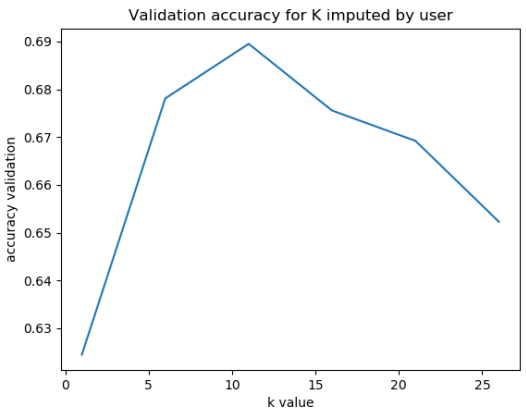
1.

(a)

```
Validation Accuracy: 0.6244707874682472
Validation Accuracy: 0.6780976573525261
Validation Accuracy: 0.6895286480383855
Validation Accuracy: 0.6755574372001129
Validation Accuracy: 0.6692068868190799
Validation Accuracy: 0.6522720858029918
```

← k = 1
← k = 6
← k = 11 max
← k = 16
← k = 21
← k = 26

Then code in knn.py



(b)

```
For user part we choose k = 11
Validation Accuracy: 0.6841659610499576
The final test accuracy is 0.6841659610499576
```

By user validation accuracy set.

We choose k = 11

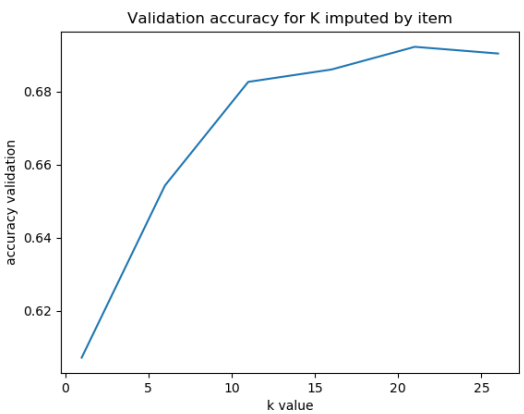
Then final test accuracy is:

0.6841659610499576 for k = 11

(c)

```
For k = 1
validation accuracy imputed by item is:
0.607112616426757
For k = 6
validation accuracy imputed by item is:
0.6542478125882021
For k = 11
validation accuracy imputed by item is:
0.6826136042901496
```

```
For k = 16
validation accuracy imputed by item is:
0.6860005644933672
For k = 21
validation accuracy imputed by item is:
0.6922099915325995
For k = 26
validation accuracy imputed by item is:
0.69037538808919
```



k=21
accuracy is max

Based on item validation accuracy

we choose k = 21

which we get

```
For item part we choose k = 21
Validation Accuracy: 0.6683601467682755
The final test accuracy is 0.6683601467682755
```

Final test accuracy is 0.6683601467682755

(d) By user validation accuracy set.

We choose $k = 11$

Then final test accuracy is:

0.6891659610499576 for $k = 11$

Based on item validation accuracy

we choose $k = 21$

Final test accuracy is 0.6683601467682275

we get a better performs for user-based collaborative

(e)

(1) Knn need a long computation time with lots of memory (A lot of matrix multiplication)

(2) For task given we have 542 students and 1774 diagnostic questions, that may lead to curse of dimensionality.

(3) prediction accuracy is a little bit low.

2.

(a)

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

For all students and questions

$$p(C \mid \theta, \beta) =$$

$$\prod_{i=1}^{542} \prod_{j=1}^{1774} \left[\left(\frac{e^{(\theta_i - \beta_j)}}{1 + e^{(\theta_i - \beta_j)}} \right)^{I(C_{ij}=1)} \left(\frac{1}{1 + e^{(\theta_i - \beta_j)}} \right)^{I(C_{ij}=0)} \right]$$

