دانشگاه صنعتی شریف
دانشکده مهندسی برق

پروژه پایانی درس یادگیری آماری

استاد درس: دکتر محمدزاده

دانشجو: امید جفائی

شماره دانشجویی: ۴۰۱۲۰۴۲۶۸

نیمسال دوم تحصیلی ۰۲ - ۰۱

فاز اول

الگوریتم K-Nearest Neighbors

الف: تابع knn_impute_by_user() همانند تصویر زیر تکمیل شده است.

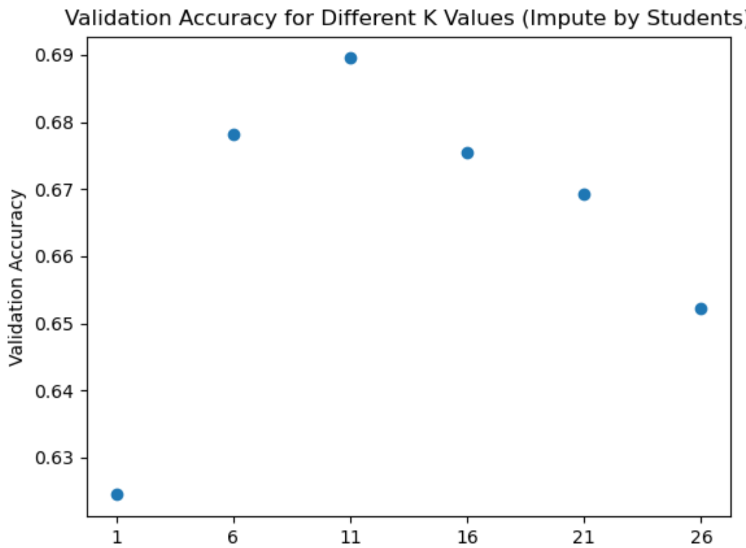
```python
def knn_impute_by_user(matrix, valid_data, k):
    """ Fill in the missing values using k-Nearest Neighbors based on
    student similarity. Return the accuracy on valid_data.

    See https://scikit-learn.org/stable/modules/generated/sklearn.
    impute.KNNImputer.html for details.

    :param matrix: 2D sparse matrix
    :param valid_data: A dictionary {user_id: list, question_id: list,
    is_correct: list}
    :param k: int
    :return: float
    """
    #####################################################################
    # TODO:                                                             #
    # Implement the function as described in the docstring.             #
    #####################################################################
    imputer = KNNImputer(n_neighbors=k)
    completed_mat = imputer.fit_transform(matrix)
    #####################################################################
    #                       END OF YOUR CODE                            #
    #####################################################################
    acc = sparse_matrix_evaluate(valid_data, completed_mat)
    print("Validation Accuracy: {}".format(acc))
    return acc
```

ب: دقت تابع knn_impute_by_user() برای داده های اعتبارسنجی به صورت زیر است:

Validation Accuracy for k=1	0.6244707874682472
Validation Accuracy for k=6	0.6780976573525261
Validation Accuracy for k=11	**0.6895286480383855**
Validation Accuracy for k=16	0.6755574372001129
Validation Accuracy for k=21	0.6692068868190799
Validation Accuracy for k=26	0.6522720858029918



پ: بهترین دقت به ازای k=11 حاصل می شود و مقدار دقت برابر است با:

