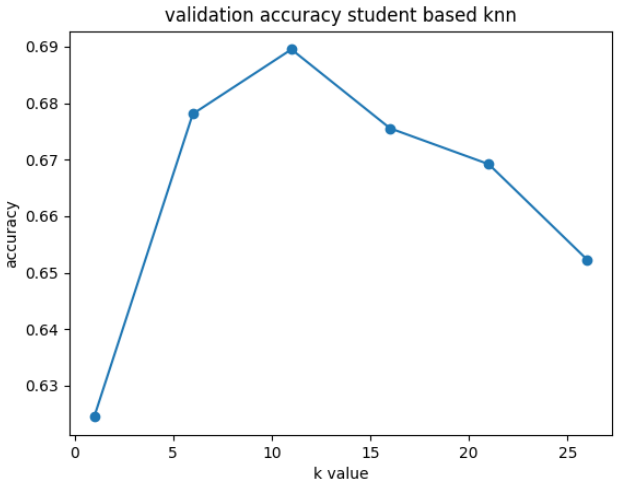
Part 1

Q1.

(a)

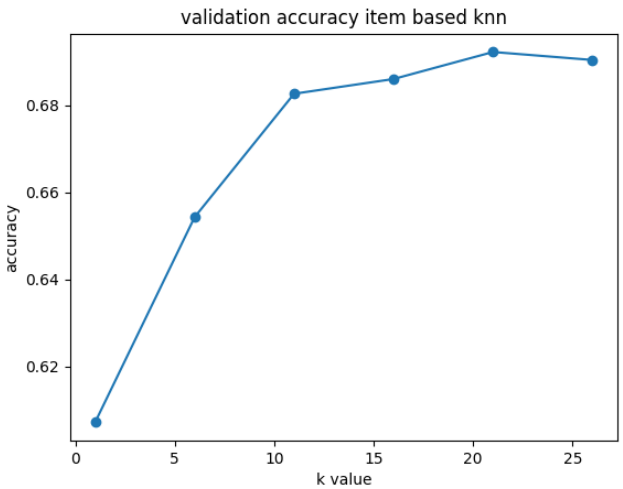



```
k-nn based on student similarity
Validation Accuracy
k = 1: 0.6244707874682472
k = 6: 0.6780976573525261
k = 11: 0.6895286480383855
k = 16: 0.6755574372001129
k = 21: 0.6692068868190799
k = 26: 0.6522720858029918
```

(b)

```
Test accuracy with chosen k*
k* = 11: 0.6841659610499576
```

(c)




```
k-nn based on item similarity
Validation Accuracy
k = 1: 0.607112616426757
k = 6: 0.6542478125882021
k = 11: 0.6826136042901496
k = 16: 0.6860005644933672
k = 21: 0.6922099915325995
k = 26: 0.69037538808919
Test accuracy with chosen k*
k* = 21: 0.6683601467682755
```

If a question A and question B are both answered correctly by a student, A and B are more likely to be correctly answered by a different student than that of a randomly chosen question. Similar assumption holds for wrongly answered questions.

(d) user-based collaborative filtering performs better. It has higher test accuracy.

(e)
- Time consuming. The algorithm needs to find the k closest neighbours for each data point which needs to go through every data points. Although the data has small dimension, we would have a huge data set (if we were given the full data set).
- Not accurate. Given dataset contains many missing values. What the KNNImputer do here is to use the mean value from k nearest neighbours found in the training set. However, the true value for those missing data might be very different.

2.

(a)

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

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

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

$$p(C \mid \theta, \beta) = \prod_i \prod_j p(C_{ij} \mid \theta_i, \beta_j)$$

$$\log p(C \mid \theta, \beta) = \sum_i \sum_j \log p(C_{ij} \mid \theta_i, \beta_j)$$

$$= \sum_i \sum_j \left[C_{ij}(\theta_i - \beta_j) - \log(1+\exp(\theta_i-\beta_j)) \right]$$

$$\frac{\partial}{\partial \theta_i} \log p(C \mid \theta, \beta) = \sum_j \left[C_{ij} - \frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)} \right]$$

