TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

HỆ THỐNG GỢI Ý SẢN PHẨM

BÁO CÁO CUỐI KÌ

HỆ HỖ TRỢ QUYẾT ĐỊNH

Giảng viên hướng dẫn : TS. LÊ HẢI HÀ

Sinh viên thực hiện : NGUYỄN MINH ĐỨC

Lớp : Toán Tin 01-K64

Mã lớp học : 133604

Học kì : 20212

HÀ NỘI, 8-2022

Mục lục

Lời cảm ơn

Môn học hệ hỗ trợ quyết định đã mang lại cho em cái nhìn rõ hơn về các thuật toán giúp con người đưa ra quyết định tốt hơn, thông qua việc cài đặt các thuật toán đã cải thiện cho bản thân em các kĩ năng về lập trình cũng như là kĩ năng đọc tài liệu và trình bày.

Em xin chân thành cảm ơn thầy Lê Hải Hà đã hướng dẫn em tận tình và giải đáp những thắc mắc kịp thời trong môn học. Nhờ sự hướng dẫn của thầy đã giúp em hoàn thành môn học một cách tốt nhất.

Hà Nội, tháng 8 năm 2022

Sinh viên thực hiện

Nguyễn Minh Đức

Chương 1

Cơ sở lí thuyết

1.1 Giới thiệu vấn đề

Trong thế giới ngày nay, với sự phát triển vượt bậc của nền văn minh nhân loại, công nghệ có nhiều những đột phá mới, con người được hưởng lợi từ sự hiện đại hóa của xã hội. Những công nghệ mới được sản sinh ra để phục vụ cho những tiện ích của con người, với sự bùng nổ của mạng internet những năm vừa qua, các nền tảng mạng xã hội trực tuyến cùng với đó cũng kéo theo những vấn đề xung quanh nó. Con người hiện đại ưa chuộng các hình thức mua sắm trực tuyến vì sự tiện lợi và dễ dàng sử dụng. Một vấn đề đặt ra làm sao đưa đến thông tin sản phẩm phù hợp với người dùng. Tức chúng ta cần gợi ý cho người dùng những sản phẩm phù hợp với họ, để họ có thể dễ dàng tiếp cận và sử dụng từ đó nâng cao doanh số của các của hàng cũng như là khách hàng có thể được sử dụng loại sản phẩm mà họ yêu thích.

Các cửa hàng truyền thống, để thu hút được khách hàng, họ thường trưng bày các sản phẩm đang được ưa thích hoặc phổ biến trên thị trường để tạo ấn tượng với khách hàng, và các sản phẩm ít được phổ biến hơn thì họ thường cất trong kho, hoặc không để ra trưng

bày. Với cách làm đó thì những sản phẩm được trưng bày ra chưa chắc đã phù hợp với người mua, còn những sản phẩm không được trưng bày thì có thể phù hợp khách hàng nhưng họ lại không tìm thấy sản phẩm. Đó là một hạn chế với những cửa hàng truyền thống khi họ không thể trưng bày tất cả sản phẩm của họ. Nhưng đối với các cửa hàng trực tuyến thì tất cả sản phẩm của họ được đưa đến với người dùng những thông tin đầy đủ nhất. Và khi mà lượng thông tin về các sản phẩm quá nhiều, làm cho người dùng không biết nên lựa chọn sản phẩm nào thì lúc này một vấn để xảy ra là làm thế nào có thể gợi ý cho người dùng những sản phẩm mà phù hợp với họ nhất. Lúc này những ý tưởng về bài toán gợi ý đã ra đời và mang lại những hiệu quả nhất định, giúp cho những công ty có thể tăng doanh số của mình. Hệ thống gợi ý là công cụ lọc thông tin mạnh mẽ có thể thúc đẩy các dịch vụ cá nhân hóa và cung cấp trải nghiệm riêng biệt cho từng người dùng. Nói ngắn gọn, hệ thống đề xuất đóng vai trò nòng cốt trong việc tận dụng nguồn dữ liệu dồi dào hiện có để giúp việc đưa ra lựa chọn dễ dàng hơn.

1.2 Mô hình bài toán

Như đã đề cập, những dữ liệu trong bài toán của chúng ta là thông tin người dùng, thông tin sản phẩm, và một chỉ số thể hiện người dùng có quan tâm đến sản phẩm đó hay không. Ta có biến *user* thể hiện cho người dùng, *item* thể hiện cho sản phẩm, và một chỉ số đo mức độ quan tâm của *user* với *item* là *rating*, ta có thể hiểu đó là đánh giá của người dùng cho sản phẩm đó.