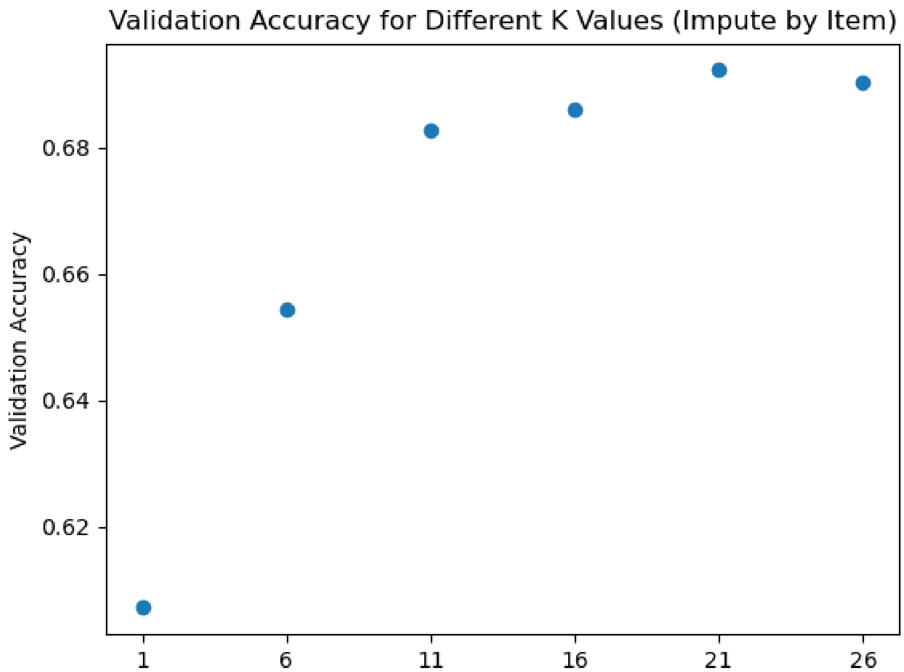
Validation Accuracy for k=11: 0.6895286480383855

ت: در اینجا می‌توان باز هم از تابع قسمت الف استفاده کرد. به این صورت تابع، ماتریس ورودی را Transpose می‌کند. سپس KNNImputer روی آن عمل می‌کند. و در نهایت Transpose آن برای محاسبه دقت به تابع sparse_matrix_evaluate داده می‌شود. فرضیه اصلی در این قسمت به این صورت است که جواب های همه دانش آموزان به یک سوال مشخص در نظر گرفته می‌شود. سپس نزدیک ترین سوال به آن، بر اساس جواب های همه دانش آموزان، پیدا شده و Impute بر اساس آن صورت می‌گیرد.

Validation Accuracy for k=1	0.607112616426757
Validation Accuracy for k=6	0.6542478125882021
Validation Accuracy for k=11	0.6826136042901496
Validation Accuracy for k=16	0.6860005644933672
Validation Accuracy for k=21	**0.6922099915325995**
Validation Accuracy for k=26	0.690375388808919

بهترین دقت به ازای k=21 حاصل می شود و مقدار دقت برابر است با:

Validation Accuracy for k=21: 0.6922099915325995



ث: روش Impute_by_item در k های بزرگتر اندکی بهتر عمل می‌کند.

	Impute_by_item	Impute_by_user
Validation Accuracy for k=1	0.607112616426757	0.6244707874682472
Validation Accuracy for k=6	0.6542478125882021	0.6780976573525261
Validation Accuracy for k=11	0.6826136042901496	**0.6895286480383855**
Validation Accuracy for k=16	0.6860005644933672	0.6755574372001129
Validation Accuracy for k=21	**0.6922099915325995**	0.6692068868190799
Validation Accuracy for k=26	0.69037538808919	0.6522720858029918

ج: در دیتاست های بزرگ محاسبه فاصله ی دو به دو داده ها، وقت گیر خواهد بود. همچنین اگر ابعاد داده ها زیاد باشد، وضعیت بدتر خواهد شد. علاوه بر زمان محاسبه، منابع زیادی (CPU , RAM) را نیز در صورت زیاد بودن تعداد و ابعاد داده ها به کار می‌گیرد. این روش در دیتاست های نویزی خوب عمل نمی‌کند.

