KHOA HỌC DỮ LIỆU

BÁO CÁO ĐỒ ÁN CUỐI KỲ

Đề tài: Xây dựng hệ thống gợi ý phim

Nhóm thực hiện:

Võ Quốc Cường – 1412071

Tiêu Thanh Sơn – 1412467

Giảng viên:

Trần Trung Kiên

I. Giới thiệu bài toán:

- Gợi ý phim cho người dùng dựa trên dữ liệu thu thập được trên imdb.com
- Dữ liệu training:
  - Movies: dữ liệu chứa thông tin của từng bộ phim như: tiêu đề, năm phát hành, danh sách thể loại, danh sách diễn viên, thời lượng phim (dài tập hay 1 tập)
  - Ratings: dữ liệu rating của user cho các bộ phim
- Ứng dụng:
  - Tặng lượt view của các bộ phim => tăng doanh thu của trang web.
  - Giúp trang web có thể thu hút và giữ người dùng hiệu quả hơn.

II. Thu thập dữ liệu:

1. Dữ liệu movies:

- **Dựa trên Advance Search của imdb.com dạng:** http://www.imdb.com/search/title?release_date=[%year%]&page=[%page%]

movies.csv

```
1  ,movie_id,title,genres
2  0,1,Toy Story (1995),Animation|Children's|Comedy
3  1,2,Jumanji (1995),Adventure|Children's|Fantasy
4  2,3,Grumpier Old Men (1995),Comedy|Romance
5  3,4,Waiting to Exhale (1995),Comedy|Drama
6  4,5,Father of the Bride Part II (1995),Comedy
7  5,6,Heat (1995),Action|Crime|Thriller
8  6,7,Sabrina (1995),Comedy|Romance
9  7,8,Tom and Huck (1995),Adventure|Children's
10 8,9,Sudden Death (1995),Action
11 9,10,GoldenEye (1995),Action|Adventure|Thriller
```

2. Dữ liệu ratings:

- Ở mỗi bộ phim, ta sẽ thu thập các rating, sau đó tổng hợp lại.
- Dữ liệu rating gồm các cột: user_id, movie_id, rating

ratings.csv

```
1  ,user_id,movie_id,rating
2  0,1,1177,5
3  1,1,656,3
4  2,1,903,3
5  3,1,3340,4
6  4,1,2287,5
```

III. Tiền xử lý dữ liệu:

1. Dữ liệu *movies*:

- Chuyển đổi *movie_id* thành dạng số thứ tự.
- Đối với những trường dữ liệu rời rạc ta chuyển về dạng *one hot*: mỗi giá trị rời rạc sẽ là một cột nhị phân [0, 1].

- Ví dụ, ta có dữ liệu cột *genre* như sau:

Genre
Advanture|Comedy|Animation
Children|Animation|Action|Comedy
Fantasy|Animation
Children|Advanture|

- Thì dạng one hot sẽ như sau:

Advanture	Comedy	Animation	Children	Action	Fantasy
1	1	1	0	0	0
0	1	1	1	1	1
0	0	1	0	0	1
1	0	0	1	0	0

- Trong dữ liệu thu thập không có trường dữ liệu *numeric* nên ta không cần chuẩn hóa.

2. Dữ liệu *ratings*:

- Chuyển *movie_id* và *rating_id* thành dạng số thứ tự.

IV. Máy học:

1. Mô hình content-based:

a. Sơ lược về mô hình:

- *Content-based* là gì? Là mô hình gợi ý đánh giá đặc tính của *movies* để *recommended.*

b. Các bước xây dựng thuật toán:

- Gọi U, M lần lượt là số lượng *user* và *movie* thu thập được trong dữ liệu.