Từ các khái niệm đó ta có được một ma trận tiện ích (Utility matrix) là tập hợp tất cả *rating* của tất cả người dùng với tất cả sản phẩm, kể cả các *rating* chưa biết.

	user_1	user_2	user_3	user_4	user_5
item_1	1	4	3	5	2
item_2	5	?	?	3	?
item_3	0	4	?	2	2
item_4	?	4	?	1	2

Bảng 1.1: Ma trận tiện ích (Utility matrix)

Đây là ví dụ về ma trận tiện ích có 5 *user* và 4 *item* và các *rating* tương ứng, ở đây ta thấy có một số *rating* chưa được người dùng đánh giá sản phẩm và để ở trạng thái dấu "?". Công việc của ta là dự đoán các *rating* đó để đoán xem người dùng có quan tâm đến sản phẩm đó hay không để gợi ý cho họ. Hay nói cách khác bài toán của chúng ta là đang đi điền các giá trị "?" để hoàn thành ma trận.

1.3 Phương pháp Matrix factorization

Cách tiếp cận bằng việc đi khám phá các nhân tố ẩn bên trong ma trận tiện tích được lựa chọn phổ biến và đạt được sự thành công dựa trên phương pháp phân tích ma trận thành nhân tử (Matrix factorization - MF).

1.3.1 Mô hình Matrix factorization

Để biểu diễn những nhân tố ẩn quyết định đến sự đánh giá của người dùng bên trong *item_i*, ta định nghĩa vector $q_i \in R^f$, và định nghĩa các nhân tố tiềm ẩn bên trong người dùng u là vector $p_u \in R^f$. Đối với *item i* nó sẽ mang nhân tố ẩn ở một mức độ nào đó được thể hiện trong các hệ số tương ứng trong vector q_i, có thể là tiêu cực hoặc tích cực.

Tương tự với mỗi *user u* các yếu tố của p_u đo lường mức độ quan tâm của người dùng đối với các mặt hàng trên các yếu tố ẩn tương ứng. Và kết quả của phép nhân 2 vector $q_i^T p_u$ là kết quả đánh giá của *user u* lên *item i*. Để dự đoán cho các *rating* bị thiếu trong ma trận tiện tích, kí hiệu $\hat{r}_{ui}$ là các giá trị đánh giá còn thiếu của *user u* đối với *item i*, được tính như sau:

$$\hat{r}_{ui} = q_i^T p_u \tag{1.1}$$

Bằng cách học từ dữ liệu ta sẽ tìm các q_i và p_u xấp xỉ được gần nhất với các giá trị còn thiếu trong ma trận tiện ích.

Bài toán của chúng ta có m sản phẩm, n người dùng, do đó ma trận tiện ích U có kích cỡ là $m \times n$.

Với m *item* ta sẽ có được ma trận chứa các nhân tố ẩn của các *item* là $Q_{f \times m}$, đối với n *user* ta có ma trận $P_{f \times n}$. Số f ở đây chính là số các nhân tố tiềm ẩn.

Mô hình hóa dưới dạng các ma trận ta có:

$$U \approx \hat{U} = Q^T P \tag{1.2}$$

Bằng cách này ta đang đi tìm một xấp xỉ ma trận U bằng tích 2 ma trận $Q \in R^{f \times m}$ và ma trận $P \in R^{f \times n}$. Thông thường ta chọn f là số nhỏ hơn rất nhiều so với m, n.

1.3.2 Hàm mất mát (Loss function)

Để tính lỗi cho các giá trị dự đoán, ta dựa vào phương pháp bình phương cực tiêu. Mục đích của ta là đi tối thiểu sai số giữa giá trị đúng và giá trị dự đoán:

$$\min_{q*,p*} L = \frac{1}{2k} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \frac{\lambda}{2}(\|q_i\|^2 + \|p_u\|^2) \tag{1.3}$$

Ta có K là tập các cặp (u,i) mà ta đã biết giá trị đánh giá r_{ui} hay chính là tập training của chúng ta, k là số phần tử tập K. Thành phần thứ nhất là trung bình sai số của mô hình, thành phần thứ hai chính là regularized giúp cho mô hình tránh được việc học quá khớp (hiện tượng overfitting). λ là siêu tham số. $\|q_i\|^2$ và $\|p_u\|^2$ là chuẩn Euclidean.