خروجی نهایی:

```
-----------------KNN Part B-----------------
Validation Accuracy: 0.6244707874682472
Validation Accuracy: 0.6780976573525261
Validation Accuracy: 0.6895286480383855
Validation Accuracy: 0.6755574372001129
Validation Accuracy: 0.6692068868190799
Validation Accuracy: 0.6522720858029918
-----------------KNN Part C-----------------
The Best K is 11 and Its Accuracy is 0.6895286480383855.
-----------------KNN Part D-----------------
Validation Accuracy: 0.607112616426757
Validation Accuracy: 0.6542478125882021
Validation Accuracy: 0.6826136042901496
Validation Accuracy: 0.6860005644933672
Validation Accuracy: 0.6922099915325995
Validation Accuracy: 0.69037538808919
The Best K is 21 and Its Accuracy is 0.6922099915325995.
```

Item Response Theory (IRT)

الف:

$$\log p(C|\theta,\beta) = \sum_{i=1}^{542} \sum_{j=1}^{1774} \log\left(p(c_{ij}=1|\theta_i,\beta_j)^{c_{ij}} \left(1-p(c_{ij}=1|\theta_i,\beta_j)\right)^{1-c_{ij}} \right)$$

$$= \sum_i \sum_j c_{ij} \log\left(\frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)}\right) + (1-c_{ij}) \log\left(1-\frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)}\right)$$

حال فرض می کنیم $x = \theta_i - \beta_j \;,\; y = \text{sigmoid}(x)$:

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

$$= \left(\sum_j \frac{c_{ij}}{y} - \frac{(1-c_{ij})}{1-y}\right)\left(y(1-y)\right)(1)$$

$$= \sum_{j=1}^{1774} c_{ij} - y = \sum_{j=1}^{1774} c_{ij} - \frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)}$$

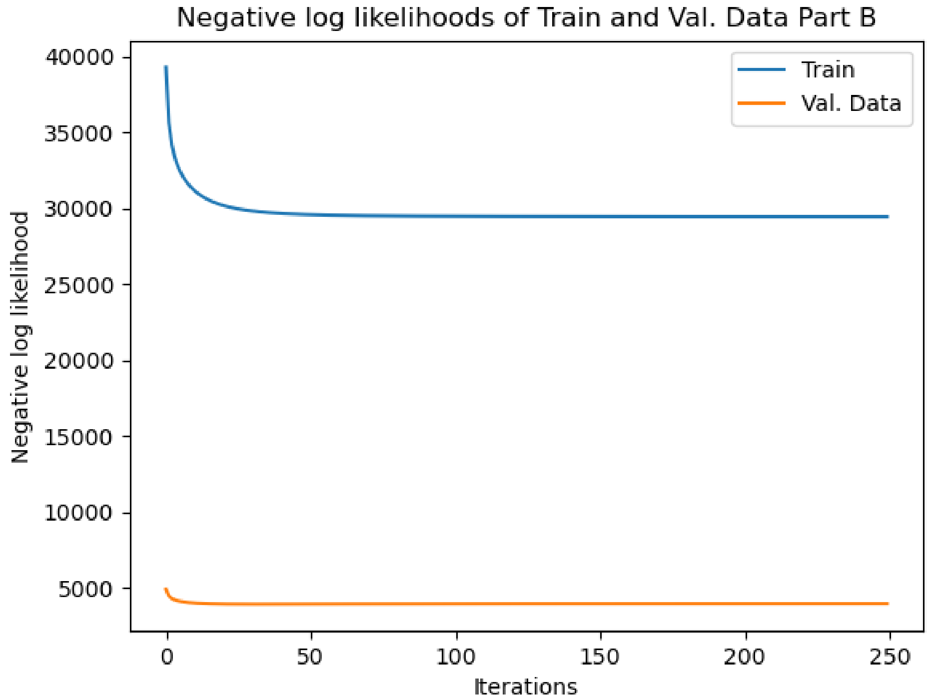
$$\frac{\partial \log p(C|\theta,\beta)}{\partial \beta_j} = \frac{\partial \log p(C|\theta,\beta)}{\partial y} \cdot \frac{\partial y}{\partial x} \cdot \frac{\partial x}{\partial \beta_j}$$

$$= \left(\sum_i \frac{c_{ij}}{y} - \frac{(1-c_{ij})}{1-y}\right)\left(y(1-y)\right)(-1)$$

$$= \sum_{i=1}^{542} y - c_{ij} = \sum_{i=1}^{542} \frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)} - c_{ij}$$