- Ma trận *ratings* Y được xây dựng dựa trên tập dữ liệu *ratings*, có kích thước M x U.

$$Y = \begin{bmatrix} y_{1,1} & \cdots & y_{1,U} \\ \vdots & \ddots & \vdots \\ y_{M,1} & \cdots & y_{M,U} \end{bmatrix}$$

- Trong đó $y_{i,j}$ là rating của *user* j cho *movie* i. Trong ma trận này có nhiều giá trị *rating* bị thiếu.
- Để gợi ý phim cho *user*, ta phải dự đoán được các giá trị *rating* thiếu trong ma trận Y.
- Với mỗi bộ phim ta sẽ xây dựng một *feature vector* dựa trên tập dữ liệu *movies*. Do đó, ta sẽ có ma trận *feature* X (M x d):

$$X = \begin{bmatrix} x_1 \\ \cdots \\ x_M \end{bmatrix}$$

- Trong đó: d là số thuộc tính của *movie*
  x_i là vector feature của *movie* i

	u1	u2	u3	u4	u5	u6	f1	f2
m1	?	4	5	2	1	?	0.11	0.89
m2	1	?	1	4	3	5	0.62	0.38
m3	4	5	2	?	?	4	0.21	0.79
m4	5	4	?	?	1	1	0.15	0.85
m5	?	3	4	5	3	1	0.2	0.8
	w_1	w_2	w_3	w_4	w_5	w_6		

- Ví dụ, *feature vector* của *movie* m1 là: x_1 = [0.11, 0.89]
- Ở phần tiền xử lý, ta đã chuyển dữ liệu *movies* thành ma trận *one hot*. Tại, đây ta sẽ sử dụng *l2 normalize* để chuẩn hóa từng vector (theo chiều ngang) trong ma trận *movie feature*.
- Với mỗi *user* j ta sẽ đi tìm *vector* w_j. Từ đó dự đoán được *rating* của *user* j cho *movie* i bằng công thức:

$$y_{i,j} = \theta((w_j)^T \times x_i)$$

- Bài toán trở thành bài toán *regression* trong trường hợp *rating* là 1đoạn giá trị liên tục hoặc bài toán phân lớp nếu *rating* là 1 vài giá trị rời rạc cụ thể. Đối với bài toán gợi ý phim, ta sử dụng *regression* để thực hiện.
- Đối với mô hình tuyến tính, đánh giá *rating* của *user* j cho *movie* i được tính bằng 1 hàm tuyến tính:

$$y_{i,j} = (w_j)^T \times x_i \quad (1)$$

- Xét một *user* j bất kỳ, nếu ta coi tập *training* là tập hợp các *rating* đã được đánh giá bởi *user* j, ta có thể xây dựng hàm mất mát như sau:

$$L_j = \frac{1}{2S_j} \sum_{i:r_{i,j}=1} ((w_j)^T \times x_i - y_{i,j})^2 + \frac{\lambda}{2S_j} \|w_j\|_2^2 \quad (2)$$

- Trong đó: S_j là số lượng *movie* đã được đánh giá bởi *user* j.

$r_{i,j} = 1$ khi *user* j đã đánh giá cho *movie* i.

- Hàm mất mát sẽ gồm 2 phần: trung bình sai số của mô hình và *l2-regularization* - giúp tránh *overfitting*.
- Vì biểu thức (2) chỉ phụ thuộc vào các *movie* đã được đánh giá bởi user j. Do đó, ta có thể rút gọn biểu thức lại như sau:

$$L_j = \frac{1}{2S_j} \|(w_j)^T \times \hat{X}_j - \hat{y}_j\|_2^2 + \frac{\lambda}{2S_j} \|w_j\|_2^2 \quad (3)$$

- Trong đó, $\hat{X}_j$ là ma trận con của ma trận *feature* X, chứa các *vector feature* của những bộ phim đã đánh giá bởi user j.
- Ta cần học ma trận: $W = [w_1 \quad \cdots \quad w_U]$. W là ma trận trọng số (d x U).
- Sử dụng mô hình *Ridge regression* để *training* => W. Tính được ma trận dự đoán *rating* Y (M x U). Trong đó, $y_{i,j}$ là dự đoán *rating* của *user* j cho *movie* i.

$$Y = X \times W = \begin{bmatrix} y_{1,1} & \cdots & y_{1,U} \\ \vdots & \ddots & \vdots \\ y_{M,1} & \cdots & y_{M,U} \end{bmatrix} \quad (4)$$

c. Tính toán lỗi:

- Từ ma trận *rating* Y ta thấy, mô hình của chúng ta dự đoán luôn cả những giá trị *rating* có sẵn, ta sẽ dựa vào những giá trị có sẵn này để tính toán lỗi.
- Sử dụng *Root Mean Squared Error* (RMSE), tức căn bậc hai của trung bình cộng bình phương của lỗi. Lỗi được tính là hiệu của true rating (những *rating* đã được người dùng đánh giá) và *predicted rating*.
- Kết quả thu được là độ lệch *rating* trung bình của giá trị dự đoán và giá trị thực (của những *rating* có sẵn).

2. Matrix Factorization (MF):

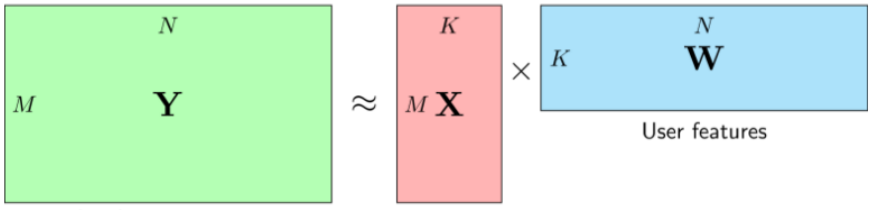
a. Sơ lược về mô hình:

- *Collaborative filtering (CF)*: hệ thống gợi ý *items* dựa trên sự tương quan (*similarity*) giữa các *users* và/hoặc *items*. Có thể hiểu rằng ở nhóm này một item được *recommended* tới một *user* dựa trên những *user* có hành vi tương tự.
- Trong mô hình này có 2 phương pháp chính: *Neighborhood-based Collaborative Filtering (NBCF)* và *Matrix Factorization Collaborative Filtering.*
- Về cơ bản, CF khắc phục được một số hạn chế của *content-based*. Thay vì recommend sản phẩm riêng biệt ở từng user, CF dựa vào sự tương quan giữa *user* và *item* để gợi nên thường hiệu quả cao hơn.

b. Các bước xây dựng thuật toán:

- Trong *Content-based*, mỗi *movie* được mô tả bằng một *vector* x được gọi là *movie feature*. *Rating* được dự đoán: $Y = X \times W$
- Bây giờ, thay vì xây dựng từ trước các *feature vector* từ tập dữ liệu *movies*, ta có thể huấn luyện X đồng thời với W. Điều này nghĩa là, biến số bài toán cần tối ưu là cả X và W; trong đó, X là ma trận của toàn bộ *movie profiles*, mỗi hàng tương ứng với 1 *movie*, W là ma trận của toàn bộ *user models*, mỗi cột tương ứng với 1 *user*.
- Với cách làm này ta chỉ sử dụng duy nhất tập *ratings* để huấn luyện.
- Ở đây, chúng ta đang cố gắng xấp xỉ ma trận *rating* $Y \in R^{M \times U}$ bằng tích của hai ma trận $X \in R^{M \times K}$ và $W \in R^{K \times U}$. Với M, U lần lượt là số lượng *movie* và *user*.

$$\mathbf{Y} \approx \hat{\mathbf{Y}} = \mathbf{X}\mathbf{W}$$

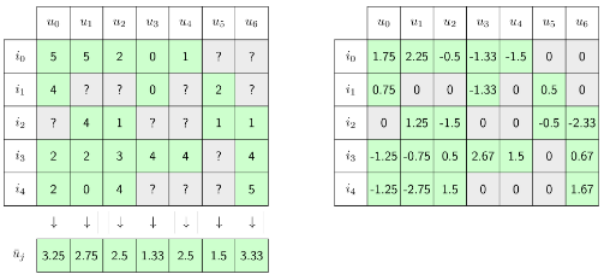


$$Y = \begin{bmatrix} y_{1,1} & \cdots & y_{1,U} \\ \vdots & \ddots & \vdots \\ y_{M,1} & \cdots & y_{M,U} \end{bmatrix} = X \times W = \begin{bmatrix} x_1 \\ \vdots \\ x_M \end{bmatrix} \times [w_1 \quad \dots \quad w_U] \quad (5)$$

