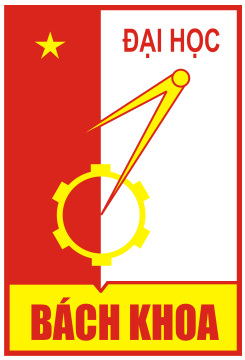
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



BÁO CÁO CUỐI KỲ

Mô hình ngẫu nhiên và ứng dụng

Tích hợp mạng đối nghịch tạo sinh cho hệ thống khuyến nghị dựa trên mô hình học tăng cường

GVHD: TS. Nguyễn Thị Ngọc Anh

Sinh viên:	Nguyễn Anh Minh	20194117
	Nguyễn Đình Nhật	20190080
	Lê Đức Tài	20195163

Hà Nội tháng 7 năm 2022

Mục lục

1 Giới thiệu

1.1 Tổng quan đề tài

Mua sắm online đang trở thành xu hướng thương mại trong thời đại công nghệ 4.0 hiện nay. Các nền tảng mạng xã hội như Facebook, Tweet hay các nền tảng nghe nhạc, xem video trực tuyến như Youtube và Tiktok cũng đang càng ngày càng phát triển. Các hệ thống khuyến nghị khách hàng cũng từ đó được xây dựng và ngày càng phát triển. Một hệ thống khuyến nghị tốt sẽ tối đa hóa độ hài lòng của khách hàng khi sử dụng dịch vụ đem lại lợi nhuận lớn cho công ty chủ quản. Trong báo cáo này, nhóm tác giả sẽ trình bày về cách tích hợp mạng đối nghịch tạo sinh cho hệ thống khuyến nghị dựa trên mô hình học tăng cường, kết hợp việc cài đặt và kiểm thử mô hình với dữ liệu thực tế để cho thấy độ hiệu quả của mô hình.

1.2 Cấu trúc bài báo cáo

Nội dung của báo cáo được chia thành 9 phần như sau:

- **Giới thiệu**.
- **Mạng đối nghịch tạo sinh (GAN)** trình bày các lý thuyết cơ bản của mạng GAN.
- **Hệ thống khuyến nghị (RS)** trình bày các lý thuyết cơ bản của RS.
- **Học tăng cường (RL)** trình bày lý thuyết cơ bản về RL.
- **Khảo sát về ứng dụng của RL trong việc xây dựng RS**.
- **Tổng hợp về tích hợp GAN cho RL-based RS**
- **Phương pháp luận** Trình bày chi tiết cách tích hợp mạng GAN cho RL-based RS và cách thiết kế mạng Cascading Q-network để học chính sách hiệu quả.
- **Kết quả thực nghiệm:** Cài đặt và đánh giá kết quả mô hình khi áp dụng cho bộ dữ liệu thực tế MovieLen.
- **Kết luận**.

Trong báo cáo này, nhóm tác giả sử dụng một số tên viết tắt cho các thuật ngữ như sau:

RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
RS	Recommendation System
GAN	Generative Adversarial Network
MDP	Makov Decision Process

Bảng 1: Bảng thuật ngữ viết tắt

Dù đã rất cố gắng xong báo cáo này vẫn không tránh khỏi những hạn chế cần khắc phục. Vì vậy, tác giả rất mong quý thầy cô đưa ra những ý kiến góp ý bổ ích để đồ án này tiếp tục được phát triển và có những kết quả mới tốt hơn.

Lời cảm ơn

Báo cáo này được thực hiện và hoàn thành tại Trường Đại học Bách Khoa Hà Nội, nằm trong nội dung học phần *Mô hình ngẫu nhiên và ứng dụng* của kì học 2021-2.

Tác giả xin được dành lời cảm ơn chân thành tới TS. Nguyễn Thị Ngọc Anh, là giảng viên đã trực tiếp hướng dẫn và khuyến nghị cho tác giả đề tài rất thú vị này, đồng thời thầy cũng đã giúp đỡ tận tình và có những đóng góp bổ ích để tác giả có thể hoàn thành báo cáo này một cách tốt nhất.

2 Mạng đối nghịch tạo sinh (GAN)

Trong phần này, tác giả sẽ trình bày khái niệm mạng đối nghịch tạo sinh, ứng dụng của mô hình trong các mô hình Deep Learning nói chung và mô hình học sâu tăng cường áp dụng cho hệ khuyến nghị nói riêng. Các nội dung trình bày bao gồm ý tưởng phát triển, mô hình tối ưu và ví dụ mô tả.

2.1 Cơ sở lý thuyết

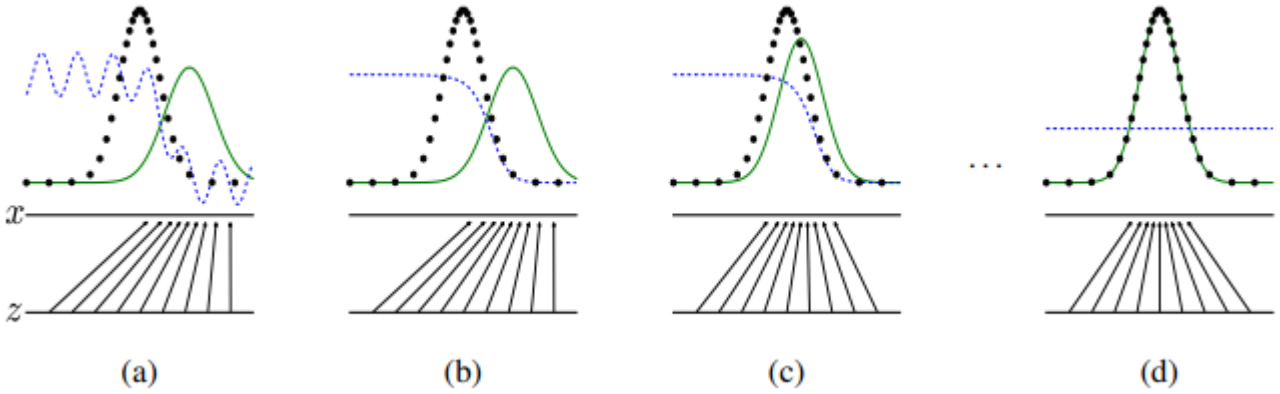
Mạng đối nghịch tạo sinh (*Generative Adversarial Nets*, GAN) là phương pháp được phân loại vào một trong các Mô hình tạo sinh (*Generate Model*). Mô hình này được áp dụng trong trường hợp bộ dữ liệu thực tế quá ít, cần tạo sinh thêm các mẫu mới có chất lượng tương tự các mẫu ban đầu để có thể huấn luyện các thuật toán. Mô hình được mô phỏng dựa trên quá trình thực hiện hai mô hình con: mô hình tạo sinh G (*Generator*) và mô hình phân biệt D (*Discriminator*). Để có thể mô tả một cách trực quan sự đối lập của hai mô hình này, ta có thể lấy một ví dụ trên thực tế về việc tạo ra tiền giả. Dữ liệu ban đầu là đồng tiền được nhà nước phát hành, G đại diện cho những kẻ xấu cố gắng tạo tiền giả thông qua việc bắt chước các chi tiết trên tiền thật. Trong khi đó, D mô phỏng cho cảnh sát, tìm cách phân biệt tiền ảo so với tiền thật bằng các chi tiết khác nhau trên hai đối tượng. Mô hình tạo sinh G nắm bắt cách phân bố của dữ liệu, trong khi mô hình phân biệt D ước lượng xác suất một mẫu được lấy từ dữ liệu huấn luyện hay là từ G.

2.2 Mạng đối nghịch

Mô hình mạng đối thủ được áp dụng dễ nhất trong trường hợp cả hai mô hình đều là perceptrons nhiều lớp. Để nghiên cứu về phân bố của mô hình tạo p_g trên tập dữ liệu x, ta định nghĩa biến nhiễu $p_z(z)$, và một song ánh tới không gian dữ liệu $G(z, \theta_g)$, với G là hàm tạo sinh đại diện cho perceptron nhiều lớp với tham số θ_g. Ta cũng định nghĩa perceptron nhiều lớp khác là $D(x, \theta_d)$ với đầu ra là một véc tơ vô hướng. $D(x)$ đại diện cho xác suất mà x được lấy từ dữ liệu thật thay vì mô hình sinh p_g. Ta huấn luyện mô hình D để cực đại hóa xác suất D gán đúng nhán cho cả hai mẫu lấy từ dữ liệu gốc và mẫu lấy từ G (hay trong ví dụ trên, ta huấn luyện cảnh sát làm sao phân biệt được đúng nhất tiền giả so với tiền thật), đồng thời cực tiểu hóa hàm $\log(1 - D(G(z)))$ (tương ứng với việc huấn luyện kẻ xấu có thể làm giả tiền sao cho cảnh sát khó phân biệt nhất có thể).