ب: هایپرپارامتر هایی که استفاده شدند، تعداد تکرار (num_iterations) و نرخ یادگیری (lr) بودند.

num_iterations = 250 , lr = 0.01

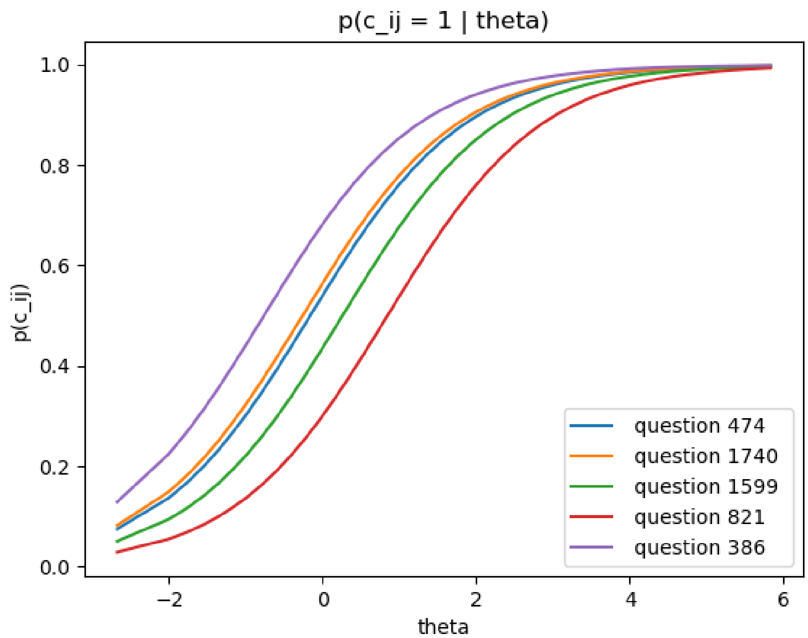


پ: مقادیر خواسته شده به شرح زیر هستند:

The Best Accuracy for Val. Set is **0.7085802991814846**.

Log likelihood is **3935.1620834070377**.

ت: با توجه به اینکه θ نشان دهنده توانایی دانش آموز در پاسخ گویی به سوالات است، انتظار می‌رود هر چه θ بیشتر باشد، احتمال پاسخگویی به سوال به ۱ نزدیکتر شود و برعکس. همچنین در نمودار ترسیم شده، می‌توان احتمال پاسخگویی به سوال مربوطه را با توجه به توانایی دانش آموز در پاسخ به آن، ملاحظه کرد.



خروجی نهایی :

```
-----------------Item Response Part B-----------------
Iterations = 250
Learning Rate = 0.01
-----------------Item Response Part C-----------------
The Best Accuracy for Val. Set is 0.7085802991814846 and
log likelihood is 3935.1620834070377.
```

Matrix Factorization

الف:SVD

Validation Accuracy for k=2	0.42012418854078465
Validation Accuracy for k=3	0.42675698560541914
Validation Accuracy for k=4	0.43141405588484333
Validation Accuracy for k=5	0.4381879762912786
Validation Accuracy for k=6	0.44213942986169913
Validation Accuracy for k=7	**0.4451030200395145**

The Best K for Validation Set is 7 and Its Accuracy is 0.4451030200395145.

ب: یکی از محدودیت ها، این بود که برای SVD نباید مقادیر خالی می‌داشتیم. همچنین در این روش خطای درایه هایی که خالی بودند ولی با صفر پر شدند، یک مشکل به حساب می‌آید. چون ما از مقدار دقیق آن ها خبر نداشتیم ولی صفر گذاشتیم. و این کار باعث کاهش دقت و افزایش خطا می‌شود.

مقادیر خالی باید با صفر پر شوند.