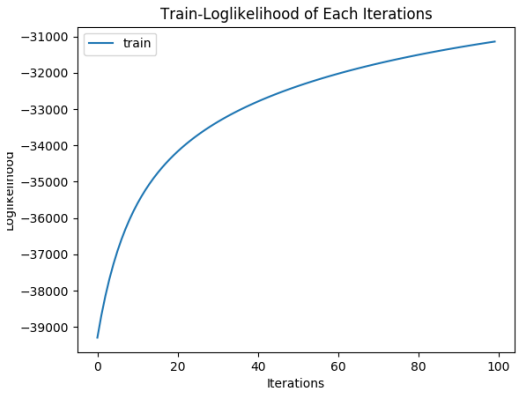
$$\frac{\partial}{\partial \beta_j} \log p(C \mid \theta, \beta) = \sum_i \left[-C_{ij} + \frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)} \right]$$

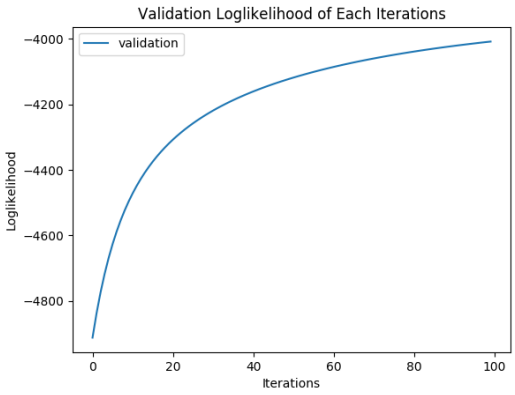
(b)

hyperparameters:

```
lr: 0.001
iterations: 100
```

plots:

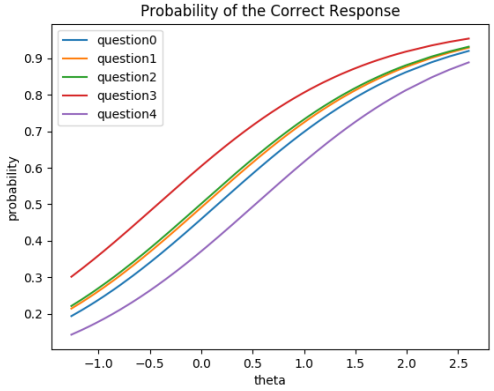




(c)

```
Validation Accuracy: 0.7071690657634773
Test Accuracy: 0.6994072819644369
```

(d)



Given a question j, the probability of the correct response is linearly associated with theta. The association is linear and strong.
Because theta represents the student's ability, the shape of the curves represents that given a question j, the higher level a student's ability, the higher the probability that he could response the question correctly.

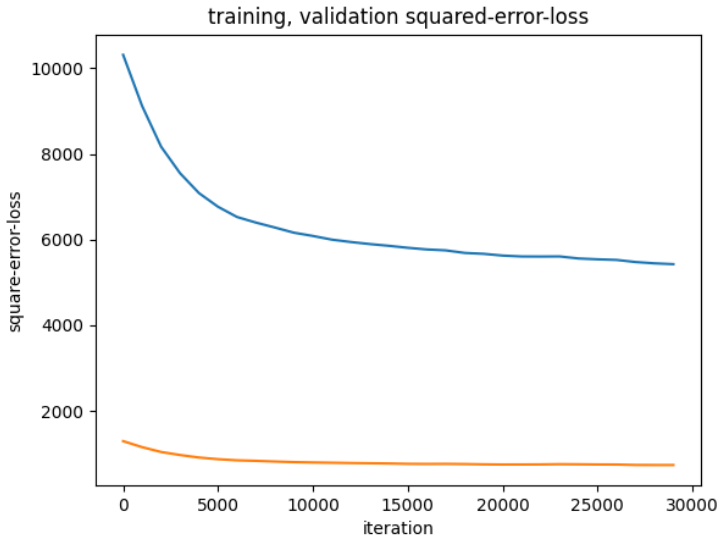
Q3 Option 1 Matrix Factorization

(a) chosen K=9
validation accuracy : 0.661304 test accuracy : 0.658763

(b) Our data contains many missing value. In 'svd_reconstruct' those missing values were filled by the average of whole data set. However, we do not know the true value of those missing data. So our transformation is inaccurate.

