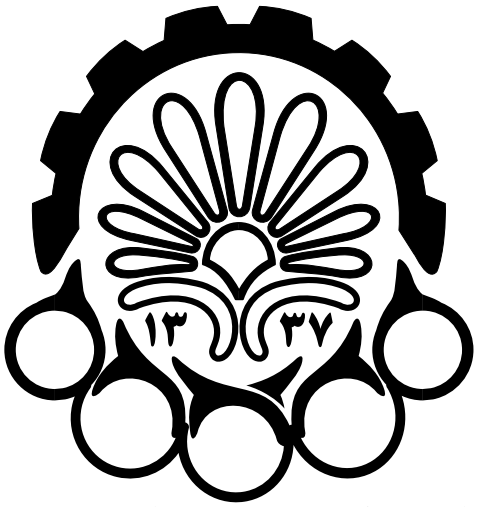


دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

درس : کلان داده

نام و نام خانوادگی : فاطمه توکلی

۴۰۰۱۳۱۰۱۶

تمرین شماره ۰۳

بخش اول **Data stream**

.۱

Adaptive random forest (ARF) توسعه‌ایd از الگو ریتم random forest اصلی است که برای مقابله با جریان‌های داده در حال تکامل طراحی شده‌است. الگوریتم تطبیقی تکنیک‌های Random Forest را با روش‌های مورد استفاده برای مقابله پویا با انواع مختلفی از concept drifts ترکیب میکند. Random forest استاندارد یک مجموعه دسته‌ای است که با استفاده از تمام مجموعه داده‌های استاتیک موجود آموزش داده شده است. بنابراین، RF برای یاد گیری از جریان‌های متوالی داده‌هایی که میتوانند به طور مداوم وارد شوند، مناسب نیست.

برای اینکه الگوریتم RF در حالت آنلاین عمل کند، برخی از سازگاری‌ها مورد نیاز است. یکی از انطباق‌ها این است که یادگیرندگان پایه در ARF، درخت‌های Hoeffding هستند که به جای درخت‌های تصمیم استاندارد مورد استفاده در RF، قادر به یادگیر ی از جریان‌های داده عظیم هستند.

در RF اصلی، دو روش Bagging و انتخاب تصادفی ویژگی‌ها، برای کاهش همبستگی بین مدل‌های پایه و افزایش تنوع مجموعه استفاده میشود. به همین شکل، ARF از طریق انتخاب زیرمجموعه تصادفی از ویژگی‌ها برای تقسیم گره‌ها تنوع را میافزاید.

در یاد گیری آنلاین، استفاده از روش غیر جریانی Bagging برای ترسیم نمونه‌های تصادفی با جایگزینی از داد ه‌های آموزشی اصلی غیرممکن است. دلیل آن این است که این روش به چندین pass روی داده نیاز دارد که اندازه آن ناشناخته است و دائماً در حال افزایش است. با این حال، ARF شامل یک الگوریتم نمونه‌گیری مجدد مؤثر است که بر اساس الگوریتم bagging آنلاین، که در آن با ورود هر نمونه آموزشی جدید و برای هر مدل پایه در مجموعه، از نمونه فعلی استفاده میشود که W بار پشت سر هم مدل پایه را آموزش ببیند. این به معنی وزن دادن به نمونه با یک مقدار است که در آن w یک عدد تصادفی است که توسط توزیع پواسون($\lambda=1$) تولید میشود. در واقع w وزن هر نمونه ورودی است.

در نتیجه، این محدوده وزنی متفاوتی را به نمونه ها نسبت میدهد و بنابراین تنوع فضای ورودی را در داخل مجموعه افزایش می دهد. یکی از چالش‌های رایج در یادگیری آنلاین که در آن داده‌ها در طول زمان جمع‌آوری میشوند، این است که concept داده ممکن است به طور غیرقابل پیش‌بینی drift شود که بر عملکرد مدل پیش‌بینی تأثیر منفی میگذارد و آن را در طول زمان منسوخ میکند. با این حال، هدف ARF شامل مکانیسم‌هایی برای مقابله با انواع مختلف concept drift است. به طور مشخص ARF، از یک آشکارساز drift برای هر درخت در مجموعه برای نظارت بر هشدارها استفاده میکند. به محض شناسایی هشدار در یک درخت، الگوریتم یک درخت پس زمینه ایجاد می‌کند و آموزش آن را همراه با گروه شروع می‌کند. درخت پس‌زمینه را میتوان بعداً برای پیش‌بینی‌ها به‌جای درخت فعال در صورتی که اخطار به یک drift افزایش داده شد، استفاده کرد. این استراتژی از رویکرد پیش‌فرض متفاوت است که درخت‌های پایه را بلافاصله پس از تشخیص drift بازنشانی میکند.