پ: این روش ماتریس $C_{n\times m}$ را به صورت ضرب دو ماتریس $U_{n\times k}$ و $Z_{m\times k}$ تجزیه می‌کند. که k یک هایپرپارامتر است و latent factor نام دارد. این روش جز matrix factorization algorithm است و معمولا برای مسائل recommender systems به کار می‌رود. همچنین این روش وقتی که داده ها sparse هستند عملکرد خوبی دارد.

$C_{n\times m} = U\,Z^T$

رابطه ی آپدیت ها:

```
u[n, :] = u[n, :] + lr * (c - np.dot(u[n, :], z[m, :]) )* z[m, :]
z[m, :] = z[m, :] + lr * (c - np.dot(u[n, :], z[m, :]) )* u[n, :]
```

ت:

Learning Rate = 0.01 , Iterations = 200

Validation Accuracy for k=2	**0 .69616144510302**
Validation Accuracy for k=3	0 .6875529212531752
Validation Accuracy for k=4	0.6780976573525261
Validation Accuracy for k=5	0.6652554332486593
Validation Accuracy for k=6	0.6666666666666666

The Best K for Validation Set is k=2 and Its Accuracy is 0.69616144510302**.**

```
Final U Matrix is [[ 1.99626771e-01  7.41142127e-01]
 [ 1.81502846e-01  1.14001903e+00]
 [ 5.84018481e-04  1.24201201e+00]
 ...
 [ 4.71259706e-01  1.68225759e-01]
 [ 7.92528952e-01  7.84872888e-01]
 [-4.94158888e-01  1.05800321e+00]]
Final Z Matrix is [[0.80895587 0.29164358]
 [0.3313012  0.56416134]
 [0.30084385 0.67017247]
 ...
 [0.6190033  0.3728025 ]
 [0.72367598 0.6271724 ]
 [0.56381893 0.41234505]]
Final Matrix is [[ 0.37763859  0.48426033  0.55674954 ...  0.39986927  0.60928899
   0.41815964]
 [ 0.47930702  0.70328678  0.81861339 ...  0.5373528   0.84633772
   0.57241594]
 [ 0.36269727  0.70088865  0.83253796 ...  0.46338669  0.77937829
   0.51246678]
 ...
 [ 0.43029027  0.25103538  0.25451586 ...  0.3544263   0.44654588
   0.3350722 ]
 [ 0.87002409  0.70536074  0.76442767 ...  0.78318061  1.06578478
   0.77048127]
 [-0.09119289  0.43316907  0.56037996 ...  0.08854025  0.30593949
   0.15764625]]
```

ث:

بهترین دقت (دقت نهایی) 0.69616144510302 است که به ازای k=2 حاصل می‌شود.

ج: تابع هزینه باید شبیه تابع هزینه logistic regression شود.

$$\cos t = -\sum_{(n,m)\in O} \log(sigmoid(C_{nm} - u_n{}^T z_m)^{C_{nm}} (1 - sigmoid(C_{nm} - u_n{}^T z_m)^{1-C_{nm}}))$$

*توجه شود که در این بخش مقادیری که تابع als برمی‌گردانند تغییر کرده است (اضافه شده است).

*همچنین اجرای این قسمت زمانبر است (بین ۱ ساعت تا ۴۵ دقیقه).

Ensemble

در این قسمت از سه knn_impute_by_user() به عنوان base learner استفاده شد. و در هرکدام از یکی از مقادیر k=6,11,26 به عنوان k استفاده شد. با توجه به bootstrap داده ها، هر base learner نتیجه ی بدتری نسبت به حالت بدون base learner داشت (قسمت اول فاز اول).

k	knn_impute_by_user (without bootstrap)	knn_impute_by_user (with bootstrap)
6	0.6780976573525261	0.5946937623482924
11	0.6895286480383855	0.5956816257408976
26	0.6522720858029918	0.6062658763759525

میانگین ماتریس خروجی base learner ها، خروجی نهایی مدل در نظر گرفته شد. و سپس از voting برای تعیین نهایی درایه ها استفاده شد. به این صورت اگر هر درایه‌ای از ۰.۵ بزرگتر باشد ۱ و در غیر اینصورت ۰ مقداردهی می‌شود.

```
final_output = np.mean( np.array([method1_output, method2_output, method3_output]), axis=0 )

for i in range(min(final_output.shape)):
    for j in range(max(final_output.shape)):
        if final_output[i][j] > 0.5:
            final_output[i][j] = 1
```

دقت مدل نهایی از هر base learner بیشتر بود. ولی کماکان نسبت به سایر قسمت های فاز اول نتیجه خوب نبود.