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{1}$$

Trên thực tế, phương trình (1) không cung cấp đủ thông tin về hướng (*gradient*) cho mô hình G để huấn luyện tốt. Trong các nghiên cứu trước đây, khi mô hình G không tốt, D có thể phản hồi kết quả các mẫu cần phân biệt với độ tin cậy cao, bởi vì dễ dàng phân biệt chúng với tập ban đầu. Trong trường hợp này, $\log((1 - D(G(z)))$ chiếm ưu thế. Thay vì việc huấn luyện mô hình G bằng cách cực tiểu hóa hàm $\log(1 - D(G(z)))$ thì ta có thể cực đại hóa hàm $\log(D(G(z)))$. Hàm mục tiêu



Hình 1: So sánh mô hình mạng GAN.

cho kết quả trên các điểm giống nhau đối với G và D tuy nhiên lại cung cấp gradient mạnh hơn các kết quả học trước đây.

Trong hình 1, tác giả mô tả mô hình mạng đối nghịch tạo sinh được huấn luyện thông qua việc mô phỏng cập nhật phân phối của discriminative (D,xanh, đường nét đứt) sao cho nó có thể phân biệt được giữa mẫu ban đầu (đen, nét chấm) p_x với các mẫu được tạo sinh p_G. Đường thẳng ngang bên dưới là miền z được tạo mẫu, trong trường hợp này các mẫu đều đồng nhất. Đường thẳng ngang bên trên là một phần của miền x. Đường mũi tên hướng lên trên chỉ ánh xạ $x = G(z)$ thể hiện phân phối không đồng nhất p_g trong các mẫu khác nhau. G thu hẹp ở những nơi có mật độ cao và mở rộng ở những vùng có mật độ p_g thấp.

- Hình (a) xem xét trường hợp cặp đối nghịch gần điểm hội tụ: p_g tương tự p_{data} và D là một phần các phân loại chính xác một phần
- Hình (b), trong vòng lặp của thuật toán D được huấn luyển để phân biệt mẫu từ dữ liệu, hội tụ tới $D^* = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_g(x)}$
- Hình (c) Sau khi cập nhật G, gradient của D định hướng $G(z)$ tới miền có xu hướng được phân loại đúng theo dữ liệu
- Hình (d) Sau một vài bước huấn luyện, nếu G và D có đủ thời gian, chúng sẽ hội tụ tới điểm mà cả hai không thể cải thiện hơn vì $p_g = p_{\text{data}}$. Discriminator không thể phân biệt được giữa hai tập tạo sinh và tập ban đầu. Khi đó thì $D(x) = \frac{1}{2}$.

3 Hệ thống khuyến nghị (RS)

Trong phần này, tác giả sẽ trình bày một số kiến thức liên quan tới Hệ thống khuyến nghị như khái niệm về Hệ thống khuyến nghị, Utility Matrix và hai nhóm chính của Hệ thống khuyến nghị là: Khuyến nghị dựa trên nội dung và Hệ khuyến nghị dựa trên lọc cộng tác...

3.1 Giới thiệu chung

Hệ thống khuyến nghị (Recommendation System/Recommender System - RS) là một dạng của hệ thống lọc thông tin, nó được sử dụng để dự đoán sở thích hay xếp hạng mà người dùng(user) có thể dành cho một thông tin (item) nào đó mà họ chưa xem xét tới trong quá khứ (item có là bài hát, bộ phim, video, sách, báo, v.v...).

Hệ thống khuyến nghị được ứng dụng trong nhiều lĩnh vực khác nhau như thương mại điện tử, truyền thông giải trí, ý tế và giáo dục, v.v...Chính vì có những ứng dụng rộng rãi trong cuộc sống nên RS mở ra nhiều tiềm năng trong nghiên cứu cũng như trong xây dựng các thống thực tế.

Hệ thống khuyến nghị được chia thành hai nhóm chính:

- Hệ khuyến nghị dựa trên nội dung chính (Content-based system): đánh giá đặc tính của sản phẩm được khuyến nghị. Ví dụ: nếu người dùng Netflix đã xem nhiều bộ phim thể loại tình cảm, thì hệ thống khuyến nghị một bộ phim trong cơ sở dữ liệu có thể loại phim tình cảm cho người dùng này.
- Hệ khuyến dựa trên lọc cộng tác (Collaborative filtering): hệ thống khuyến nghị sản phẩm dựa trên các tương quan giữa người dùng và/hoặc sản phẩm. Ví dụ: người dùng A, B, C đều thích những bộ phim do Tom Holland thủ vai. Ngoài ra, hệ thống còn biết rằng người B, C cũng thích những bộ phim do Robert Downey thủ vai nhưng chưa có thông tin về việc liệu người dùng A có thích bộ phim do Robert Downey thủ vai hay không. Dựa trên thông tin của những người dùng tương tư B và C, hệ thống có thể dự đoán A cũng thích những bộ phim do Robert Downey thủ vai. Từ đó khuyến nghị các bộ phim mà do Robert Downey thủ vai.

3.2 Utility Matrix

3.2.1 Utility Matrix

Trong hệ thống khuyến nghị có hai thực thể chính là: người dùng(user) và sản phẩm(item). Môi người dùng sẽ có mức độ quan tâm tới từng sản phẩm khác nhau. Mức độ quan tâm này, nếu đã biết trước được gán cho một giá trị ứng với mỗi cặp người dùng-sản phẩm (user-item). Giả sử mức độ quan tâm được đo bằng giá trị người dùng(user) đánh giá (rate) cho sản phẩm (item), ta gọi giá trị này *rating*. Tập hợp các giá trị rating, bào gồm những giá trị chưa biết cần dự đoán tạo nên ma trận gọi là *utility matrix*.

Ví dụ: Xét ma trận Utility Matrix, người dùng đánh giá các bộ phim trên thang điểm 1-5, với 5 xếp hạng cao nhất. Các cặp người dùng- sản phẩm để trống có, nghĩa là người dùng không đánh giá bộ phim. Ta ký hiệu tên phim HP1, HP2, HP3 cho bộ phim *Harry Potter* I, II và II, IM1,IM2, IM2 cho bộ phim *Iron Man* I, II, III. Người dùng được ký hiểu bằng các chữ cái viết hoa từ A đến D.

	HP1	HP2	HP3	IM1	IM2	IM3
A	4			5	1	
B	5	5	4			5
C				2	4	
D		3				3

Bảng 2: Ví dụ utility matrix với hệ thống khuyến nghị phim.

Mục tiêu của hệ thống khuyến nghị là dự đoán các ô trống trong utility matrix. Ví dụ, người dùng A có thích IM3 hay không? Có rất ít thông tin từ utility matrix . Vì vậy, chúng ta cần thiết kế hệ thống khuyến nghị để tính đến các thuộc tính của phim như nhà sản xuất, đạo diễn, các ngôi sao. Từ đó chúng ta có thể ghi nhận sự giống nhau giữa IM2 và IM3, và sau đó kết luận rằng vì A không thích IM2 nên A cũng không thích IM3. Ngoài ra, với nhiều dữ liệu hơn,chúng ta có thể đưa ra xu hướng đánh giá xếp cả IM2 và IM3 tương tự. Do đó, chúng ta có thể kết luận rằng A cũng sẽ đánh giá IM3 thấp, tương tự như đánh giá IM2 của A.

Thông thường, có rất nhiều người dùng *(user)* và sản phẩm *(item)* trong hệ thống mà mỗi người dùng thường chỉ đánh giá một lượng rất nhỏ các sản phẩm, thậm chí có những người dùng không đánh giá sản phẩm nào. Vì vậy, lượng ô để trống của **utility matrix** trong các bài toán đó thường rất lớn, và lượng các ô được điên là một lượng rất nhỏ.