پس دو تغییر داشتیم یکی آموزش کل درخت‌ها به ازای هر داده ورودی و یکی استفاده از یک اخطار برای مشاهده تغییر concept داده‌ها

۱. الگوریتم با مقداردهی اولیه درخت شروع می شود. سپس شروع به دریافت جریان‌های داده و ارسال هر نمونه جدید به هر درخت در مجموعه می‌کند.

۲. ARF به صورت test-then-train setting کار میکند، جایی که نمونه جدید ابتدا برای آزمایش مدل، پیش بینی و تخمین عملکرد آن استفاده می‌شود. سپس از این نمونه برای آموزش مدل استفاده می‌شود. این بدان معناست که یادگیرنده همیشه بر روی داده‌هایی که هنوز ندیده است آزمایش می شود.

۳. تابع آموزش درخت اساساً بر اساس bagging آنلاین و انتخاب یک و یژگی تقسیم از زیر مجموعه تصادفی ویژگی های اندازه M ساخته میشود.

٤. سپس الگوریتم از یک آستانه مجاز برای تشخیص هشدارها در درخت استفاده می‌کند و در صورت شناسایی هشدار، درخت پس‌زمینه را ایجاد می‌کند.

٥. بعداً، اگر یک drift با استفاده از آستانه drift تشخیص داده شود، درخت اصلی با درخت پس‌زمینه جایگزین می‌شود.

٦. در نهایت، ARF تمام درختان پس‌زمینه را در نمونه فعلی آموزش می‌دهد.

Algorithm 2 Adaptive random forests. **Symbols:** m: maximum features evaluated per split; n: total number of trees ($n = |T|$); δ_w: warning threshold; δ_d: drift threshold; $c(\cdot)$: change detection method; S: Data stream; B: Set of background trees; $W(t)$: Tree t weight; $P(\cdot)$: Learning performance estimation function.

```
1: function ADAPTIVERANDOMFORESTS(m, n, δ_w, δ_d)
2:     T ← CreateTrees(n)
3:     W ← InitWeights(n)
4:     B ← ∅
5:     while HasNext(S) do
6:         (x, y) ← next(S)
7:         for all t ∈ T do
8:             ŷ ← predict(t, x)
9:             W(t) ← P(W(t), ŷ, y)
10:            RFTreeTrain(m, t, x, y)          ▷ Train t on the current instance (x, y)
11:            if C(δ_w, t, x, y) then          ▷ Warning detected?
12:                b ← CreateTree()             ▷ Init background tree
13:                B(t) ← b
14:            end if
15:            if C(δ_d, t, x, y) then          ▷ Drift detected?
16:                t ← B(t)                     ▷ Replace t by its background tree
17:            end if
18:        end for
19:        for all b ∈ B do                     ▷ Train each background tree
20:            RFTreeTrain(m, b, x, y)
21:        end for
22:    end while
23: end function
```

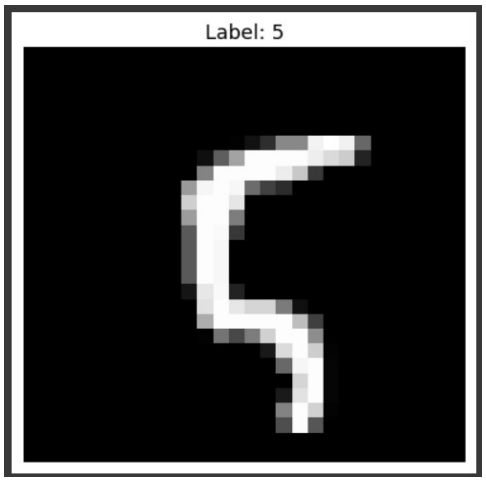
شبه کد الگوریتم ARFS

References:

https://www.diva-portal.org/smash/get/diva2:1473045/FULLTEXT01.pdf

https://link.springer.com/article/10.1007/s10994-017-5642-8

با استفاده از keras.datasets مجموعه داده خواسته شده را import کرده و در نهایت یکی از نمونه های آن را به نمایش گذاشته:



سپس داده‌ها را نرمالایز کرده و تغییر ابعاد می‌دهیم زیرا در ادامه به صورت vector به آن ها نیاز داریم :

```
# Normalize the image of the dataset
x_train = (x_train / 255) - 0.5
x_test = (x_test / 255) - 0.5

# Reshape the input data
x_train = x_train.reshape(x_train.shape[0], -1)
x_test = x_test.reshape(x_test.shape[0], -1)
```

.۳

از هرکدام از مجموعه‌داده آموزش و آزمون به ترتیب ۲۰۰۰ و ۱۰۰۰ نمونه به صورت تصادفی انتخاب کرده و با استفاده از آن‌ها جریان‌داده خود را میسازیم:

```
# Select 2000 random samples from the training set
train_samples = 2000
random_indices = np.random.choice(len(x_train), size=train_samples, replace=False)
x_train_subset = x_train[random_indices]
y_train_subset = y_train[random_indices]

# Select 1000 random samples from the test set
test_samples = 1000
random_indices = np.random.choice(len(x_test), size=test_samples, replace=False)
x_test_subset = x_test[random_indices]
y_test_subset = y_test[random_indices]

# Create DataStream Object for training subset
stream_train = DataStream(x_train_subset, y_train_subset)

# Create DataStream Object for test subset
stream_test = DataStream(x_test_subset, y_test_subset)
```

۴. ابتدا دسته‌بند adpativerandomForest را تعریف کرده:

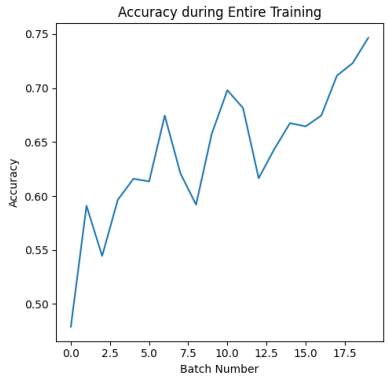
```
# Initialize and Train the Adaptive Random Forest
arf = AdaptiveRandomForestClassifier(n_estimators=10)
```

در اینجا میتوان پارامترهایی مانند n_estimator که تعداد درخت ها در ensemble هستند یا max_featuresint حداکثر تعداد ویژگی برای هر تقسیم گره است یا مقدار λ در توزیع پواسون و .. را مشخص کرد.

سپس ۱۰۰ تا ۱۰۰ تا به صورت دسته ایی و همینطور incremental داده های آموزشی را به دسته بند میدهیم:

```
for i in range(0, n_samples_train, batch_size):
    X_batch, y_batch = stream_train.next_sample(min(batch_size, n_samples_train - i))
    arf.partial_fit(X_batch, y_batch, classes=np.unique(y_train))
```

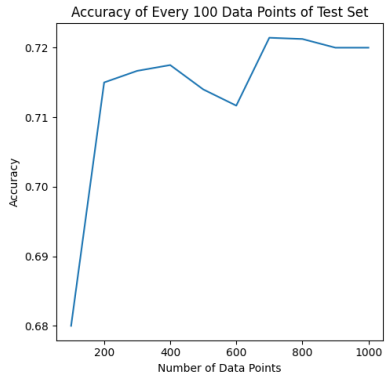
دقت آن روی ۲۰۰۰ داده آموزشی و با استفاده از ۱۰ درخت جایگزین برای آموزشی به صورت زیر میباشد:



۵.

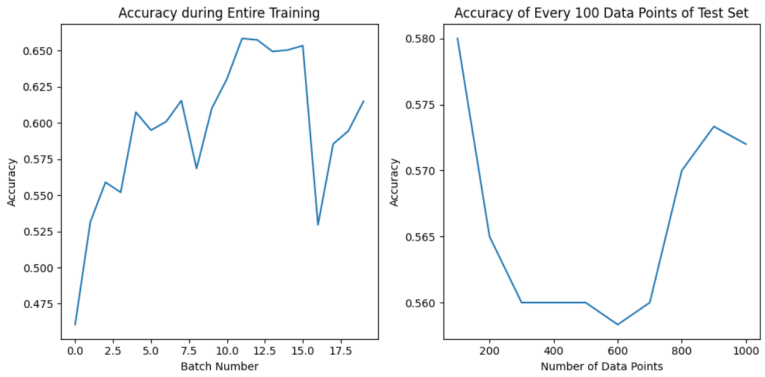
با استفاده از ۱۰۰۰ داده آزمون که به صورت رندوم انتخاب شده بودند دقت مدل به صورت زیر می باشد:

```
Accuracy on Test Set: 0.724
```