So $\log p(C \mid \theta, \beta)$

$$= \sum_{i=1}^{542} \sum_{j=1}^{1774} \left[I(C_{ij}=1)(\theta_i - \beta_j) - I(C_{ij}=0 \text{ or } C_{ij}=1) \log(1 + e^{\theta_i - \beta_j}) \right]$$

$$\frac{\partial \log p(C \mid \theta, \beta)}{\partial \theta_i}$$

$$= \sum_{j=1}^{1774} \left[I(C_{ij}=1) - I(C_{ij}=0 \text{ or } C_{ij}=1) \frac{e^{\theta_i - \beta_j}}{1 + e^{\theta_i - \beta_j}} \right]$$

$$\frac{\partial \log p(C|\theta,\beta)}{\partial \theta_i} = \sum_{i=1}^{54} \left[-I(c_{ij}=1) + I(c_{ij}=0 \text{ or } c_{ij}=1) \frac{e^{\theta_i-\beta_j}}{1+e^{\theta_i-\beta_j}} \right]$$

(b) learning rate : 0.01

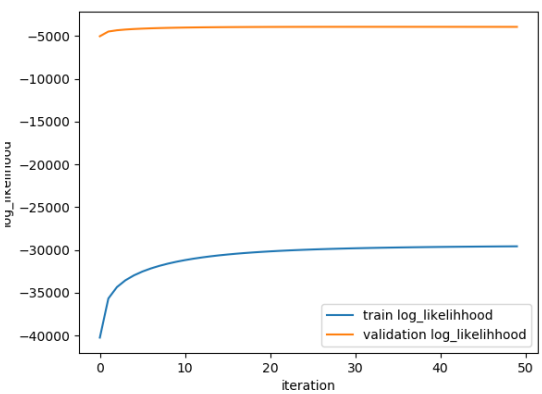
Number of iterations : 50

θ zero vector

β zero vector.

} hyperparmeters we selected

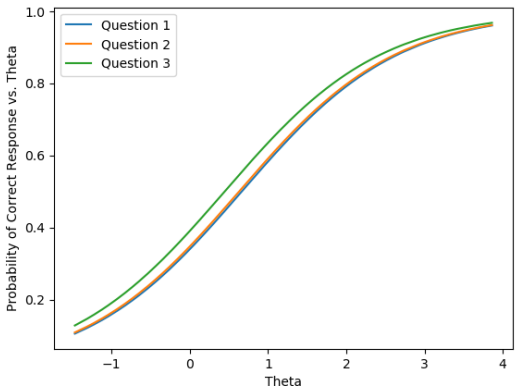
Training curve



(c)

```
For learning rate = 0.01, iteration = 50
the final validation accuracy is 0.7058989556872707
the final test accuracy is 0.7075924357888794
```

(d) The curve of 3 Questions



These curve shows that probability of correct response increase as the ability of student increase.

Q3

Option 1

a) different k values and the accuracy:

```
1 0.6428168219023427
5 0.659046006209427
10 0.6586226361840248
25 0.6594693762348293
50 0.648461755574372
100 0.6470505221563647
```

Take $k = 25$, which gives best validation result, the final validation and test performance:

```
accuracy with validation data:  0.6594693762348293
accuracy with test data:  0.6556590460062094
```

b) SVD is filling missing entries, so it's using data that actually does not exist for prediction, which can make the results inaccurate.

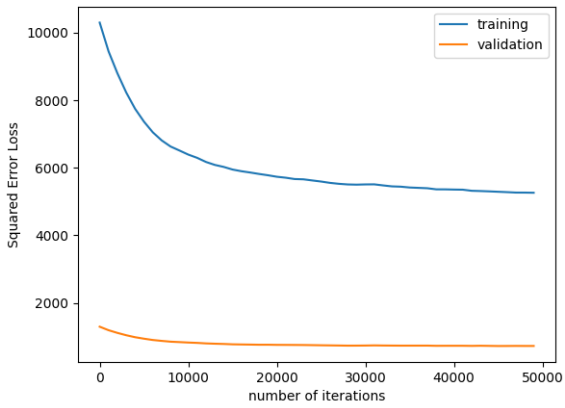
c) see code

d) $num\_iteration = 50000$, $lr = 0.1$ is the chosen hyperparameters. It provides good results and convergence rate is not too slow. Different k values and accuracy:

```
1 0.6762630539091166
5 0.6824724809483489
10 0.6847304544171606
25 0.6847304544171606
50 0.6939034716342083
100 0.6851538244425628
```

Take $k = 50$, which gives best validation result.

e) Training and validation squared-error-losses



The final validation accuracy is 0.691645, the final test accuracy is 0.688117

Q4

We would like to see if bagging improve performance of matrix factorization. So we generate 3 training data with size being the same as the original training data. We also use the same hyperparameter as in Q3 to see if there is an improvement in the accuracy with all other settings being the same. Since matrix factorization predicts with real valued probabilities, we can directly average the result to get averaged prediction of the models.