(d) learning rate = 0.15 # iteration = 30,000

(e) $k^* = 101$ validation accuracy = 0.681767 test accuracy = 0.682190



(f) we would use logistic-cross-entropy function as the loss function

$$u_n^T z_m \log(1+e^{-c_{nm}}) + (1-u_n^T z_m)\log(1+e^{c_{nm}})$$

4.

```
def bootstrap(data, val_data, size, iterations):
    """ Bootstrapping the data to get a sample and use irt to get a model.

    :param data: the sparse matrix of the training set
    :param val_data: A dictionary {user_id: list, question_id: list,
    is_correct: list}
    :param size: size of the bootstrap sample
    :iterations: number of iterations in the irt algorithm
    :return: beta, theta from irt
    """
    sample = resample(data, n_samples=size, replace=True)
    theta, beta, train_nllk_lst, valid_nllk_lst = \
        item_response.irt(sample, val_data, 0.001, iterations)
    return beta, theta
```

First, we generate 3 different datasets each by sampling students from the training set with replacement. The 3 new datasets have exactly the same number of students as the original training set, including repetitions. Based on the item response theory, we train a new model from each dataset and get 3 sets of theta and beta.

```
def evaluate(data, betas, thetas):
    """ Evaluate the ensemble model given data and return the accuracy.

    :param data: A dictionary {user_id: list, question_id: list,
    is_correct: list}
    :param betas: List of vectors
    :param theta: List of vectors
    :return: float
    """
    pred = []
    for i, q in enumerate(data["question_id"]):
        u = data["user_id"][i]
        p_a = 0
        for j in range(len(betas)):
            beta = betas[j]
            theta = thetas[j]
            x = (theta[u] - beta[q]).sum()
            p_a += item_response.sigmoid(x)
        p_a = p_a / len(betas)
        pred.append(p_a >= 0.5)
    return np.sum((data["is_correct"] == np.array(pred))) \
           / len(data["is_correct"])
```

Then, based on each set of theta and beta, we calculate the probability of correct response for each response in the validation and test set and get 3 probabilities. After that, we take the mean of these 3 probabilities as the probability of correct response for a given response. If the probability is larger or equal to 0.5, we predict it to be response correctly; otherwise, it would be predicted to be responded incorrectly. Finally, we compare our predictions to the results and get final validation and test accuracy.

```
Validation Accuracy: 0.6114874400225797
Test Accuracy: 0.6110640699971776
```

The results are slightly lower than that before the ensemble process, probably because the total number of students is not large enough compare to the number of responses: each student has approximately 100 responses. In this case, we resample based on the students but the prediction is based on specific responses, so there are lots of repetitions and correlations in the newly generated data sets. Therefore, the model after the ensemble process is even weaker.

Part B

1. Formal Description

- Our algorithm extends the IRT model in Question 2 of Part 1.We add a new parameter k_j to the original model, representing the discrimination of question j. Now, the algorithm becomes a two-parameter IRT model, in which β_j represents the difficulty of the question j, θ_i represents the i-th student ability, and k_j represents the probability of endorsing correctly answering question j.
- The probability that question j is correctly answered by student i is formulated as:

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

- The likelihood function is formulated as:

$$p_{ij}(\theta_i,\ \beta_j,\ k_j) = \frac{exp(c_{ij}k_j(\theta_i - \beta_j))}{1+exp(k_j(\theta_i - \beta_j))}$$

- We train the model by maximizing the log-likelihood function:

$$logp_{ij}(\theta_i,\ \beta_j,\ k_j) = c_{ij}k_j(\theta_i - \beta_j) - log[1 + exp(k_j(\theta_i - \beta_j))]$$

- By the rule of gradient descent, we update the parameters:

$$\theta_i \leftarrow \theta_i + c_{ij}k_j - \frac{exp(k_j(\theta_i - \beta_j))}{1+exp(k_j(\theta_i - \beta_j))} * k_j$$