Chúng ta thấy rằng càng nhiều ô được điền thì độ chính xác của hệ thống sẽ càng được cải thiện. Vì vậy, các hệ thống luôn hỏi người dùng về sự quan tâm của họ tới sản phẩm và muốn người dùng đánh giá càng nhiều sản pẩm càng tốt. Việc đánh giá các sản phẩm, không những giúp các người dùng khác biết được chất lượng sản phẩm của sản phẩm mà còn giuups hệ thống biết được sở thích của người dùng, qua đó có thể đưa ra các chính sách marketing.

3.2.2 Xây dựng Utility Matrix

Nếu không có Utility Matrix, hầu như không thể giới thiệu các sản phẩm tới người dùng. Tuy nhiên, việc thu thập dữ liệu để xây dựng Utility Matrix thường rất khó khăn. Có hai cách tiếp cận để xác định biến giá trị *rating* cho mỗi cặp người dùng-sản phẩm trong Utility Matrix:

1. Chúng ta có thể yêu cầu người dùng đánh giá sản phẩm. Các sàn thương mại điện tử thường yêu cầu khách hàng đánh giá sản phẩm của họ sau khi trải nghiệm sản phẩm của họ. Rất nhiều hệ thống cũng hoạt động tương tự như vậy. Tuy nhiên, cách tiếp cận này còn hạn chế, vì người dùng không sẵn sàng cung cấp đánh giá phản hồi. Và nếu có, đó có thể là những đánh giá thiên lệch bởi những người sẵn sàng đánh giá.

2. Chúng ta có thể đánh giá từ hành vi của người dùng. Nếu người dùng mua sản phẩm tại Shoppe, xem phim trên Youtube hoặc đọc tin tức thì người dùng có thể được cho là "thích"sản

phẩm này. Với cách xây dựng này thì ta được một ma trận với các thành phần là 1 và 0, với 1 thể hiện người dùng thích sản phẩm, 0 thể hiện chưa có thông tin. Trong trường hợp này, 0 không có nghĩa là đánh giá thấp hơn 1, nó chỉ có nghĩa là người dùng chưa cung cấp thông tin. Nói cách khác, chúng ta có thể suy ra sự quan tâm từ hành vi khác hơn là mua hàng. Ví dụ: nếu một khách hàng của Shopee xem thông tin về một mặt hàng, chúng ta có thể suy ra rằng họ quan tâm đến mẫ hàng đó, ngay cả khi họ không mua.

3.3 Hệ khuyến nghị dựa trên nội dung

Trọng một hệ thống dựa trên nội dung, chúng ta cần tạo cho mỗi sản phẩm một bộ hồ sơ (Profiles), đó là một bản ghi hoặc tập hợp các bản ghi đại diện cho các thuộc tính quan trọng của sản phẩm. Trong những trường hợp đơn giản, hồ sơ bao gồm một số thuộc tính của sản phẩm dễ dàng phát hiện ra. Ví dụ: Một số thuộc tính của một bộ phim có thể được sử dụng trong Hệ thống khuyến nghị:

1. Dàn diễn viên của phim. Một số người xem thích phim có diễn viên yêu thích.
2. Đạo diễn. Một số người xem có thích xem các phim nhất định của đạo diễn.
3. Năm sản xuất bộ phim. Một số khán giả thích xem phim cũ hơn, những người khác chỉ xem các bản phát hành mới nhất.
4. Thể loại phim. Một số người xem chỉ thích phim hài, phim tình cảm.

Có nhiều thuộc tính khác của phim cũng có thể được sử dụng. Ngoại trừ Thể loại khó định nghĩa, các yếu tố khác đều được xác định rõ ràng.

3.4 Hệ thống khuyến nghị dựa trên lọc cộng tác

Hệ thống phân tích và tổng hợp các điểm số đánh giá của các đối tượng, nhận ra có sự tương đồng giữa những người sử dụng trên sơ sở các điểm số đánh giá của họ và tạo ra các khuyến nghị dưa trên so sánh này.

Một số giải thuật thường được sử dụng:

- Phương pháp láng giềng (Neighborhood-based): giải thuật dự trên dữ liệu quá khứ của người dùng "tương tự -similarity"(user-based approach), hoặc là dựa trên dữ liệu quá khứ của những item "tương tự"(item-based approach).
- Dựa trên mô hình (Model-based): Nhóm này liên quan đến việc xây dựng mô hình dự đoán dựa trên dữ liệu thu thập được trong quá khứ. như mô hình Bayesian, các mô hình nhân tố tiềm ẩn (latent factor models): trong đó kỹ thuật phân rã ma trận là một điển hình.

3.5 Nhận xét, đánh giá

Một trong những khuyết điểm của phương pháp lọc theo nội dung là khó khắn trong việc thu thập thông tin, đòi hỏi chúng ta phải thu thập rất nhiều thông tin về các mục tin tương tự. Chính việc xác định xem mục tin nào là tương tự với mục tin hiện tại đòi hỏi chúng ta phải thu thập và phân tích, xử lý toàn bộ các mục tin trong cơ sở dữ liệu. Trong khi phần lớn các mô hình dựa trên lọc cộng tác không cần quá nhiều thông tin, chỉ cần 3 thông tin (user id, item id, rate) là có thể hoạt động tốt. Do vậy khuynh hướng hiện nay đa phần sử dụng phương pháp lọc cộng tác để xây dựng các hệ thống khuyến nghị.

4 Học tăng cường (RL)

Trong phần này, tác giả sẽ trình bày các kiến thức liên quan tới Học tăng cường như mô hình toán học của Học tăng cường là quá trình quyết định Markov MDP, các khái niệm trong Học tăng cường như phần thưởng, chính sách, các cách xấp xỉ cho chính sách cũng như cập nhật chính sách để đạt phần thưởng tối đa,... Các kiến thức trong phần này sẽ được tham khảo chính tại sách [4].

4.1 Quá trình quyết định Markov (MDP)

Một quá trình quyết định Markov MDP được định nghĩa thông qua 5 thành phần như sau $(\mathcal{S}, \mathcal{A}, P(\cdot \mid \cdot, \cdot), R(\cdot, \cdot\cdot\cdot), \gamma)$, trong đó:

- $\mathcal{S}$ là tập hữu hạn biểu thị tất cả các trạng thái của quá trình. Ký hiệu trạng thái tại thời điểm t của quá trình là S_t.
- $\mathcal{A}$ là tập hữu hạn các hành động. Với mỗi trạng thái $s \in \mathcal{S}$ ta có tập hữu hạn các hành động tại s là $\mathcal{A}_s$. Ký hiệu hành động tại thời điểm t là A_t.
- $P(s' \mid s, a) = P(S_{t+1} = s' \mid S_t = s, A_t = a)$ là xác suất quá trình chuyển sang trạng thái s' tại thời điểm $t+1$ với $S_t = s$ và $A_t = a$.
- $R(s', a, s)$ là phần thưởng khi quá trình chuyển từ trạng thái s chọn hành động a và chuyển sang trạng thái s'.
- $0 \leq \gamma \leq 1$ là độ chiết khấu giữa phần thưởng hiện tại và phần thưởng trong tương lai.

Tại mỗi thời điểm t, quá trình đang ở trạng thái S_t và đưa ra quyết định chọn hành động $A_t \in \mathcal{A}_{S_t}$. Quá trình sẽ chuyển đến trạng thái S_{t+1} với xác suất chuyển $P(S_{t+1} \mid S_t, A_t)$ và được nhận phần thưởng tương ứng $R_t = R(S_{t+1}, A_t, S_t)$.

Ta có định nghĩa một chính sách π là một phương án lựa chọn hành động mà quá trình tuân theo. Chính sách π gồm hai loại sau:

1. **Xác định**: khi quá trình đang ở trạng thái bất kì $s \in \mathcal{S}$, hành động được chọn tiếp theo sẽ là $\pi(s)$;
2. **Ngẫu nhiên**: xác suất lựa chọn hành động $a \in \mathcal{A}_s$ là $\pi(a|s)$.