1.3.3 Tối ưu hàm mất mát

Để tìm được nghiệm tối ưu cho bài toán tối thiểu hàm mất mát, ta sử dụng phương pháp Gradient descent để giải bài toán tối ưu. Tư tưởng của Gradient descent là ta sẽ dịch chuyển dần các điểm chấp nhận được về nghiệm tối ưu.

Đối với bài toán tối ưu hàm mất mát (1.3) nghiệm tối ưu là ma trận $Q*$ và $P*$ làm cho giá trị hàm mất mát đạt nhỏ nhất.

Công việc của ta là đi tối ưu từng vector q_i và p_u.

Ta tính đạo hàm của hàm mất mát theo q_i và p_u:

$$\frac{\partial L}{\partial q_i} = -\frac{1}{k} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u) p_u + \lambda q_i \tag{1.4}$$

$$\frac{\partial L}{\partial p_u} = -\frac{1}{k} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u) q_i + \lambda p_u \tag{1.5}$$

Để dễ dàng cho việc tính toán ta thay thế tổng ở trên bằng ma trận. Ta định nghĩa ma trận $\hat{P}_i$ là ma trận các *user* đã *rating* cho *item i*. Ma trận $\hat{Q}_u$ là ma trận các *item* mà *user u* đã *rating*. Ta được công thức đạo hàm mới:

$$\frac{\partial L}{\partial q_i} = -\frac{1}{k}(\hat{r}_i - q_i^T \hat{P}_i)\hat{P}_i^T + \lambda q_i \tag{1.6}$$

$$\frac{\partial L}{\partial p_u} = -\frac{1}{k}\hat{Q}_u^T(\hat{r}_u - \hat{Q}_u^T p_u) + \lambda p_u \tag{1.7}$$

Công thức cập nhập cho mỗi vector q_i và p_u theo Gradient descent là:

$$q_i = q_i - \eta \frac{\partial L}{\partial q_i} \tag{1.8}$$

$$p_u = p_u - \eta \frac{\partial L}{\partial p_u} \tag{1.9}$$

Với η là tốc độ học (learning rate).

1.3.4 Thuật toán

- Bước 1: Khởi tạo giá trị ngẫu nhiên các ma trận Q, P.
- Bước 2: Tính hàm mất mát.

$$Loss = \frac{1}{2k}\sum_{(u,i)\in K}(r_{ui} - q_i^T p_u)^2 + \frac{\lambda}{2}(\|q_i\|^2 + \|p_u\|^2)$$

- Bước 3: Nếu giá trị của hàm mất mát $Loss \leq \varepsilon$ cho trước thì chuyển đến bước 5, nếu không thì chuyển đến bước 4.
- Bước 4: Cập nhật ma trận P, Q mới và quay lại bước 2.
- Bước 5: Kết thúc thuật toán, giá trị tối ưu là ma trận P và Q

Chương 2

Thực hành và kết quả

2.1 Bộ dữ liệu

Tập dữ liệu này là "Social Recommendation Data" từ "Hệ thống đề xuất và tập dữ liệu cá nhân hóa". Nó chứa các đánh giá của người dùng về sách trên Librarything. Dữ liệu sử dụng cho mô hình bao gồm 3 trường được trích rút từ tập dữ liệu trên là người dùng, tên sách, và số điểm đánh giá.

2.1.1 Giải nén dữ liệu

Đầu tiên ta cần mở dữ liệu và giải nén bằng đoạn code Ptyhon sau:

```
import tarfile
with tarfile.open(path_data) as tar:
    print("Files in tar archive:")
    tar.list()
    with tar.extractfile("lthing_data/reviews.json") as file:
```

```
        count = 0
        for line in file:
            print(line)
            count += 1
            if count > 3:
                break

import ast
reviews = []
with tarfile.open(path_data) as tar:
    with tar.extractfile("lthing_data/reviews.json") as file:
        for line in file:
            record = ast.literal_eval(line.decode("utf8"))
            if any(x not in record for x in ['user', 'work', 'stars']):
                continue
            reviews.append([record['user'], record['work'],
                record['stars']])
print(len(reviews), "records retrieved")
```

Kết quả đoạn code trên là ta giải nén được: 1387209 bản ghi dữ liệu.