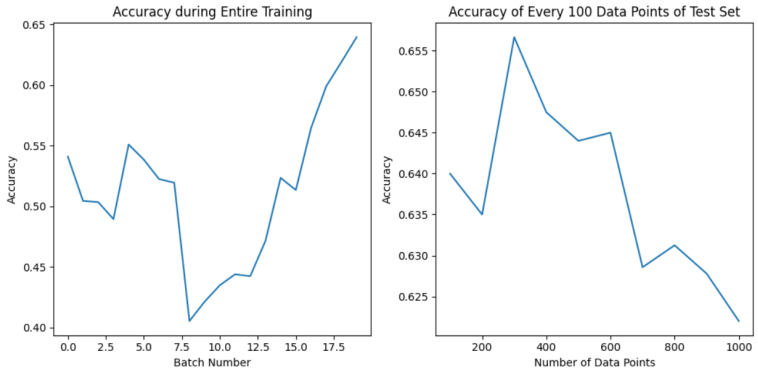
n_estimator = 5

```
Accuracy on Test Set: 0.572
```



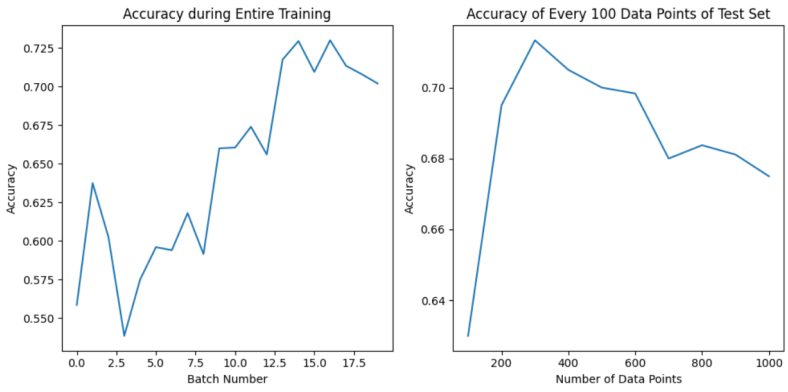
n_estimators=10 , max_features = 10

```
Accuracy on Test Set: 0.622
```



n_estimators=5 , max_features = 50

```
Accuracy on Test Set: 0.675
```



بخش دوم **Singular Value Decomposition**

الف)

$$Y = Mx$$

$$C = CoV(Y) = (Y-\bar{Y})(Y-Y)^T = (Mx - \bar{M}x)(Mx - \bar{M}x)^T \xrightarrow{M \text{ فاکتورگیری}} = M \underbrace{(x-\bar{x})(x-\bar{x})^T}_{Covariance(x) = I} M^T$$

$$\Rightarrow C = MIM^T = MM^T$$

$$C = U_m \Sigma_m V_m^T V_m \Sigma_m^T U_m^T = U_m \Sigma_m \Sigma_m^T U_m^T$$

$$C = U_m \Sigma_m^2 U_M^T \quad , \quad C = U_c \Sigma_c V_c^T$$

$$\Rightarrow U_m = U_c \quad , \quad V_c = V_m \quad , \quad \Sigma_c = \Sigma_m^2$$

ب)

