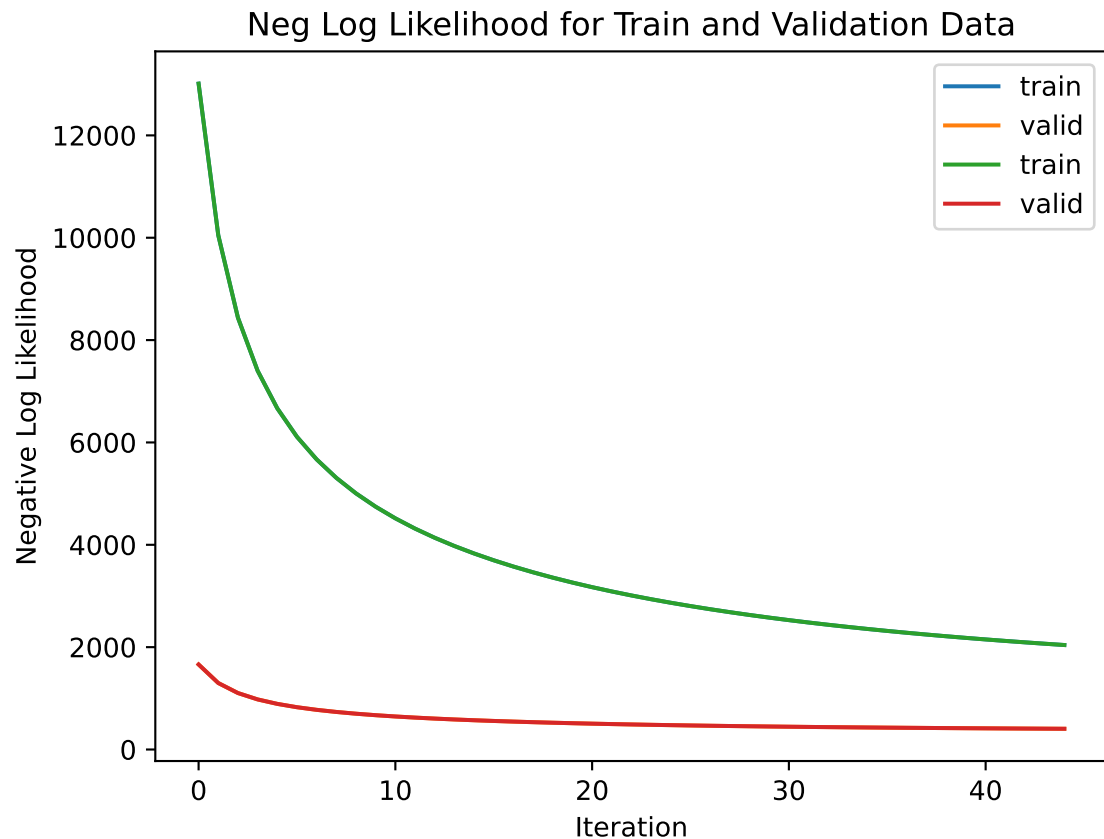
The implementation of IRT is in `part_a/item_response.py`. We chose the hyperparameters α and iterations number by performing multiple combinations of them and seeing which one had the highest validation score (automated, see mentioned code file for this automation). We then manually adjusted the set of tested values and repeated. Doing this a few times resulted in:

```
import codebase.part_a.item_response as irt
print()
irt_results = irt.main("./codebase/data")
```

Training with lr of 0.005 and 45 number of iterations.
Final accuracy: 0.7056167090036692
lr*: 0.005 iterations*: 45 acc*: 0.7056167090036692
test accuracy: 0.7070279424216765

The following is a plot of the training curve (and the code used to generate it):

```
plot(irt_results["train_nllks"], label="train")
plot(irt_results["val_nllks"], label="valid")
ylabel("Negative Log Likelihood")
xlabel("Iteration")
title("Neg Log Likelihood for Train and Validation Data")
legend()
```



c Reporting Validation and Test Accuracies

Validation and test accuracies have been calculated in the previous call to the main function. Again, implementation is in `part_a/item_response.py`.

Our validation accuracy:

```
print(irt_results["val_acc"])
```

```
0.7056167090036692
```

Test accuracy:

```
print(irt_results["test_acc"])
```

```
0.7070279424216765
```

d Plots of Questions With Respect to θ and β

3 Neural Networks

- a Differences Between ALS and Neural Networks
- b Implementing AutoEncoder
- c Tuning and Training NN
- d Plotting and Reporting
- e Implementing L_2 Regularization

4 Ensemble

Part II

Modifying for Higher Accuracy

5 Formal Description

6 Figure or Diagram

7 Comparison or Demonstration