Accuracy of validation data: 0.679274; accuracy of test data: 0.679744. Comparing to the result from Q3, there is no sign of better performance with ensemble. This is probably because ensemble reduces overfitting, and overfitting is not happening in this setting. So bagging does not improve performance.

Part B

1

We decided to make two improvements to our model.

First, we observe that the base model is treating all the questions as the same. In other word, the model believes all question reflect same level of understanding on a topic, which is unlikely to be true in reality. We believe some questions are more reflective than others on how a student understands the question. For example, if most people answer a question correctly, we believe the prediction power of this question on whether the user understand the material is quite weak. If most people answer a question wrongly, the prediction power of that question is also quite weak.

So we decide to apply a weight to each question for each correct value. This should help improve the optimization. If the question is weak on prediction power, it has less weight during our calculation.

We calculated the average correctness of all questions, which is about 0.6, so we choose 0.6 as the base. If the correctness of a question deviates a lot from 0.6, we consider it not having enough prediction power.

Another thing we add is two regularization term in the cost function. This is because after we add the weight the model becomes more important, and we want to add some regularization term to prevent it from overfitting. From our research online, these two terms are quite common for ALS model, so we choose this.

Now our cost function becomes:

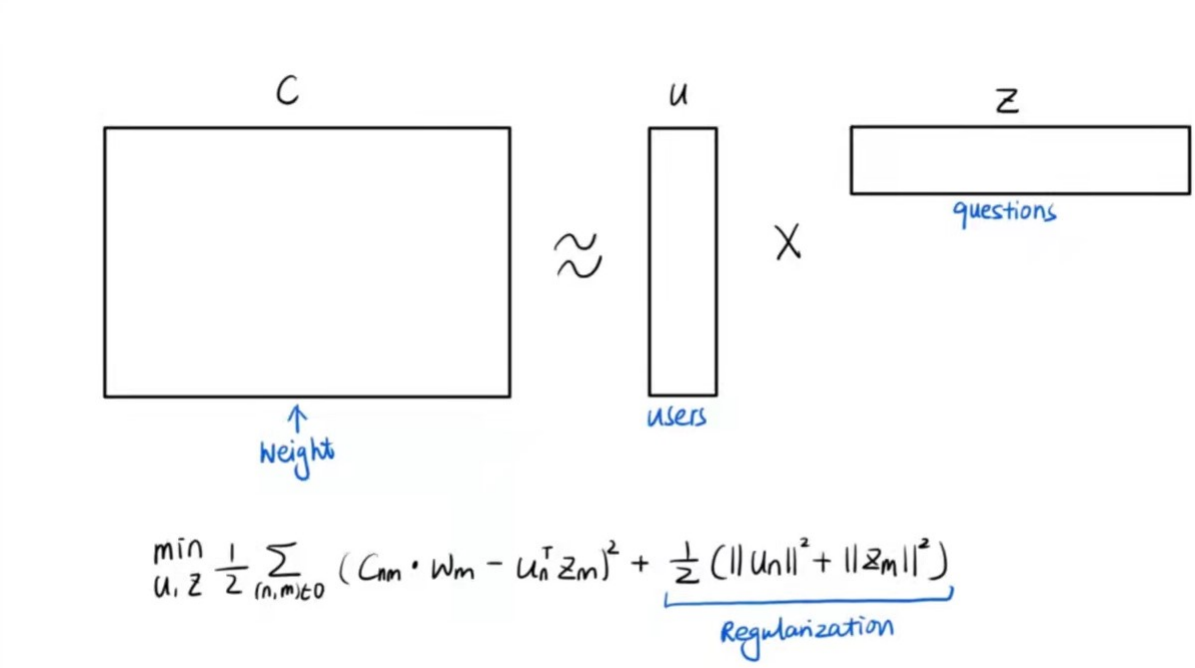
$$\min_{U,Z} \frac{1}{2} \sum_{(n,m)\in O} (C_{nm} \cdot w_m - u_n^\top z_m)^2 + \frac{1}{2}]\lambda(||u_n||^2 + ||z_m||^2)$$