2.1.2 Tiền xử lí dữ liệu

Ta sẽ xử lí dữ liệu bằng thư viện Pandas của Python. Chuyển dữ liệu về dạng ma trận về mức độ đánh giá của những người dùng khác nhau đối với mỗi cuốn sách.

```
import pandas as pd
reviews = pd.DataFrame(reviews, columns=["user", "work", "stars"])
print(reviews.head())
```

Kết quả:

```
           user     work  stars
0      van_stef  3206242   5.0
1      dwatson2 12198649   5.0
2      amdrane2 12981302   4.0
3  Lila_Gustavus  5231009   3.0
4     skinglist   184318   2.0
```

Tiếp theo ta sẽ lọc ra số lượng người dùng có lượng đánh giá lớn hơn 30 lượt để tiết kiệm thời gian tính toán và chi phí cho mô hình thực nghiệm.

```
usercount = reviews[["work","user"]].groupby("user").count()
usercount = usercount[usercount["work"] >=30]
print(usercount)
usercount["user_ID"] = range(len(usercount))
```

Output:

```
           work
user
             84
-AlyssaE-    41
-Eva-       602
06nwingert  370
0703         46
...         ...
zquilts      67
zsms         33
```

```
zwaantje    121
zyabuko      30
zzshupinga  246

[8907 rows x 1 columns]
```

Và ta cũng làm tương tự, lọc ra số sách có lượt đánh giá từ 30 lượt trở lên.

```
workcount = reviews[["work","user"]].groupby("work").count()
workcount = workcount[workcount["user"] >= 30]
workcount['items_ID'] = range(len(workcount))
```

Output:

```
work     user  items_ID
10000      106  0
10001      53   1
1000167    186  2
10001797   53   3
10002      45   4
...          ...  ...
9993888    34   6012
9997       42   6013
9997232    64   6014
9998       66   6015
9998836    31   6016
```

Kết quả ta có: 8907 user và 6017 item.

Bộ dữ liệu được lọc ra như sau:

```
reviews = reviews[reviews["user"].isin(usercount.index) &
   reviews["work"].isin(workcount.index)]
print(reviews)
```

Output:

```
  user     work  stars
0            van_stef  3206242   5.0
3       Lila_Gustavus  5231009   3.0
6             justine     3067   4.5
10         curvymommy 13206089   3.5
15        jwhenderson     1518   4.0
...               ...      ...   ...
1387177    BruderBane    24623   4.5
1387182  whymaggiemay     7683   4.0
1387192   StuartAston  8282225   4.0
1387202       danielx  9759186   4.0
1387206      jclark88  8253945   3.0
```

Tiếp theo ta sẽ thay thế các tên người dùng và tên sách bằng các ID của chúng để thuận tiện cho việc lập ma trận đánh giá.

```
reviews = pd.merge(reviews, workcount, on= ['work'] )
reviews = reviews.rename(columns={'user_x': 'user'})
reviews = pd.merge(reviews, usercount, on= ['user'] )
reviews = reviews.rename(columns={'work_x': 'work'})
data = reviews[['user_ID','items_ID','stars']].sort_values('user_ID')
data.head()
```

Output:

```
user_ID items_ID stars
   0      2806    3.5
   0      1549    3.0
   0      2975    3.0
   0      233     4.0
   0      250     4.0
```

Tiếp theo ta sẽ chia tập dữ liệu ban đầu thành 2 tập là tập Train để huấn luyện mô hình và tập Val đề đánh giá mô hình.

```
val = data.sample(frac=0.05,random_state= 42)
data_1 = data.drop(val.index)
```

2.2 Xây dựng mô hình

Đầu tiên ta tạo ra một class có tên là MF, chính là mô hình Matrix Factorization. Hàm khởi tạo của class chứa các tham số đầu vào như dữ liệu train, dữ liệu val, tham số k, λ, learning rate,...

```
def __init__(self, Y_data, val_set, k, lam = 0.75, X_init = None, W_init =
    None,n_users = None, n_items = None, learning_rate = 0.2, Max_iter =
    1000, n_ratings = None,print_every = 10):
        self.Y_data = Y_data.values
        self.Y_val_set = val_set.values
        self.lamda = lam
        self.learning_rate = learning_rate
        self.Max_iter = Max_iter
```

```
        self.n_users = n_users
        self.n_items = n_items
        self.n_ratings = n_ratings
        self.print_every = print_every
        self.K = k
        #init P,Q
        if Q_init is not None:
            self.Q = Q_init
        else:
            self.Q = np.random.normal(0.5, 0.5,size=(n_items, k))
        if P_init is not None:
            self.P = P_init
        else:
            self.P = np.random.normal(0.5, 0.5, size= (k, n_users))
        self.Y_data_raw = self.Y_data.copy()
        self.val_set = self.Y_val_set.copy()
```

Ta viết các hàm bên trong thân class.

