Deep Matrix Factorization Models for Recommender Systems – 2017

1. **Paper**
2. **Tóm tắt:**

+ Matrix factorization là một phương pháp phổ biến của việc lọc cộng tác có ý tưởng cơ bản là dự đoán xếp hạng cá nhân thông qua một tập hạng mục cho mỗi người dùng tương đồng với những người dùng và hạng mục. Phương pháp này sẽ học một không gian tiềm ẩn để biểu diễn một người dùng hoặc một hạng mục với những ưu điểm về tính co giãn, tính đơn giản, và tính linh hoạt.

+ Bài báo này trình bày một phương pháp matrix factorization mới kết hợp mạng neural. Hai điểm chính của bái báo là: xây dựng một ma trận user-item với rating cho vào học trong một mạng học sâu và thiết kế một hàm mất mát dựa trên độ đo binary cross entropy.

1. **Công trình liên quan:**

+ Matrix Factorization (Sarwar et al., 2001; Linden et al., 2003; Billsus and Pazzani, 1998; Koren *et al.*, 2009).

+ MF kết hợp với các thông tin khác để giải quyết tính thưa thớt của rating (Ma et al., 2008; Tang et al., 2013)

+ MF theo chủ để với nội dung hạng mục hay văn bản đánh giá (McAuley and Leskovec, 2013; Bao *et al.*, 2014)

+ MF sử dụng phản hồi từ người dùng (The SVD++ [Koren, 2008], Rendle *et al.*, 2009; Mnih and Teh, 2012; He and McAuley, 2015)

+ Sử dụng mạng học sâu: RBM (2007), Autoencoders and the denoising autoencoders [Li *et al.*, 2015; Sedhain *et al.*, 2015; Strub and Mary, 2015], CDAE (2016), NCF (2017)

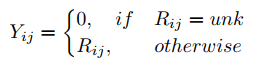
1. **Đóng góp:**

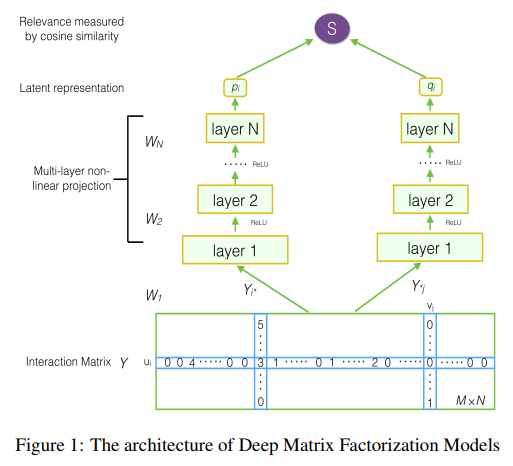
* Sử dụng cả hai thông tin explicit rating và implicit feedback. Với đầu vào là explicit rating và 0 cho implicit feedback. Cho dữ liệu này học trong một không gian có số chiều tiềm ẩn thấp bởi những dùng và các hạng mục. Kiến trúc mạng này được lấy từ mạng DSSM (deep structured sematic model) được sử dụng trong web search.
* Thiết kế một hàm mất mát dựa trên cross entropy, hiệu quả trên explicit ratings và implicit feed back.

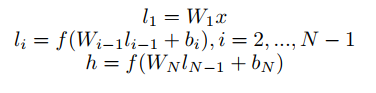
1. **Mô hình:**

* **Deep structure sematic model**:Đây là mô hình được sử dụng cho web search. Nó sử dụng mạng học sâu để xếp hạng một tập các kết quả phù hợp với câu truy vấn. DSSM đầu tiên kết nối câu truy vấn và kết quả vào không gian ngữ nghĩa thấp hơn với một phép biển đổi nhiều tầng phi tuyến tính. Sau đó web search xếp hạng những câu truy vấn liên quan với kết quả được tính theo độ tương quan cosine giữa vector chiều nhỏ của câu truy vấn và kết quả.
* **Deep matrix factorization model (DMF):**

Ma trận tương tác user-item:

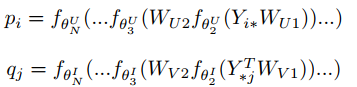


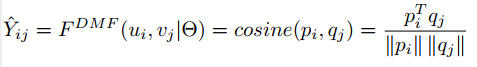
* là rating của user i trên item j.
* là ma trận tương tác user-item.
* là tập các tương tác quan sát được, là tất cả phần tử bằng không trong , là tập các thể hiện âm. là tất các các tương tác huấn luyện.
  + 
* Đầu vào là ma trận Y.
* Sau đó được chia thành 2 nhánh: một nhánh là ma trận user-item, một nhánh là ma trận item-user (đảo hàng thành cột)
* Với mỗi nhánh là một mạng neural nhiều tầng với công thức tổng quát như sau:



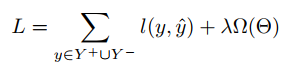
Hàm relu là activation function: f(x) = max(0,x).

* Sau đó 2 nhánh mạng này được map vào một vector có số chiều nhỏ trong không gian tiềm ẩn với công thức:



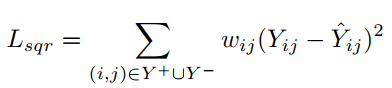
* Cuối cùng đo độ tương quan cosine: 
* **Hàm mất mát:**

+ Hàm mất mát tổng quát

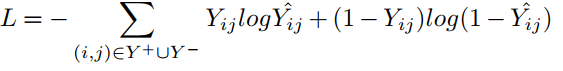


+ Sử dụng hàm mục tiêu là pointwise (Có thể cải thiện thành pointwise).

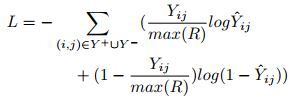
+ Hàm square loss dùng cho explicit rating, không dùng square loss vì không hiệu quả với implicit feedback.



+ Hàm mất mát binary cross-entropy hiệu quả với implicit feedback.



+ Vì bài báo dùng cả implicit feedback và explicit rating nên tác giả đề xuất một hàm mất mát mới kết hợp cả hai hàm mất mát phía trên:

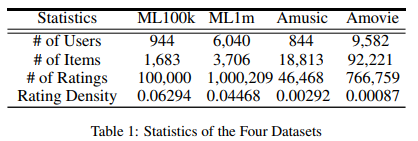


Vì độ đo binary cross entropy có thể dẫn đến là số âm, nên dùng công thức sau để không âm.



Với

* Bài báo sử dụng 4 tập dữ liệu như hình:

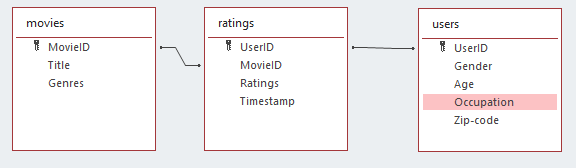


* Độ đo được sử dụng là Hit ratio và NDCG (*Normalized Discounted Cumulative Gain)*

1. **Dữ liệu**

Thư mục dữ liệu Movielens-1m bao gồm 3 tập tin: ratings.dat, users.dat, movies.dat; chứa thông tin của 1,000,209 ratings với khoảng 3900 bộ phim được bình chọn bởi 6040 người vào năm 2000.

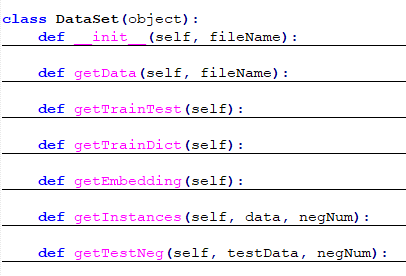
* Tập tin users.dat: gồm các thuộc tính UserID, Gender, Age, Occupation, Zip-code.
* Tập tin movies.dat gồm các thuộc tính MovieID, Title, Genres.
* Tập ratings.dat: gồm các thuộc tính UserID, MovieID, Rating, Timestamp.