And the gradient descent becomes:

$$u = u + lr(c \cdot w - u \cdot z) \cdot z + \lambda||u||$$

$$z = z + lr(c \cdot w - u \cdot z) \cdot u + \lambda||z||$$

2



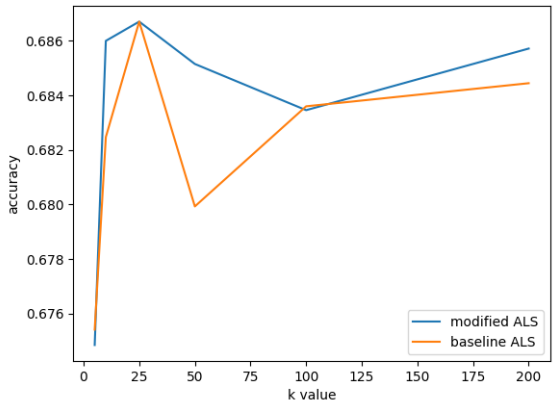
3

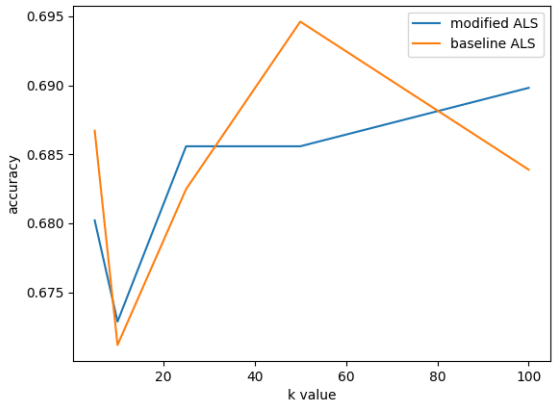
After experimenting with a few hyper-parameters, our accuracy on validation set and test set are as followed. The hyperparaters used are $k = 50, lr = 0.1, \lambda = 0.01, num\_iteration = 50000$. (results can vary with same input)

```
validation set accuracy:  0.7079565340107254
test set accuracy:  0.6944482077335591
```

The final validation accuracy and test set accuracy of our new model are almost the same as the baseline model, which has 0.691645 validation accuracy and 0.688117 test accuracy.

We compare the performance of the models using different values of k. The results for validation data and test data are as followed.





It seems the accuracy is not improved by introducing the weights and regularization term. Probably because our weights do not reflect the prediction power of data points precisely, so optimization is not improved. More factors need to be taken into account for better performance. Also we're punishing the correct answers for outlier questions only, but the wrong answer data point still have value 0. And regularization term does not seem to have an effect on the result because the baseline model is not affected by over-fitting. So even if the regularization do reduce overfitting, we can't see this by comparing results from the two model. From this perspective, since this model is more complicated but have similar result as the base model, we can say the regularization term does prevent over-fitting.

Other than the performance, we do have an improvement on running speed. The regularization term in the cost function helps the convergence process faster. It's using less time than the older version

```
The old version runtime:
0:00:35.425259
```

```
The new version(part b) runtime:
0:00:06.529537
```

4

Limitation 1

When we apply weight improvement to our data, we consider difficulty of the question the only factor affecting the "prediction power" of answers to that question. But this may not

be the case in reality. There are other factors affecting usefulness of data point, for example, users may generate answers that do not reflect their understand of the concept by guessing. **Possible extension**: evaluate the difficulty of each question, if a user gives wrong answer to an easy question but gives correct answer to a hard question, then we punish the second data point by adding a weight to it.

Limitation 2

When we apply weight improvement to our data, we assume the difficulty of each question are independent and follow a normal distribution. But this may not be the case in reality. If several questions are testing on related concepts, then the answers should somehow be related to each other. Also difficulties of the questions may not follow normal distribution. **Possible extension**: we can test more distribution to find the best distribution for our model.

Reference

*M*eira, Dania & Viterbo, Jose & Bernardini, Flavia. (2018). *An Experimental Analysis on Scalable Implementations of the Alternating Least Squares Algorithm.* 351-359.