Hàm chuẩn hóa dữ liệu, ta chuẩn hóa điểm đánh giá của những người dùng bằng cách trừ đi điểm đánh giá trung bình. Việc trừ đi trung bình cộng khiến các điểm đánh giá có những giá trị dương và âm. Những giá trị dương tương ứng với việc user thích item, những giá trị âm tương ứng với việc user không thích item. Những giá trị bằng 0 tương ứng với việc chưa xác định được liệu user có thích item hay không.

```
def normalize_data(self):
        n_objects = self.n_users
        users = self.Y_data_raw[:, 0].astype(np.int32)
        users_val = self.val_set[:,0].astype(np.int32)
```

```
        self.mu = np.zeros((n_objects,))
        for n in range(n_objects):
            # row indices of rating done by user n
            # since indices need to be integers, we need to convert
            ids = np.where(users == n)[0].astype(np.int32)
            # indices of all ratings associated with user n
            item_ids = self.Y_data_raw[ids, 1]
            # and the corresponding ratings
            ratings = self.Y_data_raw[ids, 2]
            # take mean
            m = np.mean(ratings)
            if np.isnan(m):
                m = 0 # to avoid empty array and nan value
            self.mu[n] = m
            # normalize
            self.Y_data_raw[ids, 2] = ratings - self.mu[n]

        for n in range(n_objects):
            ids = np.where(users_val == n)[0].astype(np.int32)
            ratings = self.val_set[ids, 2]
            self.val_set[ids, 2] = ratings - self.mu[n]
```

Tiếp theo là hàm mất mát theo công thức đã nêu ở trên:

```
def loss(self):
        loss = 0
        for i in range(self.n_ratings):
            n, m , r = int(self.Y_data_raw[i, 0]),
                int(self.Y_data_raw[i,1]), self.Y_data_raw[i,2]
```

```
            loss += (r - self.Q[m, :].dot(self.P[:, n]))**2
        loss = loss/(2*self.n_ratings)
        loss += 0.5*self.lamda* (np.linalg.norm(self.Q,'fro')**2 +
            np.linalg.norm(self.P, 'fro')**2)
        return loss
```

Hàm *get_items_rated_by_user()* có chức năng tìm các điểm đánh giá của item mà một user đã đánh giá.

```
def get_items_rated_by_user(self, user_id):
        """
        get all items which are rated by user and the corresponding ratings
        """
        ids = np.where(self.Y_data_raw[:,0].astype(np.int32) == user_id)[0]
        item_ids = self.Y_data_raw[ids, 1].astype(np.int32) # indices need
            to be integers
        ratings = self.Y_data_raw[ids, 2]
        return (item_ids, ratings)
```

Tương tự hàm *get_users_who_rate_item()* trả về điểm đánh giá của các user đã đánh giá item đó.

```
def get_users_who_rate_item(self, item_id):
        """
        get all users who rated item item_id and get the corresponding
            ratings
        """
        ids = np.where(self.Y_data_raw[:,1].astype(np.int32) == item_id)[0]
        user = self.Y_data_raw[ids, 0].astype(np.int32)
```

```
        ratings = self.Y_data_raw[ids, 2]
        return (user, ratings)
```

Tiếp theo ta viết các hàm để cập nhật ma trận tham số Q, P.

```
def update_Q(self):
        for m in range(self.n_items):
            user, ratings = self.get_users_who_rate_item(m)
            Pm = self.P[:, user]
            error = -(ratings - self.Q[m, :].dot(Pm))
            # gradient
            grad_qm = error.dot(Pm.T)/self.n_ratings + self.lamda*self.Q[m,
                :]
            self.Q[m, :] -= self.learning_rate*grad_qm.reshape((self.K,))

    def update_P(self):
        for n in range(self.n_users):
            item_ids, ratings = self.get_items_rated_by_user(n)
            Qn = self.Q[item_ids, :]
            # gradient
            error = ratings - Qn.dot(self.P[:, n])
            grad_pn = -Qn.T.dot(error)/self.n_ratings +
                self.lamda*self.P[:, n]
            self.P[:, n] -= self.learning_rate*grad_pn.reshape((self.K,))
```

Hàm *fit()* thực hiện việc huấn luyện mô hình.

```
def fit(self):
        self.normalize_data()
        for it in range(self.Max_iter):
            self.update_Q()
            self.update_P()
            if (it + 1) % self.print_every == 0:
                rmse_train = self.evaluate_RMSE(self.Y_data)
                rmse_val = self.evaluate_RMSE(self.Y_val_set)
                print ('iter =', it + 1, ', loss =', self.loss(), ', RMSE
                    train =', rmse_train, ', RMSE val = ', rmse_val)
            if self.loss() <=0.01:
                break
```