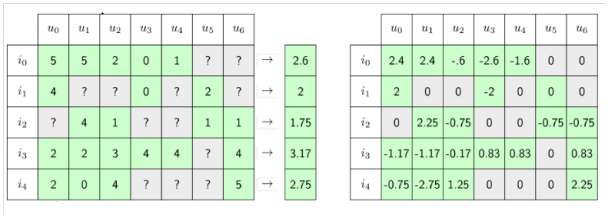
- Dự đoán *rating* của *user* u cho *movie* m: $y_{m,u} = x_m \times w_u$
- Ở đây K là gì?

- Ý tưởng chính đằng sau *Matrix Factorization* là tồn tại các *latent features* (tính chất ẩn) mô tả sự liên quan giữa các *movies* và *users*. Ví dụ với hệ thống gợi ý các bộ phim, tính chất ẩn có thể là hình sự, chính trị, hành động, hài, ...; cũng có thể là một sự kết hợp nào đó của các thể loại này; hoặc cũng có thể là bất cứ điều gì mà chúng ta không thực sự cần đặt tên.
- Vì vậy, K trong mô hình chính là số tính chất ẩn. Trong quá trình huấn luyện ta có thể cho K một giá trị (như một siêu tham số). Giá trị của K sẽ ảnh hưởng đáng kể đến quá trình học.

- Chuẩn hóa ma trận rating:
  - Giá trị *ratings* được chuẩn hoá bằng cách trừ mỗi hàng của *ratings matrix* đi trung bình cộng của các giá trị đã biết của hàng đó (*movie-based*) hoặc trừ mỗi cột đi trung bình cộng của các giá trị đã biết trong cột đó (*user_based*). Các giá trị thiếu sẽ thay bằng giá trị 0.
  - *User_based*:



  - *Movie_based*:



- Xây dựng và tối ưu hàm mất mát:
  - Việc xây dựng hàm mất mát trong MF khá giống với trong *content-based*, chỉ khác một chút là biến tối ưu là cả X và W.

$$L(X,W) = \frac{1}{2S}\sum_{u=1}^{U}\sum_{m:r_{m,u}=1}\left(x_m \times w_u - y_{m,n}\right)^2 + \frac{\lambda}{2}(\|X\|_F^2 + \|W\|_F^2) \quad (7)$$

  - Trong đó: S là tổng số *ratings*.

$r_{m,u} = 1$ khi *user* u đã đánh giá cho *movie* m

- Ta sẽ tối ưu X và W bằng *Gradient Descent*:
  - Khi tối ưu W, ta cố định X, khi đó ta có:

$$L(W) = \frac{1}{2S}\sum_{u=1}^{U}\sum_{m:r_{m,u}=1}\left(x_m \times w_u - y_{m,n}\right)^2 + \frac{\lambda}{2}\|W\|_F^2 (8)$$

  - Xét *user* u, ta có:

$$L(w_u) = \frac{1}{2S}\sum_{m:r_{m,u}=1}\left(x_m \times w_u - y_{m,n}\right)^2 + \frac{\lambda}{2}\|w_u\|_2^2 (9)$$

  - Biểu thức trong dấu $\sum$ của (9) chỉ phụ thuộc vào các *movies* đã được đánh giá bởi *user* u, ta có thể đơn giản nó: $\widehat{X}_u$ là *sub matrix* của X: là ma trận được tạo bởi các hàng của X ứng với các *movies* đã được đánh giá bởi *user* u và $\hat{y}_u$ là các *ratings* tương ứng. Khi đó:

$$L(w_u) = \frac{1}{2S}\left\|\widehat{X}_u \times w_u - \hat{y}_u\right\|_2^2 + \frac{\lambda}{2}\|w_u\|_2^2 (10)$$

  - Đạo hàm của (10):

$$\frac{\partial L(w_u)}{\partial w_u} = \frac{1}{s}\widehat{X}_u{}^T\left(\widehat{X}_u \times w_u - \hat{y}_u\right) + \lambda w_u (11)$$