Mục tiêu đặt ra cho MDP: Cho một quá trình Markov quyết định $(\mathcal{S}, \mathcal{A}, P(\cdot \mid \cdot, \cdot), R(\cdot, \cdot, \cdot), \gamma)$ hãy đi xây dựng chính sách π nhằm tối ưu các phần thưởng thu được trong thời gian dài hạn (còn gọi là *lợi nhuận (return)*), trong đó công thức của lợi nhuận G_t với chiết khấu γ là:

$$G_t = R_t + \gamma R_{t+1} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}.$$

G_t là tổng tất cả các phần thưởng nhận được kể từ trạng thái s_t đến tương lai, với $0 < \gamma < 1$ thì các phần thưởng càng xa hiện tại sẽ càng bị *giảm giá (discount)* nhiều và ngược lại.

Định nghĩa 4.1 *Xét chuỗi biến ngẫu nhiên* $\{S_t\}_{t\in\mathbb{N}}$*, trạng thái* s *được gọi là có tính Markov nếu* $\forall s' \in \mathcal{S}$ *thì*

$$P\left(S_{t+1} = s' \mid S_t = s\right) = P\left(S_{t+1} = s' \mid h_{t-1}, S_t = s\right)$$

với mọi khả năng các sự kiện xảy ra trong quá khứ $h_{t-1} = \{S_1, \ldots, S_{t-1}, A_1, \ldots, A_{t-1}, R_1, \ldots, R_{t-1}\}$.

Trong MDP thì mọi trạng thái đều được giả sử có tính Markov, tức là trạng thái s biểu thị đầy đủ các thông tin liên quan từ các trạng thái trong quá khứ.

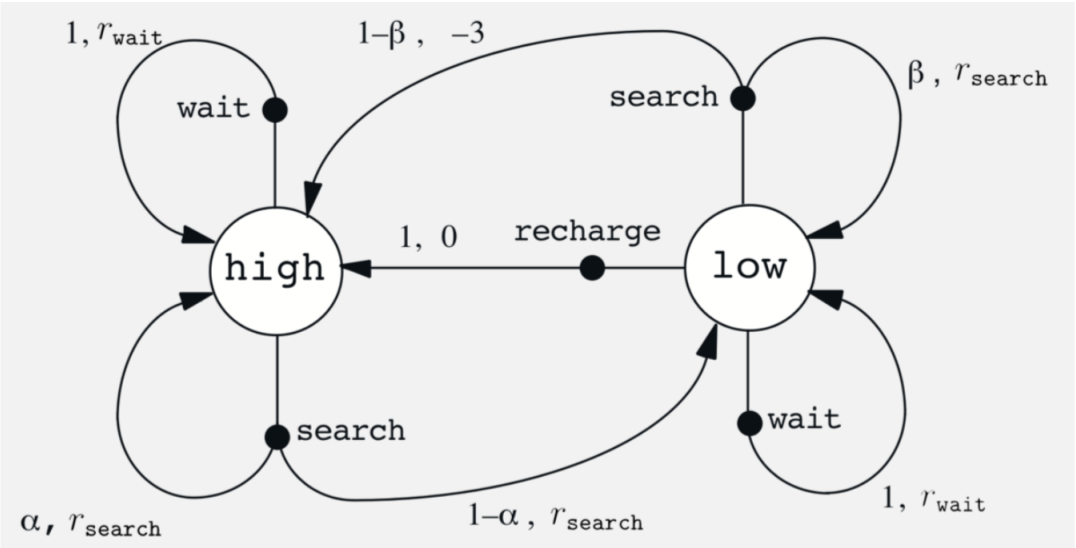
Ví dụ về MDP

Bài toán đặt ra là cần điều khiển một robot với nhiệm vụ thu gom vỏ lon cho công ty. Robot được thiết kế với cảm biến để nhận diện vỏ lon và có tay cầm để nhặt vỏ lon và bỏ vào thùng rác và có một lượng pin dự trữ để hoạt động. Robot sẽ hoạt động ở 2 trạng thái $\mathcal{S} = \{$ high, low $\}$ với trạng thái *high* là khi Robot có ngưỡng pin trên ngưỡng cho phép và *low* là ngược lại . Ở trạng thái *low*, robot sẽ quyết định 1 trong 3 hành động: (1) chủ động tìm kiếm vỏ lon trong 1 khoảng chu kỳ, (2) tiết kiệm năng lượng và đứng im tại chỗ đợi cho có người đưa vỏ lon cho Robot, hoặc (3) trở về nhà và sạc lại năng lượng. Khi ở trạng thái *high* Robot chỉ thực hiện 2 hành động (1) và (2). Tập hành động tương ứng là $\mathcal{A}($ high $) = \{$ search, wait $\}$ và $\mathcal{A}($ low $) = \{$ search, wait, recharge $\}$

s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	r_{search}
high	search	low	$1-\alpha$	r_{search}
low	search	high	$1-\beta$	-3
low	search	low	β	r_{search}
high	wait	high	1	r_{wait}
high	wait	low	0	r_{wait}
low	wait	high	0	r_{wait}
low	wait	low	1	r_{wait}
low	recharge	high	1	0
low	recharge	low	0	0

Hình 2: Ma trận xác suất chuyển

Xác suất chuyển giữa hai trạng thái và phần thưởng tương ứng các hành động được miêu tả dựa vào Hình (2). Phần thưởng sẽ bằng 0 trong phần lớn thời gian hoạt động nhưng sẽ có giá trị dương khi robot thu được vỏ lon cũng như phần thưởng âm (bị phạt) khi pin của robot cạn kiệt. Tương ứng mô hình hoạt động của Robot được miêu tả tại Hình (3).

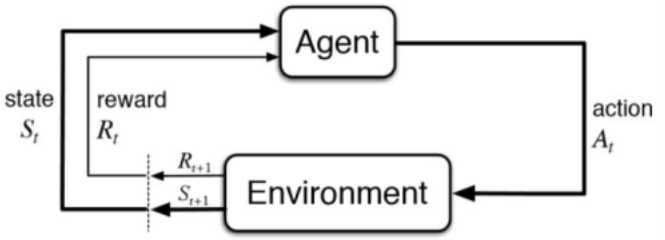


Hình 3: Mô hình chuyển trạng thái của Robot

4.2 Mô hình hóa cho RL

MDP là biểu diễn mô hình toán học của Học tăng cường, ứng với các thành phần của MDP ta định nghĩa một số thuật ngữ được sử dụng trong RL như sau

- *Agent* (Tác tử): Tác tử quan sát môi trường và thực hiện các hành động tương ứng. Ta cần điều khiển tác tử sao cho đạt mục tiêu đề ra là tối đa hóa lợi nhuận.
- *Environment* (Môi trường): là không gian mà tác tử tương tác.
- *Policy* (Chính sách): Tác tử sẽ thực hiện các hành động theo chính sách để đạt được mục đích cụ thể với từng nhiệm vụ khác nhau.
- *Reward* (Phần thưởng): phần thưởng tương ứng từ môi trường mà tác tử nhận được khi thực hiện một hành động.
- *State* (Trạng thái): trạng thái của môi trường mà tác tử nhận được.
- *Episode* (tập): một chuỗi các trạng thái và hành động cho đến trạng thái kết thúc : $s_1, a_1, s_2, a_2, ...s_T, a_T$
- *Accumulative Reward* (phần thưởng tích lũy/lợi nhuận): tổng phần thưởng tích lũy từ trạng thái đầu tiên đến trạng thái cuối cùng. Như vậy, tại s_t, tác tử tương tác với môi trường và thực hiện hành động a_t, dẫn đến trạng thái mới s_{t+1} và nhận được phần thưởng tương ứng r_{t+1}. Quá trình lặp diễn ra như vậy cho đến trạng thái cuối cùng s_T.



Hình 4: Mô hình học tăng cường

4.2.1 Chính sách và hàm giá trị (value function)

Chiến sách là cốt lõi của tác tử trong việc xác định hành vi. Các thuật toán RL sẽ chỉ định cách mà tác tử thay đổi chính sách dựa vào chính những trải nghiệm (các bước đi và hành động trong quá khứ) của nó. Trong một số trường hợp, chiến lược có thể là một hàm hoặc bảng tra cứu đơn giản. Trong một số trường hợp khác, chiến lược có thể liên quan đến tính toán mở rộng, ví dụ như quá trình tìm kiếm.