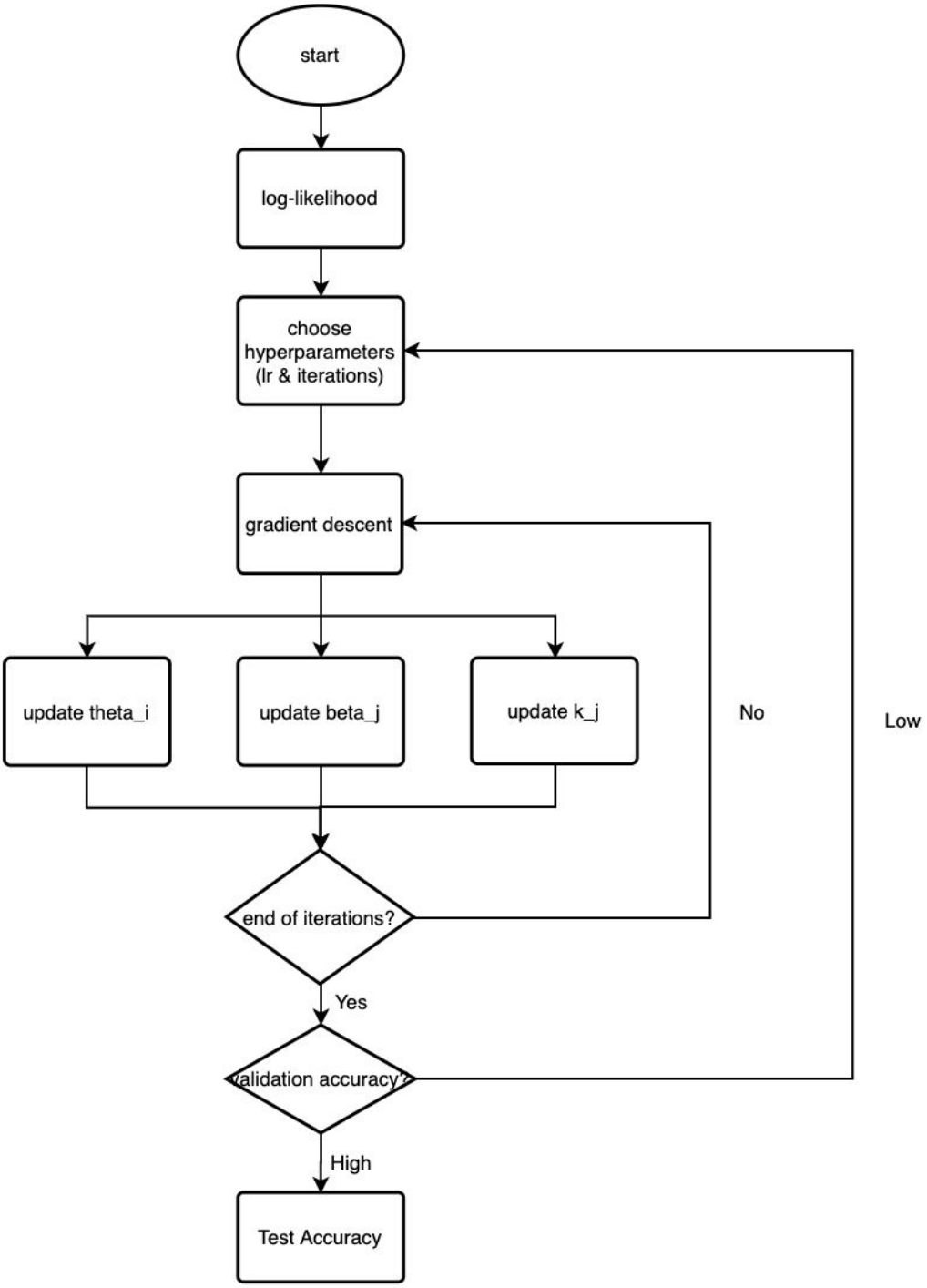
$$\beta_j \leftarrow \beta_j - c_{ij}k_j + \frac{exp(k_j(\theta_i - \beta_j))}{1+exp(k_j(\theta_i - \beta_j))} * k_j$$

$$k_j \leftarrow k_j + c_{ij}(\theta_i - \beta_j) - \frac{exp(k_j(\theta_i - \beta_j))}{1+exp(k_j(\theta_i - \beta_j))} * (\theta_i - \beta_j)$$

- We expect our model would improve the optimization as we introduced a new parameter.