  - Công thức cập nhật w_u, với η là hệ số học

$$w_u -= \eta\left(\frac{1}{S}\widehat{X}_u{}^T\left(\widehat{X}_u \times w_u - \hat{y}_u\right) + \lambda w_u\right) (12)$$

  - Tương tự với X, ta sẽ cố định W:

$$L(X) = \frac{1}{2S}\sum_{u=1}^{U}\sum_{m:r_{m,u}=1}\left(x_m \times w_u - y_{m,n}\right)^2 + \frac{\lambda}{2}\|X\|_F^2 (13)$$

  - Xét *movie* m:

$$L(x_m) = \frac{1}{2S}\sum_{m:r_{m,u}=1}\left(x_m \times w_u - y_{m,n}\right)^2 + \frac{\lambda}{2}\|x_m\|_2^2 (14)$$

  - Đặt $\widehat{W}_m$ là ma trận được tạo bằng các cột của W ứng với các *users* đã đánh giá *movie* đó và $\hat{y}^m$ là *vector ratings* tương ứng. (14) trở thành:

$$L(x_m) = \frac{1}{2S}\left\|x_m \times \widehat{W}_m - \hat{y}^m\right\|_2^2 + \frac{\lambda}{2}\|x_m\|_2^2 (15)$$

  - Đạo hàm:

$$\frac{\partial L(x_m)}{\partial x_m} = \frac{1}{s}\left(x_m \times \widehat{W}_m - \hat{y}^m\right) \times \widehat{W}_m{}^T + \lambda x_m (16)$$

- Công thức cập nhật:

$$x_m -= \eta(\frac{1}{s}(x_m \times \widehat{W}_m - \hat{y}^m) \times \widehat{W}_m{}^T + \lambda x_m \quad (17)$$

- Sử dụng tập *validation* thực hiện *stop early* cũng như để chọn ra các siêu tham số tốt nhất.

c. Tính toán lỗi:

- Giống như *content-based*, ta sử dụng *Root Mean Squared Error* (RMSE) để tính toán lỗi.

3. Kết quả:

a. *Content-based*:

- Nhận xét ảnh hưởng của việc chuẩn hóa *movie feature* đến quá trình học: chuẩn hóa ma trận movie feature cho kết quả tốt hơn.

```
In [17]: # TH1: không chuẩn hóa movie feature, learning_rate=0.1
W_1 = train_ridge_regression(train_ratings, X_nn, n_users)
Y_pred_1 = X_nn.dot(W_1)
# tính độ lỗi
train_err_1 = RMSE(train_ratings, Y_pred_1)
vali_err_1 = RMSE(vali_ratings, Y_pred_1)
print 'RMSE for train: ', train_err_1
print 'RMSE for validation:  ', vali_err_1

RMSE for train:  0.908151624472
RMSE for validation:   1.07518261218
```

```
In [18]: # TH2: chuẩn hóa movie feature, learning_rate=0.1
W_2 = train_ridge_regression(train_ratings, X_n, n_users)
Y_pred_2 = X_n.dot(W_2)
# tính độ lỗi
train_err_2 = RMSE(train_ratings, Y_pred_2)
vali_err_2 = RMSE(vali_ratings, Y_pred_2)
print 'RMSE for train: ', train_err_2
print 'RMSE for validation:  ', vali_err_2

RMSE for train:  0.908769544536
RMSE for validation:   1.05801430628
```

- Nhận xét ảnh hưởng của learning_rate đến quá trình học: khi giảm *learning_rate* thì càng bị *overfitting*. Từ thử nghiệm trên ta thấy, *learning_rate*=0.5 cho kết quả tốt nhất trên *validation.*

```
In [20]: # learning_rate=0.1
W_4 = train_ridge_regression(train_ratings, X_n, n_users, learning_rate=0.1)
Y_pred_4 = X_n.dot(W_4)
# tính độ lỗi
train_err_4 = RMSE(train_ratings, Y_pred_4)
vali_err_4 = RMSE(vali_ratings, Y_pred_4)
print 'RMSE for train: ', train_err_4
print 'RMSE for validation:  ', vali_err_4

RMSE for train:  0.908769544536
RMSE for validation:   1.05801430628
```