Giá trị của trạng thái s tuân theo chính sách π, kí hiệu là $V^\pi(s)$ được định nghĩa kỳ vọng lợi nhuận khi bắt đầu từ trạng thái s và đi theo chính sách π:

$$V^\pi(s) \doteq \mathbb{E}_\pi \left[G_t \mid S_t = s \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right], \text{ với mọi } s \in \mathcal{S}$$

Chú ý rằng ta luôn có $V^\pi(s) = 0$ khi s là trạng thái dừng (terminal state). Hàm $V^\pi(s)$ được xác định như trên được gọi là hàm giá trị trạng thái (state-value function) của chính sách π.

Tương tự, ta định nghĩa giá trị của hành động a ở trạng thái s theo chính sách π, kí hiệu là $Q^\pi(s, a)$, là kỳ vọng lợi nhuận khi bắt đầu từ trạng thái s, chọn hành động a và sau đó tuân theo chính sách π:

$$Q^\pi(s, a) \doteq \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right]$$

Ta gọi Q^π là hàm giá trị hành động của policy π (action-value function) của chính sách π).

Dựa theo công thức xác suất đầy đủ ta có quan hệ giữa $V^\pi(s)$ và $Q^\pi(s, a)$:

$$V^\pi(s) = \sum_{a \in A} \pi_\theta(s, a) Q^\pi(s, a),$$

Mặt khác, với mọi $s \in \mathcal{S}$, ta có:

$$\begin{aligned} V^{\pi}(s) &\doteq \mathbb{E}_{\pi}\left[G_t \mid S_t = s\right] \\ &= \mathbb{E}_{\pi}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\ &= \sum_{a} \pi(a \mid s) \sum_{s'} \sum_{r} p\left(s', r \mid s, a\right)\left[r + \gamma \mathbb{E}_{\pi}\left[G_{t+1} \mid S_{t+1} = s'\right]\right] \\ &= \sum_{a} \pi(a \mid s) \sum_{s', r} p\left(s', r \mid s, a\right)\left[r + \gamma v_{\pi}\left(s'\right)\right] \quad \textit{(Bellman equation)} \end{aligned} \tag{2}$$

Phương trình (2) được gọi là phương trình Bellman và sẽ là phương trình cốt lõi cho việc cập nhật các hàm giá trị trong các thuật toán tối ưu chính sách của RL.

4.2.2 Chính sách tối ưu (Optimal policies) và hàm giá trị tối ưu (optimal value functions)

Mục đích của RL là tìm ra một chính sách giúp đạt được nhiều phần thưởng về mặt dài hạn. Với quá trình quyết định Markov hữu hạn, ta có thể định nghĩa chính xác chính sách tối ưu theo cách sau đây:

- Các hàm giá trị xác định thứ tự từng phần đối với các chính sách.
- Chính sách π được định nghĩa là tốt hơn hoặc chất lượng hơn một chính sách π' nếu lợi nhuận kỳ vọng của nó lớn hơn hoặc bằng π' cho tất cả các trạng thái.
- $\pi \geq \pi'$ khi và chỉ khi $V^{\pi}(s) \geq V^{\pi'}(s)$ với mọi $s \in \mathcal{S}$.

Luôn có ít nhất một chính sách tốt hơn hơn hoặc bằng tất cả các chính sách khác và được định nghĩa là chính sách tối ưu. Có thể có nhiều hơn một chính sách tối ưu, ký hiệu bằng π_*. Ứng với π_*, tác tử sẽ thực hiện các hành động và dẫn tới có một hàm giá trị trạng thái tối ưu (optimal state-value function), được ký hiệu là V_*, và được định nghĩa bởi:

$$V_*(s) \doteq \max_{\pi} V^{\pi}(s)$$

với mọi $s \in \mathcal{S}$.

Các chính sách tối ưu sẽ tương ứng hàm giá trị hành động tối ưu (optimal action-value function), kí hiệu là Q_* và được xác định bởi:

$$Q_*(s, a) \doteq \max_{\pi} Q^{\pi}(s, a)$$

với mọi $s \in \mathcal{S}$ và $a \in \mathcal{A}(s)$. Với mỗi cặp trạng thái-hành động (s, a), Q_* cho ta giá trị lợi nhuận kỳ vọng lớn nhất khi thực hiện hành động a ở trạng thái s và các hành động sau đó tuân theo chính sách tối ưu. Do đó, ta có thể viết Q_* ở dạng sau:

$$Q_*(s, a) = \mathbb{E}\left[R_{t+1} + \gamma V_*\left(S_{t+1}\right) \mid S_t = s, A_t = a\right]$$

Dựa vào phương trình Bellman và định nghĩa của $V_*(s)$, ta có:

$$\begin{aligned} V_*(s) &= \max_{a \in \mathcal{A}(s)} Q^{\pi_*}(s,a) \\ &= \max_a \mathbb{E}_{\pi_*}\left[G_t \mid S_t = s, A_t = a\right] \\ &= \max_a \mathbb{E}_{\pi_*}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a\right] \\ &= \max_a \mathbb{E}\left[R_{t+1} + \gamma V_*\left(S_{t+1}\right) \mid S_t = s, A_t = a\right] \\ &= \max_a \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma V_*\left(s'\right)\right] \end{aligned}$$

Hai công thức cuối chính là hai dạng của phương trình tối ưu Bellman cho V_*. Một cách trực quan, phương trình tối ưu Bellman diễn tả thực tế rằng giá trị của một trạng thái theo một chính sách tối ưu phải bằng với lợi nhuận mong đợi cho hành động tốt nhất từ trạng thái đó. Tương tự, ta cũng có phương trình tối ưu Bellman cho Q_* như sau:

$$\begin{aligned} Q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} Q_*\left(S_{t+1}, a'\right) \mid S_t = s, A_t = a\right] \\ &= \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma \max_{a'} Q_*\left(s', a'\right)\right] \end{aligned}$$

Đối với MDP hữu hạn, phương trình tối ưu Bellman cho v_π có một nghiệm duy nhất độc lập với chính sách. Phương trình tối ưu Bellman thực chất là một hệ phương trình, mỗi phương trình ứng với mỗi trạng thái, vì vậy nếu có n trạng thái, thì sẽ có n phương trình với n ẩn số. Nếu biết $p(.,.|.,.)$ của môi trường, thì về nguyên tắc ta có thể giải hệ phương trình này với giá v_* bằng cách sử dụng bất kỳ một trong các phương pháp giải hệ phương trình phi tuyến. Tuy nhiên điều này trên thực tế khó thực hiện được do chi phí tính toán giải hệ phương trình tốn kém với bài toán phức tạp có nhiều trạng thái và việc xác định $p(.,.|.,.)$ của môi trường là điều rất khó khăn.

4.3 Các phương pháp RL-based

Mục tiêu của *Học tăng cường* là học được chính sách đem lại lợi nhuận tối ưu, các phương pháp học theo kiểu RL-based có thể được chia ra làm 3 loại chính là: *Dynamic Programming, Monte Carlo và Temporal Differrenc.*

4.3.1 Dynamic Programming

Phương pháp *Dynamic Programming* giả sử môi trường đã được mô hình hóa tốt và sẽ tìm các chính sách tốt thông qua đánh giá các *value function.* Hai thuật toán quan trọng của Dynamic Programming là *policy iteration và value iteration.* Thuật toán *Policy iteration* bao gồm 3 bước chính đó là: khởi tạo, đánh giá chính sách và cải thiện chính sách. Ban đầu chính sách được khởi

tạo ngẫu nhiên, ví dụ khởi tạo ngẫu nhiên hành động $a \in A(s)$ cho các trạng thái $s \in \mathcal{S}$. Sau đó, giá trị của trạng thái được tính và đánh giá bằng:

$$V(s) \leftarrow \sum_{s',r} p\left(s', r \mid s, \pi(s)\right)\left[r + \gamma V\left(s'\right)\right],$$

với p là xác suất chuyển và s' là trạng thái tiếp theo. Cuối cùng, chính sách được cập nhật như sau, $\forall s \in \mathcal{S}$:

$$\pi(s) \leftarrow \underset{a}{arg\,max} \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma V\left(s'\right)\right].$$