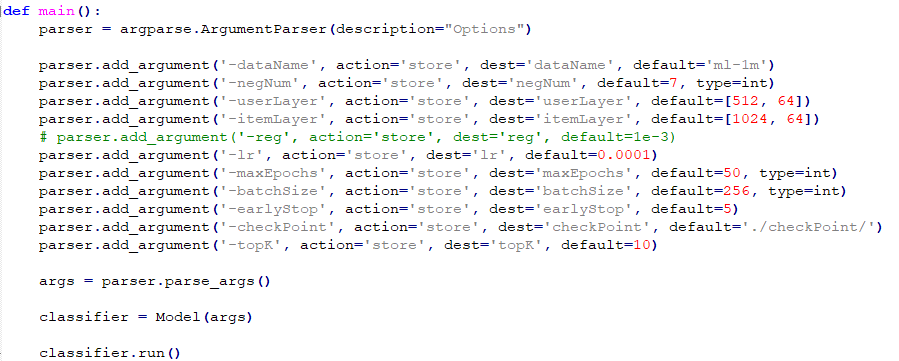
1. **Mã nguồn**

Bài báo có nhiều nguồn github, sinh viên chạy github chính thống của tác giả: <https://github.com/RuidongZ/Deep_Matrix_Factorization_Models>

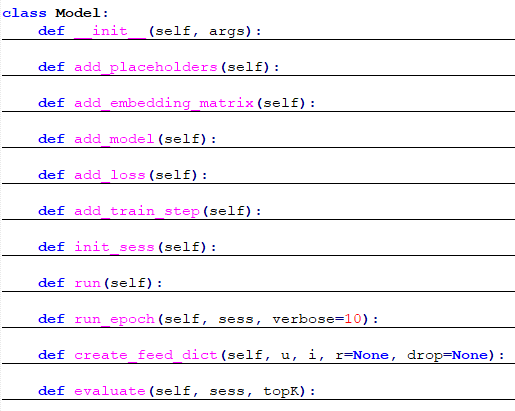
Có 2 file mã nguồn: dataset.py và model.py.

* **dataset.py:** tập tin này đọc file dữ liệu, gán vào các biến train, test, và shape của dữ liệu. Tập tin này có một lớp gồm 6 hàm: 
* **def getData(self, fileName):** nhận vào file, đọc file này rồi lưu vào các biến user, movie, score, time, và user\_size, item\_size.
* **def getTrainTest(self):** Hàm này chia thành tập train và test, nếu user dòng i khác user dòng i+1 thì thêm data dòng i vào tập test.
* **def getTrainDict(self):** tạo một từ điển dữ liệu với key =<user, movie> và key=<score>.
* **def getEmbedding(self):** Tạo một ma trận tương tác user, item.
* **def getInstances(self, data, negNum):** Phát sinh negNum dòng dữ liệu mới với ratings = 0. Trả về user, item, rating.
* **def getTestNeg(self, testData, negNum):** Tương tự hàm getIntances nhưng không trả về rating.
* **model.py:** tập tin này chứa mô hình dữ liệu, các tham số chạy mô hình và chạy chương trình. Tập tin này gồm 1 hàm main, và một lớp với 11 hàm.

+ Hàm main chứa các tham số để chạy mô hình (như hình)



+ Lớp Model:



* **def \_\_init\_\_(self, args):** Hàm khởi tạo lấy các tham số của mô hình.
* **def add\_placeholders(self):** Xác định kiểu dữ liệu của các biến rating, user, movie.
* **def add\_embedding\_matrix(self):** Thêm 2 ma trận tương tác đã embedding, ma trận item-user và ma trận user-item.
* **def add\_model(self):** Thêm 2 ma trận ở hàm **add\_embedding\_matrix(self)**  là đầu vào của mạng. Khởi tạo trọng số, nhân trọng số với các tầng, với activation function = relu.
* **def add\_loss(self):** tính độ lỗi cross-entropy loss.
* **def add\_train\_step(self):** optimizer= ‘adam’, minimize by loss.
* **def init\_sess(self):** Khởi tạo session
* **def run(self):** Huấn luyện dữ liệu và ghi nhận độ đo HR, NDCG tốt nhất.
* **def run\_epoch(self, sess, verbose=10):** Chạy epoch và cho ra độ loss của từng epoch.
* **def evaluate(self, sess, topK):** Gồm 2 hàm đánh giá HitRatio, và NDCG.