Final Accuracy of Ensemble Approach is 0.6117696867061811.

با توجه به bootstrap کردن داده ها، طبیعی است که نتیجه ی هر base learner از قسمت اول فاز اول کمتر شود. ولی در مجموع چون برای روش های قبلی داده ی کافی در اختیار داشتیم، از ensemble چندان تغییری در نتایج انتظار نمی‌رفت.

```
Validation Accuracy k=6: 0.5946937623482924
Validation Accuracy k=11: 0.5956816257408976
Validation Accuracy k=26: 0.6062658763759525
Final Accuracy of Ensemble Approach is
0.6117696867061811.
```

فاز دوم

در این قسمت، با اضافه کردن دو پارامتر، تابع قسمت IRT بهبود داده شد.

۱-توضیحات:

در واقع می‌توان پارامتری را برای هر سوال در نظر گرفت (a_j) که نشان دهد، افراد با توانایی (دانش) پایین، در واقع شانس بسیار کمتری برای پاسخ صحیح نسبت به افراد با توانایی بالاتر دارند. همچنین این پارامتر نشان دهنده میزان تیزی (sharpness) تابع p برحسب θ خواهد بود.

پارامتر بعدی γ را می‌توان به این شکل در نظر گرفت که نشان دهد در صورتی که دانش آموزی به صورت شانسی (در واقع با دانش کم) به سوالی جواب دهد، چقدر جواب او درست خواهد بود.

تابع جدید IRT به صورت زیر خواهد بود:

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

حال مجددا به محاسبه log-likelihood و مشتق آن نسبت به پارامتر های موجود می‌پردازیم:

لازم به ذکر است که γ را یک عدد ثابت در نظر می‌گیریم و نیاز به رابطه آپدیتی نیست.

$$\ell = \sum_{i=1}^{۵۴۲} \sum_{j=1}^{۱۷۷۴} c_{ij} \lg\left(\gamma + (1-\gamma)\frac{\exp(a_j(\theta_i - \beta_j))}{1+\exp(a_j(\theta_i-\beta_j))}\right)$$

$$+ (1-c_{ij})\left((1-\gamma)\left(1 - \frac{\exp(a_j(\theta_i-\beta_j))}{1+\exp(a_j(\theta_i-\beta_j))}\right)\right)$$

حال فرض می‌کنیم :

$$x = a_j(\theta_i - \beta_j) \qquad , \quad y = \text{sigmoid}(x)$$

$$\ell = \sum_i \sum_j c_{ij} \lg(\gamma + (1-\gamma)y) + (1-c_{ij})\big((1-\gamma)(1-y)\big)$$

$$\frac{\partial \ell}{\partial \theta_i} = \sum_j \frac{\partial \ell}{\partial y} \cdot \frac{\partial y}{\partial x} \cdot \frac{\partial x}{\partial \theta_i}$$

$$\frac{\partial \ell}{\partial y} = c_{ij} \frac{1-\gamma}{\gamma + (1-\gamma)y} - (1-c_{ij}) \frac{1}{1-y}$$

$$\frac{\partial y}{\partial x} = y(1-y) \qquad , \quad \frac{\partial x}{\partial \theta_i} = a_j$$

$$\frac{\partial \ell}{\partial \theta_i} = \sum_j \left(c_{ij} \frac{1-\gamma}{\gamma+(1-\gamma)y} - (1-c_{ij}) \frac{1}{1-y}\right)(y(1-y))\,(a_j)$$