Như đã chỉ ra trong sách [4], vấn đề mà thuật toán *policy iteration* gặp phải là độ lớn của khối lượng tính toán trong mỗi vòng lặp. Do đó, thuật toán cải tiến của nó là *value iteration*. Các hàm $V(s)$ sẽ được khởi tạo ngẫu nhiên $\forall s \in \mathcal{S}$, sau đó nó được cập nhật trong mỗi bước lặp dựa vào

$$V(s) \leftarrow \max_{a} \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma V\left(s'\right)\right].$$

4.3.2 Temporal difference

Phương pháp *Temporal difference* là sự kết hợp của phương pháp *Dynamic programming và Monte Carlo*. Điểm đặc biệt của phương pháp này là nó không cần mô hình mô phỏng của môi trường mà sẽ cập nhật các giá trị của các hàm xấp xỉ thông qua các đại lượng xấp xỉ khác. Q-learning [5] và Sarsa [6] là hai thuật toán nổi tiếng nhất của loại phương pháp này. Q-learning thuộc loại thuật toán model-free, off-policy để học giá trị của các hành động trong một trạng thái cụ thể. Cách cập nhật hàm giá trị của Q-learning dựa trên phương trình Bellman như sau:

$$Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) + \alpha\left[r_{t+1} + \gamma \max_{a} Q\left(s_{t+1}, a\right) - Q\left(s_t, a_t\right)\right],$$

với α là hệ số học. Sarsa là phiên bản on-policy của Q-learning với cách cập nhật như sau:

$$Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) + \alpha\left[r_{t+1} + \gamma Q\left(s_{t+1}, a_{t+1}\right) - Q\left(s_t, a_t\right)\right].$$

4.4 Các phương pháp Policy gradient

Trong phần này, chúng ta xem xét các phương pháp tìm hiểu một chính sách được tham số hóa có thể chọn các hành động mà không cần tham khảo một hàm giá trị. Hàm giá trị vẫn có thể được sử dụng để tìm hiểu tham số chính sách, nhưng không bắt buộc đối với lựa chọn hành động.

Kí hiệu $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ là vector tham số chính sách. Khi đó ta có: $\pi(a \mid s, \boldsymbol{\theta}) = \Pr\left\{A_t = a \mid S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}\right\}$ là xác suất để hành động a được chọn tại thời điểm t khi môi trường đang ở trạng thái s với tham số θ.

Trong phần này, ta xem xét các phương pháp học tham số chính sách dựa trên gradient của một số hàm độ đo hiệu suất $J(\theta)$ đối với tham số chính sách. Các phương pháp này tìm cách tối đa hóa hiệu suất, vì vậy công thức cập nhật xấp xỉ dựa trên hướng tăng của gradient trong J:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \nabla J\left(\boldsymbol{\theta}_t\right) . \widehat{\nabla J\left(\boldsymbol{\theta}_t\right)} .$$

trong đó $\nabla J\left(\boldsymbol{\theta}_t\right) \widehat{\nabla J\left(\boldsymbol{\theta}_t\right)}$ là một ước lượng ngẫu nhiên mà kỳ vọng của nó xấp xỉ gradient của độ đo hiệu suất đối với $\boldsymbol{\theta}_t$. Các phương pháp tuân theo lược đồ chung này được gọi là phương pháp gradient chiến lược (policy gradient methods).

4.4.1 Xấp xỉ policy (policy approximation)

Mục tiêu của xấp xỉ chính sách $\pi_\theta(s, a)$ là ta cần tìm tham số tối ưu $\boldsymbol{\theta}$, do đó ta cần một hàm số đo độ tốt của chính sách $\pi_{\boldsymbol{\theta}}$ đang được ước lượng bởi các giá trị thuộc vector $\boldsymbol{\theta}$. Trong báo cáo này, ta xét với môi trường chia theo từng tập (tức luôn có một trạng thái kết thúc *terminal state* ở cuối mỗi tập). Đối với môi trường loại này, một hàm dùng để đánh giá độ tốt của chính sách $\pi_{\boldsymbol{\theta}}$ là hàm lợi nhuận theo tập *Episodic-return objective:*

$$\begin{aligned} J_{\mathrm{G}}(\boldsymbol{\theta}) &= \mathbb{E}_{S_0 \sim d_0, \pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1}\right] \\ &= \mathbb{E}_{S_0 \sim d_0, \pi_\theta}\left[G_0\right] \\ &= \mathbb{E}_{S_0 \sim d_0}\left[\mathbb{E}_{\pi_\theta}\left[G_t \mid S_t = S_0\right]\right] \\ &= \mathbb{E}_{S_0 \sim d_0}\left[v_{\pi_\theta}\left(S_0\right)\right] \end{aligned} \tag{3}$$

với d_0 là phân phối xác suất biểu diễn trạng thái bắt đầu. Giá trị của $J_{\mathrm{G}}(\boldsymbol{\theta})$ chính là kỳ vọng của hàm giá trị trạng thái bắt đầu từ S_0.

4.4.2 Policy Gradient

Bài toán đặt ra là ta cần tìm một tham số $\boldsymbol{\theta}$ để cực đại hàm $J(\boldsymbol{\theta})$. Một cách tiếp cận đơn giản và hiệu quả là dùng phương pháp *stochastic gradient ascent* phương pháp hướng tăng ngẫu nhiên. Ý tưởng chính của phương pháp là tăng giá trị đạo hàm của $J(\boldsymbol{\theta})$:

$$\Delta \boldsymbol{\theta} = \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

với $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ là gradient của chính sách được định nghĩa như sau:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_1} \\ \vdots \\ \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_n} \end{pmatrix}$$

và α là tham số thể hiện bước nhảy của thuật toán. Công việc còn lại là tính toán $\nabla_{\boldsymbol{\theta}}(J(\boldsymbol{\theta})$ dựa trên hàm khả vi $\boldsymbol{\theta}$ (tính khả vi có được khi ta tham số hóa $\boldsymbol{\theta}$ bằng mạng *neural*). Dựa vào công thức (3), ta có:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[R] \tag{4}$$

4.4.3 Contextual Bandit Policy Gradient

Ta xét trong trường hợp 1 bước lặp tức chuyển từ thời gian t sang $t+1$, với giá trị cho cặp state-action (S, A) cụ thể thì giá trị $\nabla_{\boldsymbol{\theta}}\mathbb{E}[R(S,A)]$ được tính thông qua đại lượng $r_{sa} = \mathbb{E}[R(S,A)|S = s, A = a]$ như đã trình bày ở thuật toán REINFORCE [7] như sau:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\pi_{\boldsymbol{\theta}}}[R(S,A)] &= \nabla_{\boldsymbol{\theta}} \sum_s d(s) \sum_a \pi_{\boldsymbol{\theta}}(a \mid s) r_{sa} \\
&= \sum_s d(s) \sum_a r_{sa} \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a \mid s) \\
&= \sum_s d(s) \sum_a r_{sa} \pi_{\boldsymbol{\theta}}(a \mid s) \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a \mid s)}{\pi_{\boldsymbol{\theta}}(a \mid s)} \\
&= \sum_s d(s) \sum_a \pi_{\boldsymbol{\theta}}(a \mid s) r_{sa} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a \mid s) \\
&= \mathbb{E}_{d,\pi_{\boldsymbol{\theta}}} \left[R(S,A) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A \mid S) \right]
\end{aligned}$$

Với $d(s)$ là phân phối biểu thị xác suất tác tử đến các trạng thái s. Từ việc lấy mẫu và các phân phối d, ta có thể đơn giản hóa $\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\pi_{\boldsymbol{\theta}}}[R(S,A)]$ như sau:

$$\nabla_{\boldsymbol{\theta}}\mathbb{E}[R(S,A)] = \mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A \mid S) R(S,A)\right]$$

Thuật toán chính sách gradient ngẫu nhiên sẽ cập nhật tham số $\boldsymbol{\theta}$ sau mỗi bước lặp như sau:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha R_{t+1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_t}\left(A_t \mid S_t\right).$$