Hàm đánh giá RMSE.

```
def evaluate_RMSE(self, rate_test):
        n_tests = rate_test.shape[0]
        SE = 0 # squared error
        for n in range(n_tests):
            pred = self.pred(rate_test[n, 0], rate_test[n, 1])
            SE += (pred - rate_test[n, 2])**2
        RMSE = np.sqrt(SE/n_tests)
        return RMSE
```

Hàm *pred()* trả về điểm đánh giá được dự đoán.

```
def pred(self, u, i):
        u = int(u)
```

```
        i = int(i)
        pred = self.Q[i, :].dot(self.P[:, u]) + self.mu[u]
        # truncate if results are out of range [0, 5]
        if pred < 0:
            return 0
        if pred > 5:
            return 5
        return pred
```

Ta thực hiện huấn luyện mô hình.

```
FC  = MF(data_1, val, 10 ,lam= 0.75,n_users= n_users,n_items= n_items,
   n_ratings=n_ratings, learning_rate= 1.5)
FC.fit()
```

2.3 Đánh giá kết quả thực nghiệm

Thực hiện huấn luyện mô hình với số lượng nhân tố ẩn khác nhau, ta có được các đánh giá lỗi và RMSE trên các tập Train và tập Val như sau:

f	Loss	RMSE Train	RMSE Val
5	0.5535	1.0314	1.0513
15	0.5768	1.0241	1.0746
25	0.5662	1.0244	1.0278

Bảng 2.1: Kết quả huấn luyện mô hình.

Kết quả trên cho ta thấy giá trị tối ưu hội tụ về xấp xỉ gần nhau ở các giá trị nhân tố

ẩn f khác nhau. Với các giá trị f lớn hơn, cần số vòng lặp lớn hơn để huấn luyện mô hình với lượng tham số lớn hơn, nhưng kết quả cũng không tối ưu hơn, nên ta chọn $f = 5$ là phù hợp cho mô hình này. Sự chênh lệch giữa RMSE train và RMSE val nhỏ, do đó mô hình đang không bị học quá khớp. Các giá trị đánh giá RMSE vẫn còn khá lớn và cần cải thiện nhiều hơn để mô hình học tốt hơn.

Chương 3

Kết luận

Bài toán hệ thống gợi ý đã đặt ra một vấn đề rất phổ biến hiện nay, làm sao để mang lại trải nghiệm tốt nhất cho khách hàng, gợi ý đúng sản phẩm phù hợp với người dùng. Việc đưa ra được các dự đoán gợi ý chính xác cho người dùng mang lại hiệu quả rất tích cực cho các doanh nghiệp tổ chức trong việc bán sản phẩm của mình, họ sẽ bán được đúng sản phẩm cho đúng khách hàng.

Bằng việc áp dụng các thuật toán ta có thể xây dựng được một mô hình đơn giản cho việc dự đoán, nhưng kết quả lại chưa khả quan. Mô hình trong báo cáo được em xây dựng còn đơn giản và chưa tận dụng được hết nguồn dữ liệu phục vụ cho việc dự đoán.

Hệ thống gợi ý sử dụng phương pháp phân rã ma trận mang lại hiệu quả tốt, khi mô hình học được các nhân tố tiềm ẩn bên trong khách hàng hoặc sản phẩm mà các phương pháp khác không chỉ ra được.

Tài liệu tham khảo

Tiếng Việt

1. Vũ Hữu Tiệp, *Machine Learning cơ bản.*

Tiếng Anh

1. Ismail Ahmed Al-Qasem Al-Hadi, Nurfadhlina Mohd SharefMd Nasir Sulaiman and Norwati Mustapha, "Review of the Temporal Recommendation System with Matrix Factorization", *International Journal of Innovative Computing, Information and Control*, Vol 13;(5), 2017, pp 1579-1594.

2. Iateilang Ryngksai and L. Chameikho, "Recommender Systems: Types of Filtering Techniques", *International Journal of Engineering Research Technology (IJERT)*, Vol 3;(11), 2014.

3. Yehuda Koren, Robert Bell and Chris Volinsky, "Matrix factorization techniques for recommender systems", *Computer (Long. Beach. Calif.)*, Vol 42;(8), 2009, pp.42-49.