$$\frac{\partial \ell}{\partial \beta_j} = \sum_i \frac{\partial \ell}{\partial y} \cdot \frac{\partial y}{\partial x} \cdot \frac{\partial x}{\partial \beta_j} = \sum_i \left(c_{ij} \frac{1-\gamma}{\gamma+(1-\gamma)y} - (1-c_{ij}) \frac{1}{1-y}\right)(y(1-y))(-a_j)$$

$$\frac{\partial \ell}{\partial a_j} = \sum_i \frac{\partial \ell}{\partial y} \cdot \frac{\partial y}{\partial x} \cdot \frac{\partial x}{\partial a_j} = \sum_i \left(c_{ij} \frac{1-\gamma}{\gamma+(1-\gamma)y} - (1-c_{ij}) \frac{1}{1-y}\right)(y(1-y))(\theta_i - \beta_j)$$

با توجه به روابط فوق، تابع آپدیت به صورت زیر درمی‌آید:

```python
def update_theta_beta(data, lr, theta, beta, a, gamma):
    """
    :param data: A dictionary {user_id: list, question_id: list,
    is_correct: list}
    :param lr: float
    :param theta: Vector
    :param beta: Vector
    :param a: Vector
    :param gamma: float
    :return: tuple of vectors
    """
    #####################################################################
    # TODO:                                                             #
    # Implement the function as described in the docstring.            #
    #####################################################################
    for n, j in enumerate(data["question_id"]):
        i = data["user_id"][n]
        c = data["is_correct"][n]
        x = a[j] * (theta[i] - beta[j])
        y = sigmoid(x)
        p1 = (c*(1-gamma))/(gamma+(1-gamma)*y) - ((1-c)/(1-y))
        p2 = y*(1-y)
        theta[i] = theta[i] + lr * (p1 * p2 * (a[j]))
        beta[j] = beta[j] + lr * (p1 * p2 * (-a[j]))
        a[j] = a[j] + lr * (p1 * p2 * (theta[i] - beta[j]))
```

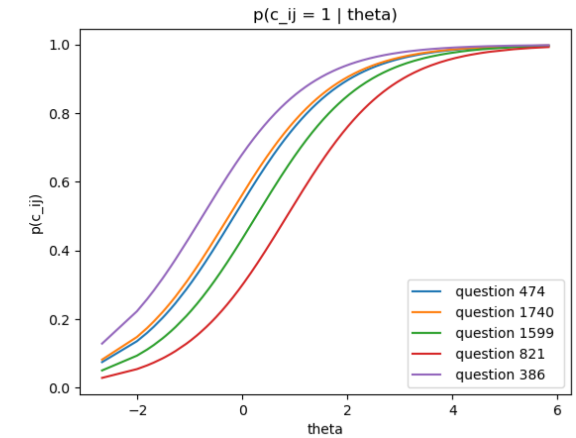
همچنین مقادیر اولیه این پارامتر ها را پس از run های متعدد به صورت زیر در نظر می‌گیریم. این مقادیر پس از run های متعدد بدست آمده‌اند:

```python
theta = 0.5 * np.ones(542)
beta = 0.5 * np.ones(1774)
a = 0.67 * np.ones(1774)
gamma = 0
```

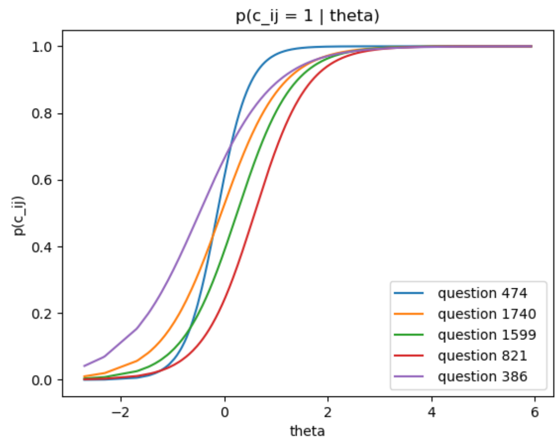
۲-شکل ها و نمودارها:

همچنین پس از چندین بار اجرا، $\gamma = 0$ به دقت بالاتری ختم می‌شود.