In [19]:
```
# learning_rate=0.5
W_3 = train_ridge_regression(train_ratings, X_n, n_users, learning_rate=0.5)
Y_pred_3 = X_n.dot(W_3)
# tính độ lỗi
train_err_3 = RMSE(train_ratings, Y_pred_3)
vali_err_3 = RMSE(vali_ratings, Y_pred_3)
print 'RMSE for train: ', train_err_3
print 'RMSE for validation:  ', vali_err_3
```

```
RMSE for train:  0.919323398701
RMSE for validation:   1.0346479408
```

In [21]:
```
# learning_rate=0.01
W_5 = train_ridge_regression(train_ratings, X_n, n_users, learning_rate=0.05)
Y_pred_5 = X_n.dot(W_5)
# tính độ lỗi
train_err_5 = RMSE(train_ratings, Y_pred_5)
vali_err_5 = RMSE(vali_ratings, Y_pred_5)
print 'RMSE for train: ', train_err_5
print 'RMSE for validation:  ', vali_err_5
```

```
RMSE for train:  0.907087641243
RMSE for validation:   1.06810553779
```

- Kết quả trên tập test (ứng với *movie feature* đã chuẩn hóa và *learning_rate*=0.5):

In [22]:
```
test_err = RMSE(test_ratings, Y_pred_3)
print 'RMSE for test: ', test_err
```

```
RMSE for test:  1.03009245862
```

b. *Matrix Factorization:*

- Ảnh hưởng của chuẩn hóa ma trận rating:
  - Cố định các giá trị *K = 10, max_patience=None, learning_rate=0.5, lamda=0.1, max_epoch=100.*

```
# user-based
# n_users và n_movies đã được tính từ lúc trước khi chia dữ liệu
X_u, W_u, avg_rating_u = train(train_ratings, vali_ratings, n_users, n_movies, user_based=1)
```

```
Info of returned: epoch  99  train RMSE:  1.02617194562 validation RMSE: 1.03934541922
```

```
# movie-based
X_m, W_m, avg_rating_m = train(train_ratings, vali_ratings, n_users, n_movies, user_based=0)
```

```
Info of returned: epoch  99  train RMSE:  0.973792050828 validation RMSE: 0.9821349886
```

  - Nhận xét: từ thử nghiệm trên ta thấy chuẩn hóa dữ liệu *theo movie-based* cho kết quả tốt hơn. Ở các thử nghiệm tiếp theo, ta sẽ mặc định sử dụng chuẩn hóa *movie-based.*
- Ảnh hưởng của K:
  - Cố định *max_patience=None, learning_rate=0.5, lamda=0.1, max_epoch=100.*

```
# K = 5
X_1, W_1, avg_rating_1 = train(train_ratings, vali_ratings, n_users, n_movies, K = 5)
```

```
Info of returned: epoch  99  train RMSE:  0.974129980177 validation RMSE: 0.979579511178
```

```
# K = 30
X_2, W_2, avg_rating_2 = train(train_ratings, vali_ratings, n_users, n_movies, K = 30)
```

```
Info of returned: epoch  99  train RMSE:  0.974129860668 validation RMSE: 0.979579912528
```

```
# K = 50
X_3, W_3, avg_rating_3 = train(train_ratings, vali_ratings, n_users, n_movies, K = 50)
```

```
Info of returned: epoch  99  train RMSE:  0.974130272562 validation RMSE: 0.979579868461
```

- Nhận xét: Giá trị K lớn có thể gây *overfitting*. Trong trường hợp này, *K=30* cho độ lỗi nhỏ nhất trên tập *validation*. Ta sẽ dùng giá trị này để huấn luyện và kiểm tra trên *test*.

