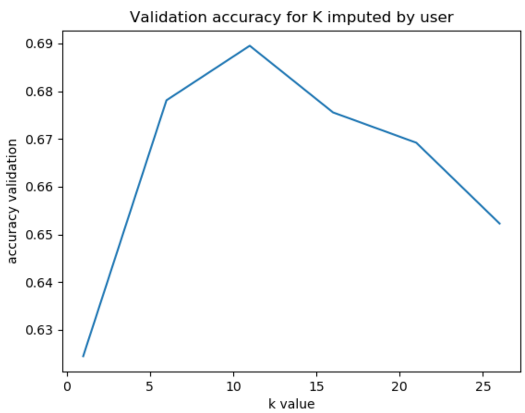
1.

(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:

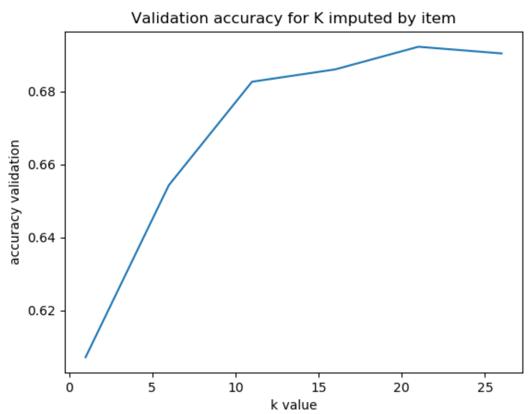
0.6841659610499576 for k = 11

(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:
0.682613604290149
```

```
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 0.6683601467682755

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

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

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

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

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

$$= \sum_{i=1}^{541} \left[- I\,(c_{ij}=1) + I\,(c_{ij}=0 \text{ or } c_{ij}=1) \frac{e^{\theta_i-\beta_j}}{1+e^{\theta_i-\beta_j}} \right]$$

(b) learning rate : 0.01

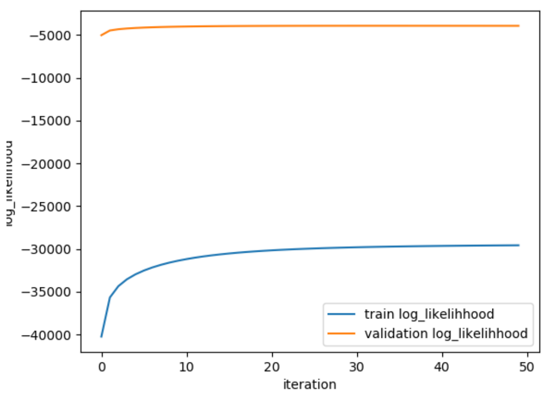
Number of iterations : 50

θ zero vector

β zero vector.

$\Big\}$ hyperparmeters we selected

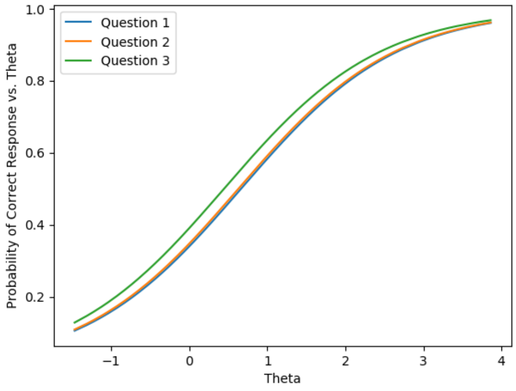
Training curve



(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.

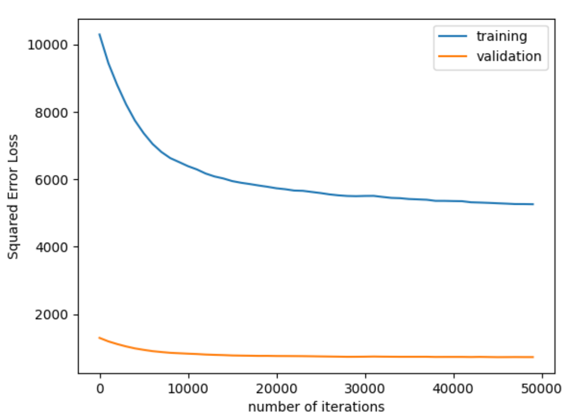
c) see code

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 happening in this setting. So bagging does not improve performance.