Để làm giảm phương sai của việc dự đoán các $\mathbb{E}[R(S,A)]$ người ta thường dùng một hàm *baseline* cơ sở như sau:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \left(R_{t+1} - b\left(S_t\right)\right) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_t}\left(A_t \mid S_t\right).$$

4.4.4 Định lý Policy Gradient

Định lý 4.2 *Với mọi hàm chính sách khả vi $\pi_{\boldsymbol{\theta}}(s,a)$, cho d_0 là phân phối xác suất cho trạng thái bắt đầu mỗi tập. Khi đó gradient của chính sách $J(\boldsymbol{\theta}) = \mathbb{E}\left[G_0 \mid S_0 \sim d_0\right]$ là:*

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_{t=0}^{T} \gamma^t q_{\pi_{\boldsymbol{\theta}}}\left(S_t, A_t\right) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\left(A_t \mid S_t\right) \mid S_0 \sim d_0\right]$$

với

$$\begin{aligned}
q_\pi(s,a) &= \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right] \\
&= \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi\left(S_{t+1}, A_{t+1}\right) \mid S_t = s, A_t = a\right]
\end{aligned}$$

Chứng minh: Xét chuỗi lịch sử tương tác giữa tác tử và môi trường là: $\tau = S_0, A_0, R_1, S_1, A_1, R_1, S_2, \ldots$ với lợi nhuận ký hiệu là $G(\tau)$, ta có:

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi) = \nabla_{\boldsymbol{\theta}}\mathbb{E}[G(\tau)] = \mathbb{E}\left[G(\tau) \nabla_{\boldsymbol{\theta}} \log p(\tau)\right]$$

Ta lại có:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \log p(\tau) &= \nabla_{\boldsymbol{\theta}} \log \left[p\left(S_0\right) \pi\left(A_0 \mid S_0\right) p\left(S_1 \mid S_0, A_0\right) \pi\left(A_1 \mid S_1\right) \cdots\right] \\ &= \nabla_{\boldsymbol{\theta}} \left[\log p\left(S_0\right) + \log \pi\left(A_0 \mid S_0\right) + \log p\left(S_1 \mid S_0, A_0\right) + \log \pi\left(A_1 \mid S_1\right) + \cdots\right] \\ &= \nabla_{\boldsymbol{\theta}} \left[\log \pi\left(A_0 \mid S_0\right) + \log \pi\left(A_1 \mid S_1\right) + \cdots\right]\end{aligned}$$

Do đó:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi) &= \mathbb{E}_\pi \left[G(\tau) \nabla_{\boldsymbol{\theta}} \sum_{t=0}^{T} \log \pi\left(A_t \mid S_t\right)\right] \\ &= \mathbb{E}_\pi \left[G(\tau) \sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi\left(A_t \mid S_t\right)\right] \\ &= \mathbb{E}_\pi \left[\sum_{t=0}^{T} G(\tau) \nabla_{\boldsymbol{\theta}} \log \pi\left(A_t \mid S_t\right)\right] \\ &= \mathbb{E}_\pi \left[\sum_{t=0}^{T} \left(\sum_{k=0}^{T} \gamma^k R_{k+1}\right) \nabla_{\boldsymbol{\theta}} \log \pi\left(A_t \mid S_t\right)\right] \\ &= \mathbb{E}_\pi \left[\sum_{t=0}^{T} \left(\sum_{k=t}^{T} \gamma^k R_{k+1}\right) \nabla_{\boldsymbol{\theta}} \log \pi\left(A_t \mid S_t\right)\right] \\ &= \mathbb{E}_\pi \left[\sum_{t=0}^{T} \left(\gamma^t \sum_{k=t}^{T} \gamma^{k-t} R_{k+1}\right) \nabla_{\boldsymbol{\theta}} \log \pi\left(A_t \mid S_t\right)\right] \\ &= \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t q_\pi\left(S_t, A_t\right) \nabla_{\boldsymbol{\theta}} \log \pi\left(A_t \mid S_t\right)\right]\end{aligned}$$

Định lý đã được chứng minh.

Để làm giảm phương sai của việc dự đoán hàm $q_\pi(S_t, A_t)$, người ta thường dùng hàm cơ sở là $v_\pi(S_t)$ như sau: $v_\pi\left(S_t\right)$

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi) = \mathbb{E}\left[\sum_{t=0} \gamma^t \left(q_\pi\left(S_t, A_t\right) - v_\pi\left(S_t\right)\right) \nabla_{\boldsymbol{\theta}} \log \pi\left(A_t \mid S_t\right)\right]$$

Từ đó, ta bỏ tổng xích ma ra ngoài và chia *gradient* thành các *gradient* thành phần như sau

$$\Delta \boldsymbol{\theta}_t = \gamma^t G_t \nabla_{\boldsymbol{\theta}} \log \pi\left(A_t \mid S_t\right)$$

sao cho $\mathbb{E}_\pi\left[\sum_t \Delta \boldsymbol{\theta}_t\right] = \nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi)$. Và từ đó ta có công thức để cập nhật cho $\boldsymbol{\theta}$.

4.4.5 Thuật toán Actor-Critic

Ý tưởng của thuật toán *Actor-Critic* trong việc tìm chính sách tối ưu như sau: Khối *Actor* có nhiệm vụ *exploitation* tận dụng các kiến thức đã biết của tác tử để thực hiện các hành động đem lại

phần thưởng lớn nhất cho từng bước sau đó dùng khối *Critic* để tối ưu chính sách một cách lâu dài dựa trên việc cập nhật các hàm giá trị trạng thái đã được tham số hóa.

One-step Actor–Critic (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

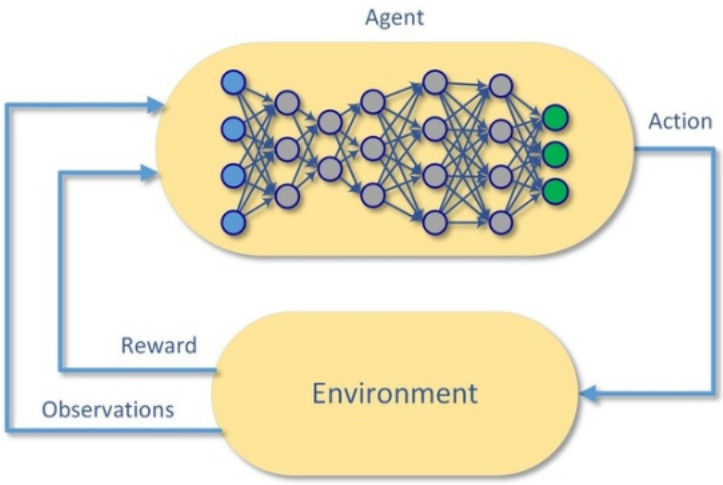
Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
  Initialize S (first state of episode)
  $I \leftarrow 1$
  Loop while S is not terminal (for each time step):
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Hình 5: Thuật toán Actor-Critic 1 bước lặp

Xét phiên bản đơn giản của thuật toán Actor-Critic cho 1 bước lặp được trình bày như ở Hình (5), hàm chính sách sẽ được điều khiển thông qua tham số $\boldsymbol{\theta}$ và hàm giá trị trạng thái sẽ được tham số hóa bởi $\boldsymbol{w}$. Trong đó, ta xấp xỉ G_t bởi $R_{t+1} + \gamma V(S_{t+1})$ và đại lượng δ được gọi là sai số *Temporal difference* trong RL.

4.5 Deep Reinforcement Learning

Mô hình *Học sâu tăng cường* là sự kết hợp của Học sâu và Học tăng cường với đặc điểm chính của mô hình là thay vì lưu trữ các cặp trạng thái - hành động và phần thưởng thì ta sẽ mô hình hóa các thông tin đó dưới dạng các mạng Neural. Việc làm này giúp giảm thiểu dung lượng lưu trữ trong trường hợp số lượng trạng thái và hành động lớn, cũng như tăng tốc độ huấn luyện của mô hình.