نمودار ۵ سوال دلخواه در این قسمت در مقایسه با قسمت قبل را در زیر مشاهده می‌فرمایید:

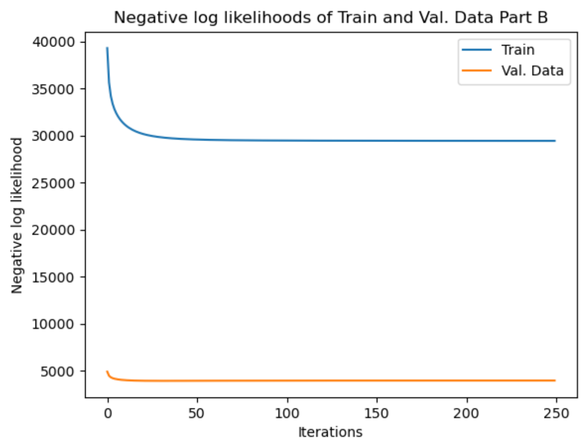


فاز اول

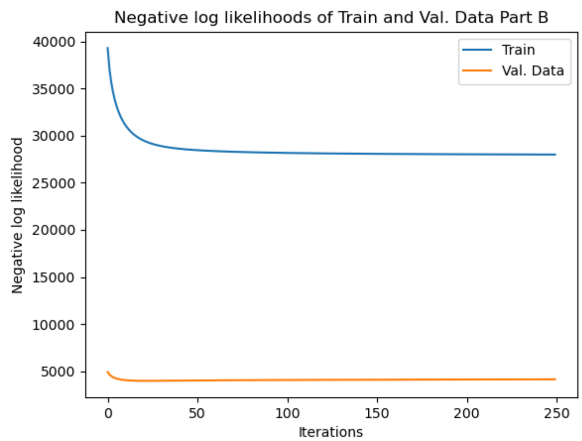


فاز دوم

همچنین نمودار log-likelihood فاز اول و دوم به صورت زیر است:



فاز اول



فاز دوم

۳-مقایسه:

دقت فاز اول و فاز دوم به شرح زیر است:

دقت داده های اعتبارسنجی فاز دوم	دقت داده های اعتبارسنجی فاز اول
0.709991532599492	0.7085802991814846

```
-----------------Modified IRT 2nd Phase-----------------
Iterations = 250
Learning Rate = 0.01
The Best Accuracy for Val. Set is 0.709991532599492 and log likelihood is 3977.9598174564167.
```

همچنین مقایسه دقت تمامی روش های فاز اول و دوم روی مجموعه ی اعتبارسنجی به شرح زیر است:

دقت داده های اعتبارسنجی	روش	
0.6895286480383855	knn_impute_by_user	فاز اول
0.6922099915325995	knn_impute_by_item	
0.7085802991814846	IRT	
0.69616144510302	MF	
0.6117696867061811	Ensemble	
0.709991532599492	Modified IRT	فاز دوم

همانطور که مشاهده می‌شود، دقت مدل فاز دوم اندکی بهبود یافته است.

۴-محدودیت ها:

یکی از محدودیت ها زمانی ایجاد می‌شود که جواب به سوالات و جواب دانش آموزان به سوالات، با هم تفاوت زیادی نداشته نباشد. مثلا تعداد سوالاتی که جواب به آن ها مشابه است یا تعداد دانش آموزانی که جواب های آنان مشابه است زیاد باشد، مدل IRT خوب عمل نمی‌کند. همچنین زمانی که تعداد سوالات و دانش آموزان زیاد با هم تفاوت داشته باشد (مثلا بیش از ۱۰ برابر)، مدل دچار مشکل می‌شود.

در این موارد استفاده از شبکه های عصبی به جواب های بهتری منجر می‌شود.

توجه: فایل test.py این قسمت را اجرا نمی‌کند و فقط فاز اول را اجرا می‌کند.