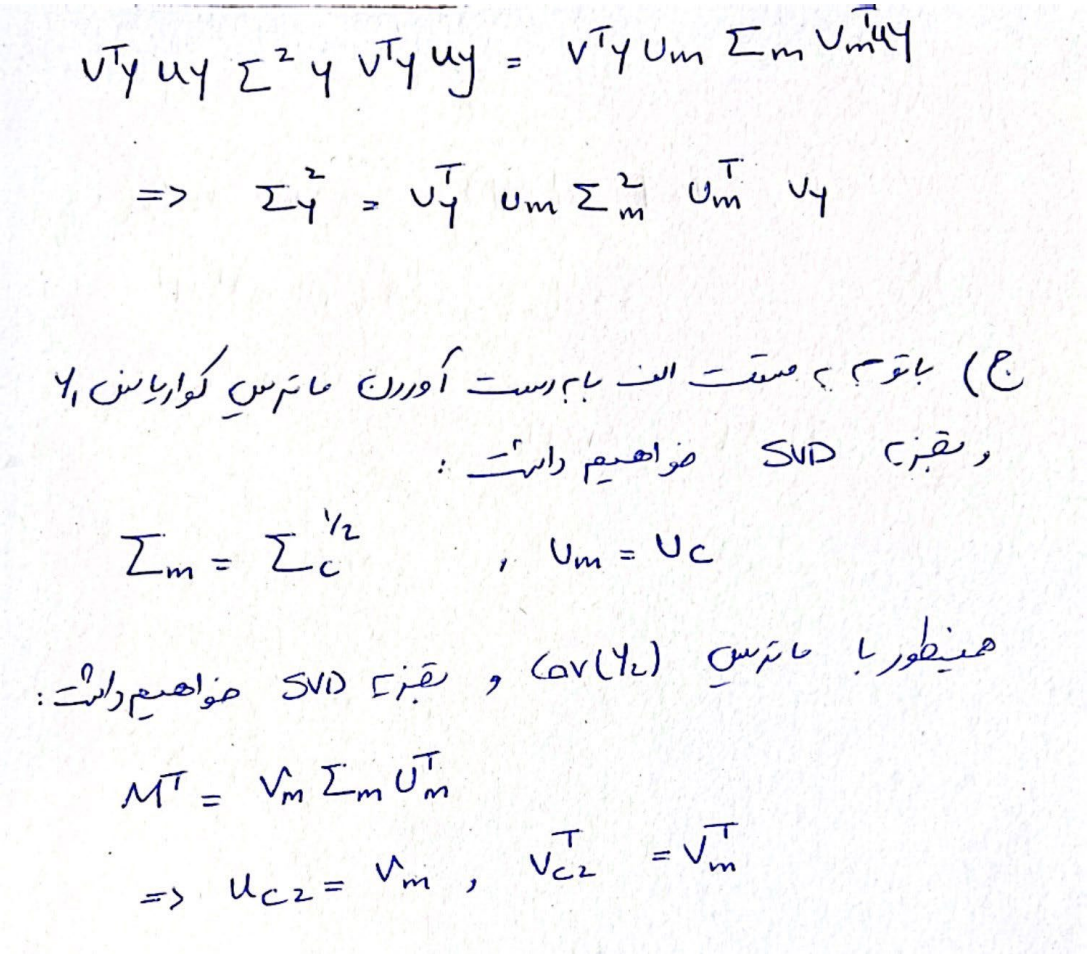
$$CoV(y) = U_m \Sigma_m^2 V_m^T$$

$$(y-\bar{y})(y-\bar{y})^T = U_m \Sigma_m^2 V_m^T \quad , \quad E(y) = E(Mx) = M E(x) = 0$$

$$\Rightarrow YY^T = U_m \Sigma_m^2 U_m^T$$

$$U_y \Sigma_y V_y^T V_y \Sigma_y U_y^T = U_m \Sigma_m^2 U_m^T$$

$$U_y \Sigma_y^2 V_y^T = U_m \Sigma_m^2 V_m^T$$



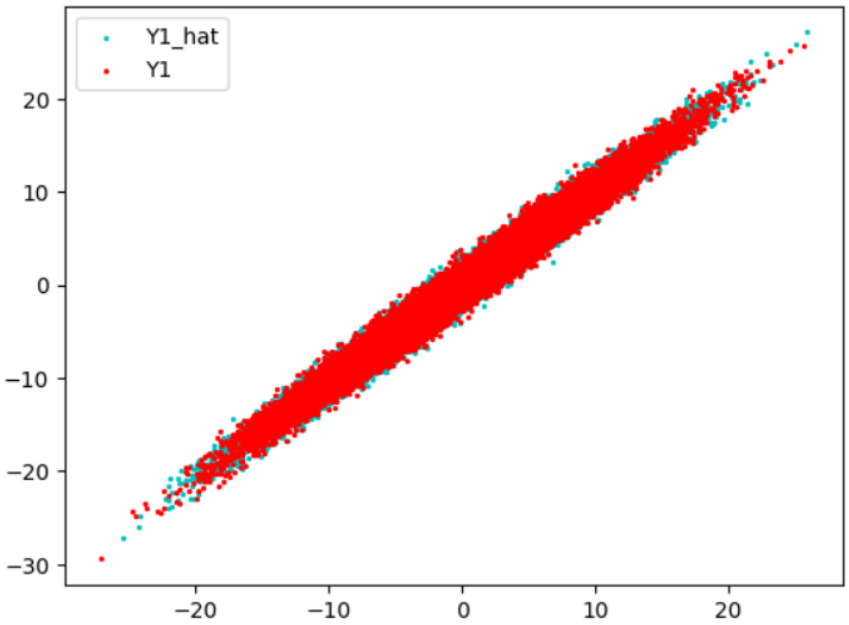
ماتریس M و M^T **به** دست آمده از داده‌ها به صورت زیر می‌باشد:

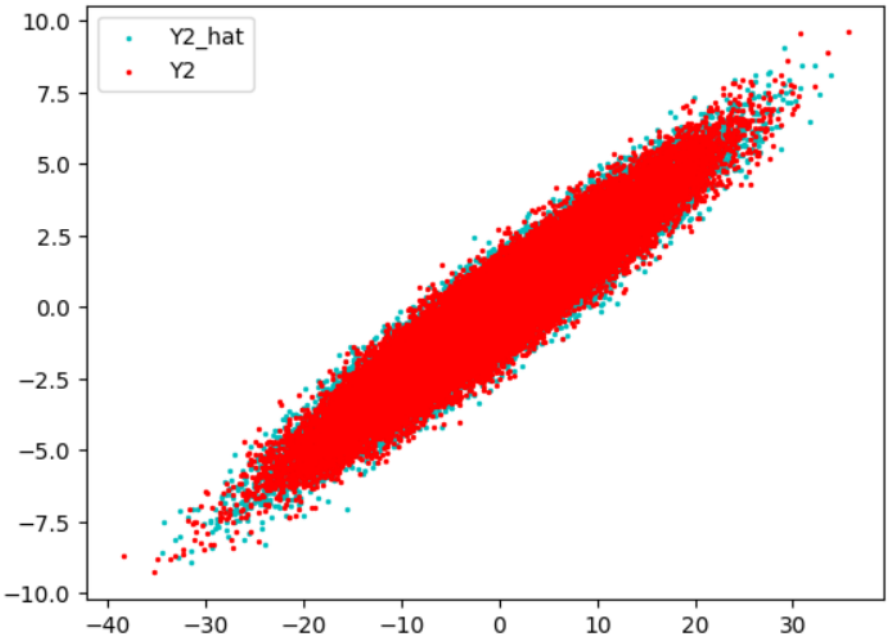
```
M equal to :
 [[6.01864836 1.00359787]
 [6.02101203 1.99876244]]
```

```
M^T equal to :
 [[6.0043988  6.00559706]
 [0.99876775 1.99609828]]
```

د.

با استفاده از ماتریس M و M^T به دست‌آمده در قسمت قبل و نمونه‌برداری از توزیع گوسی با میانگین صفر و ماترس کواریانس واحد خواهیم داشت و ضرب نقطه آن‌ها خواهیم داشت:





همانطور که مشاهده می‌شود نقاط روی هم منطبق هستند بنابراین ماتریس M و M^T به دست‌آمده درست است.

بخش سوم **Recommender System**

الف)

$$E = \Big(\sum_{(u,i)\in training} \log\left(sigmoid(r_{ui} - q_i.p_u^T)\right) \Big) + \underbrace{\lambda\Big(\sum_i \|q_i\|_2^2 \sum_u \|p_u\|_2^2\Big)}_{\text{مشتق نسبت به } r_{ui} \text{ صفر است}}$$

$$\varepsilon_{ui} = \frac{sigmoid(r_{ui} - q_i p_u^T).(1 - sigmoid(r_{ui} - q_i p_u))}{sigmoid(r_{ui} - q_i p_u^T)}$$

ب)

نتایج به ازای نرخ یادگیری متفاوت به صورت زیر است :

```
Predicted rating for item 0 and user 0: 0.8862937470958735
Accuracy for model with learning rate 0.001: 0.0014215686274509803

Predicted rating for item 0 and user 0: 0.21461710624541544
Accuracy for model with learning rate 0.1: 0.000588235294117647

Predicted rating for item 0 and user 0: 0.6851547681666874
Accuracy for model with learning rate 0.01: 0.001
```

به جز روند گرادیان گرفتن و به روز رسانی متغیرها مجموعه داده شده را به دو قسمت آموزش و آزمون جدا کرده و سپس ۲۰ درصد از مجموعه آزمون را به صورت رندوم صفرکرده :

```
Training set:
[[0 1 1 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
Test set:
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

```
1  test_set.shape
```

```
(11943, 40981)
```

```
1  train_set.shape
```

```
(17915, 40981)
```

و همچنین برای ارزیابی به این صورت عمل شده که کل مجموعه آزمون در نظر گرفتیم و برای هر کاربر predict_rating کردیم و سپس آن ها مرتب کرده ۱۰ تای برتر را انتخاب کرده و با برچسب حقیقی آن ها مقایسه کرده ایم و نتایج به صورتی که مشاهده شد در آمده است.