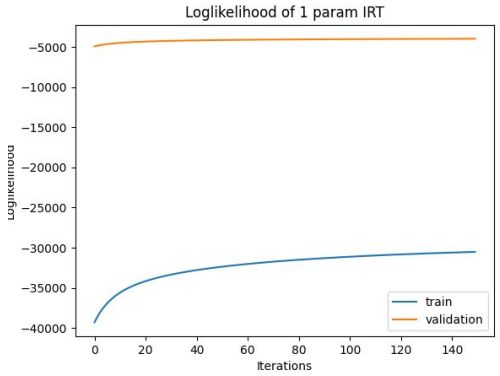
2. Figure or Diagram

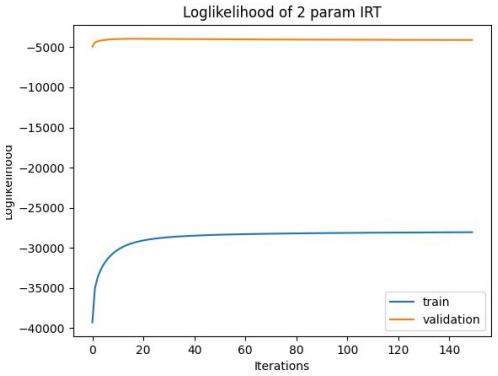
- Flow chart



3. Comparison or Demonstration

- The log-likelihood given by the two models are quite similar.



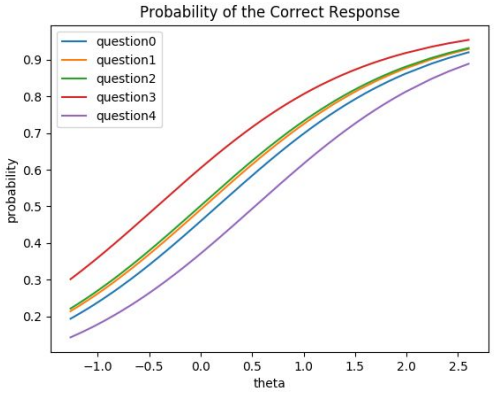


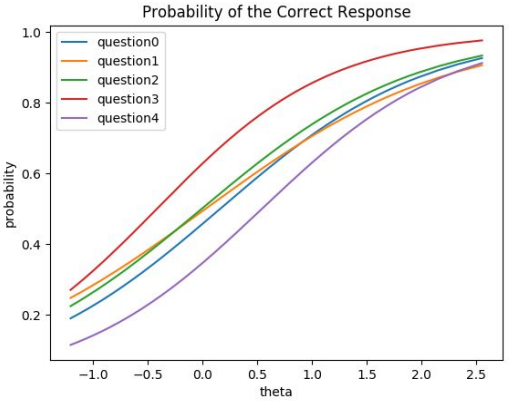
- Accuracy

	1 param IRT	2 param IRT
Validation Accuracy	0.70717	0.70731
Test Accuracy	0.69941	0.70138

Both the validation accuracy and the test accuracy are only slightly higher in the 2 parameter IRT model, so the improvement of the model is very small.

- Probability of Correct Response





The discrimination parameter k_j is allowed to vary between questions. Henceforth, the probability of correct response of the different questions can intersect and have different slopes. The steeper the slope, the higher the

discrimination of the item, as it will be able to detect subtle differences in the ability of the respondents. Therefore, the model can be optimized by adding this parameter.

- To test our hypothesis (2-param IRT model improves optimization), we trained models the same way as we trained 1-param IRT model, except we use different log-likelihood function and gradient descent rules. Keeping the hyperparameters the same at first, then tune them to find the values that work the best.

4. Limitations:

- limitation:
  - There are four assumptions we need to meet before fitting to an IRT model - monotonicity, unidimensionality, local independence, and invariance. In the context of our study, local independence cannot always be satisfied. For instance, question B requires the answer from question A; questions A and B are testing the same concept. Therefore we cannot ensure that responses given by one student are mutually independent. This is inevitable under our task as long as we want to fit an IRT model. However, we can minimize the effect of correlation if we had a large data size.
  - IRT models are unidimensional. We can only predict correct/incorrect response.
- possible extension:
  - adding another parameter to our existing formula, representing the probability of getting the question right via. random guess

$$p(c_{ij}|\theta_i,\ \beta_j)\ =\ c + [1-c] * \frac{exp(k_j(\theta_i - \beta_j))}{1+exp(k_j(\theta_i - \beta_j))}$$

  - This c-parameter indicates the probability that very low ability students will get this question correct by chance, mathematically represented as a lower asymptote. A two-parameter IRT model assumes that the guessing parameter c is 0.

References

1. Item Response Theory. (n.d.). Retrieved December 16, 2020, from https://www.publichealth.columbia.edu/research/population-health-methods/item-response-theory

Contribution

Q1 - Wenfei Wang

Q2 - Haiyue Yang

Q3 - Yijing Chen, Wenfei Wang

Q4 - Haiyue Yang

Part B - All members