- Ảnh hưởng của learning_rate: cố định các giá trị *K = 10, max_patience=None, lamda=0.1, max_epoch=100.*

```
# learning_rate=0.5
X_4, W_4, avg_rating_4 = train(train_ratings, vali_ratings, n_users, n_movies, learning_rate=0.5)
```

```
Info of returned: epoch  99  train RMSE:  0.974130296693 validation RMSE: 0.979579868751
```

```
# learning_rate=0.1
X_5, W_5, avg_rating_5 = train(train_ratings, vali_ratings, n_users, n_movies, learning_rate=0.1)
```

```
Info of returned: epoch  99  train RMSE:  1.06025183522 validation RMSE: 1.06561020565
```

```
# learning_rate=0.01
X_6, W_6, avg_rating_6 = train(train_ratings, vali_ratings, n_users, n_movies, learning_rate=0.01)
```

```
Info of returned: epoch  99  train RMSE:  1.51415658309 validation RMSE: 1.51879106153
```

- Nhận xét: learning càng giảm thì độ lỗi trên train và validation càng tăng. Giá trị learning_rate=0.5 cho kết quả tốt nhất, ta sẽ sử dụng giá trị này để thử nghiệm với tập test.

- Ảnh hưởng của weight decay (lamda): cố định các giá trị *K = 10, max_patience=None, learning_rate=0.5, max_epoch=100.*

```
# lamda=0.0
X_7, W_7, avg_rating_7 = train(train_ratings, vali_ratings, n_users, n_movies, lamda=0.0)
```

```
Info of returned: epoch  99  train RMSE:  3.14772845478 validation RMSE: 3.1572188132
```

```
# lamda=0.1
X_8, W_8, avg_rating_8 = train(train_ratings, vali_ratings, n_users, n_movies, lamda=0.1)
```

```
Info of returned: epoch  99  train RMSE:  0.974130194555 validation RMSE: 0.979580354061
```

- Nhận xét: *lamda=0.1* cho kết quả tốt hơn nhiều khi *lamda=0*.

- Kết hợp stop early huấn luyện và kiểm tra trên test:

```
X, W, avg_rating = train(train_ratings, vali_ratings, n_users, n_movies, K = 30,
                         max_patience=50, learning_rate=0.5, lamda=0.1, max_epoch=200)
```

```
Info of returned: epoch  77  train RMSE:  0.962130172264 validation RMSE: 0.967280623471
```

```
test_err = RMSE(test_ratings, X, W, avg_rating)
print "Test RMSE: ", test_err
```

```
Test RMSE:  0.966832473621
```

4. Nhận xét:

- *Content-based*:
  - Ưu điểm: khá đơn giản, thời gian chạy nhanh.
  - Nhược điểm:
    - Chỉ dựa vào thông tin *rating* của một user và đặc điểm của *movie* để gợi ý cho *user* đó.
    - Không khai thác hết mối quan hệ giữa các *user* hay *movie*, và toàn bộ tập dữ liệu *ratings*.
- *Matrix Factorization*:
  - Ưu điểm: tận dụng được toàn bộ tập dữ liệu *ratings* và mối quan hệ giữa các *user*, *movie*. Nếu điều chỉnh các tham số phù hợp sẽ cho kết quả tốt hơn so với mô hình content-based.
  - Nhược điểm: không sử dụng tập dữ liệu *movies*. Tốc độ chạy thuật toán lâu hơn nhiều so với *content-based*.

5. Tổng kết:

- Trên đây chỉ là 2 trong nhiều mô hình để xây dựng một hệ thống gợi ý người dùng.
- Vẫn có một số mô hình thuật toán tốt hơn, có thể tận dụng tối đa cả 2 tập dữ liệu ratings và movies, thậm chí cả tập dữ liệu thông tin user.

V. Tham khảo:

https://www.youtube.com/watch?v=saXRzxgFN0o&list=PL_npY1DYXHPT-3dorG7Em6d18P4JRFDvH

http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

https://hal.archives-ouvertes.fr/inria-00580523/document

https://pdfs.semanticscholar.org/7e98/a98bbc25ab2e4e425d802b3e48257984435e.pdf

http://herbrete.vvv.enseirb-matmeca.fr/IR/CF_Recsys_Survey.pdf