Hình 6: Deep Reinforcement Learning

Để có thể giải các bài toán điều khiển tối ưu trên thực tế, các phương pháp *Học tăng cường* cần có một phương pháp lưu trữ hiệu quả số lượng trạng thái lớn của môi trường cũng như cần tăng độ hiểu quả tính toán trong việc "học"giá trị từ các trạng thái. Chính vì lý do đó, các hướng tiếp cận xấp xỉ sử dụng các mạng *Học sâu* ra đời với đặc điểm:

- Ánh xạ các trạng thái s đến các hàm mô tả"đặc trưng"$\phi(s)$ một cách hợp lý.
- Tiến hành tham số hóa các hàm đặc trưng $v_\theta(\phi)$.
- Cập nhật tham số θ dẫn tới cập nhật các *value function* cho chính sách $v_\pi(s) \sim v_\theta(\phi(s))$

Mục tiêu của DRL là tìm ra tham số θ cực tiểu sai lệch giữa v_π và v_θ. Độ sai lệch giữa hai đại lượng trên được minh họa bởi:

$$L(\theta) = E_{S\sim d}\left[(v_\pi(S) - v_\theta(S))^2\right]$$

với d là phân phối lượt truy cập các trạng thái được thực hiện theo chính sách π. Chúng ta có thể dùng các phương pháp hướng giảm để tối ưu hóa hàm mục tiệu:

$$\Delta\theta = -\frac{1}{2}\alpha\nabla_\theta L(\theta) = \alpha E_{S\sim d}\left[(v_\pi(S) - v_\theta(S)\nabla_\theta v_\theta(S)\right]$$

Việc tham số v_θ bởi các *mạng Neural* để dễ dàng tính toán, lưu trữ là ý tưởng chính cho phương pháp DRL. Từ đó, việc đánh giá các chính sách sử dụng phương pháp *stochastic gradient descent* sẽ được thực hiện với các hàm giá trị v_θ.

5 Khảo sát về ứng dụng RL trong việc xây dựng RS

Các hệ thống khuyến nghị sẽ thông qua phản ứng của các khách hàng thu nhận phản hồi của họ về các sản phẩm và cố gắng khuyến nghị các sản phẩm phù hợp với khách hàng dựa trên các phản hồi mà khách hàng cung cấp. Về bản chất RS nhận các thông tin về sản phẩm từ người dùng và cố gắng tối đa hóa sự hài lòng của khách hàng đối với các sản phẩm được khuyến nghị do đó RS có thể được mô hình bằng MDP.

5.1 Mô hình hóa bài toán

Các thuật toán lõi của RS có thể được mô phỏng như các tác tử của RL tiến hành học hỏi từ thông tin mà khách hàng-môi trường cung cấp để tối đa hóa lợi nhuận-sự hài lòng của khách hàng. Cụ thể, RS có thể được mô hình hóa thông qua MDP như sau:

- Trạng thái $\mathcal{S}$: các trạng thái $s_t \in \mathcal{S}$ có thể được cài đặt thông qua lịch sử chọn của khách hàng.
- Hành động $\mathcal{A}$: một hành động $a_t \in \mathcal{A}$ sẽ khuyến nghị một số item cho khách hàng ở thời điểm t.
- Phần thưởng $\mathcal{R}$: Tác tử RL nhận được các phần thưởng $r\left(s_t, a_t\right) \in \mathcal{R}$ dựa trên phản hồi của khách hàng.
- Xác suất chuyển $\mathcal{P}$: $p\left(s' \mid s, a\right) \in \mathcal{P}$ là xác suất chuyển từ $s = s_t$ sang $s' = s_{t+1}$ nếu hành động a được tác tử chọn hành động.
- Discount factor γ.

Cho trước $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$ mục tiêu của tác tử vẫn là tìm chính sách π tối đa hóa lợi nhuận thu được:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r\left(s_t, a_t\right)\right],$$

với T là thời điểm kết thúc. Để có thể mô hình hóa được Rl-based RS, các thách thức gặp phải sẽ bao gồm 4 phần: (1) Biểu diễn các trạng thái, (2) Tối ưu chính sách, (3) Mô hình hóa hàm phần thưởng và (4) Mô hình hóa môi trường, cụ thể như sau:

- **Biểu diễn các trạng thái:** Trong môi trường mà tác tử tương tác thì các trạng thái có thể chứa bất cứ thông tin gì từ các biểu diễn hình ảnh, âm thanh đến dữ liệu từ các bộ cảm biến. Ta cần định nghĩa các trạng thái dựa trên tín hiệu thu được thỏa mãn tính Markov. Để biểu diễn các trạng thái ta sẽ bao gồm 3 cách làm chính là: Coi các item là các trạng thái, Trích xuất thông tin từ người dùng, items, ngữ cảnh để làm trạng thái và cuối cùng là Encoded Embedding thông qua các mô hình DRL.

- **Tối ưu chính sách:** Khi các trạng thái đã được định nghĩa thì mô hình RL sẽ tiến hành lựa chọn các hành động để tối ưu chính sách. Hai hướng thuật toán chính được sử dụng là: Tabular method gồm Q-learning, Sarsa và Monte Carlo Tree Search [8] và các thuật toán xấp xỉ như Deep Q-Network [9], policy gradient [7],...
- **Mô hình hóa hàm phần thưởng** Có hai hướng tiếp cận chính để mô hình hóa hàm phần thưởng: một là thiết kế các hàm đơn giản gồm các phần thưởng dạng số hoặc thiết kế các hàm phần thưởng trích xuất thông tin từ môi trường để đánh giá độ hài lòng của khách hàng.
- **Xây dựng môi trường** Có ba hướng tiếp cận chính để xây dựng môi trường là: offline, online và simualation. Đối với cách làm offline, RS sẽ được xây dựng từ bộ dữ liệu đã có sẵn và môi trường, hàm phần thưởng, trạng thái sẽ được xây dựng thông qua dữ liệu. Cách làm simulation sẽ mô phỏng lại môi trường mà tác tử làm việc. Cách làm online thì môi trường sẽ liên tục được cập nhật và thay đổi dựa vào phản hồi được liên tục nạp vào hệ thống của khách hàng.

5.2 RL-based RS

Khi xây dựng hệ thống RS dựa trên RL sẽ gặp phải 4 thách thức như đã trình bày ở phần trước. Ứng với từng thách thức, nhóm tác giả sẽ tìm hiểu một số paper được liệt kê từ hai bài khảo sát [10], [11] để hoàn thiện báo cáo.

1. **Biểu diễn các trạng thái:** Cách làm đầu tiên của các nhà khoa học là sử dụng một danh sách các items tiến hành trích xuất dữ liệu để biểu diễn các trạng thái. Ví dụ, hệ thống WebWatcher [12] sẽ sử dụng mỗi trang web làm một trạng thái trong mô hình RS khuyến nghị các trang web. Hệ thống khuyến nghị các tài liệu học thuật của [13] coi mỗi nhà khoa học là một trạng thái và khuyến nghị người dùng tìm đọc các bài báo của từng nhà khoa học. Một nhược điểm trong cách làm trên là số lượng trạng thái sẽ tăng lên rất lớn do đó ý tưởng về lưu từng cụm nhỏ item đã được khách hàng chọn để mô hình cho trạng thái được [14] mô hình hóa RS như là trò chơi gridworld game với mỗi grid cell ứng với các khách hàng và các items tương ứng mà họ chọn. Cách tiếp cận thứ hai đó là trích xuất thông tin từ khách hàng, các item và ngữ cảnh để làm trạng thái. RLradio [15] đã thông tin về các kênh radio mà khách hàng yêu thích và thói quen nghe nhạc của họ để định nghĩa các trạng thái. DJ-MC [16] đã sử dụng phương pháp encoding để biểu diễn thông tin miêu tả của các bài hát và xây dựng lên trạng thái gồm k vector biểu diễn bài hát trong 1 playlist.
2. **Tối ưu chính sách:** Các phương pháp temporal difference như Q-learning và Sarsa là hai thuật toán RL phổ biến nhất được sử dụng cho các Rl-based RS [14], [17], [12] dựa vào sự tiết kiệm chi phí tính toán, tính cập nhật online và model-free. Thuật toán Monter Carlo tree search (MCTS) được sử dụng trong [16] với mục đích tìm kiếm hiệu quả trong không gian bài hát vô cùng lớn dựa vào việc phân cụm các bài hát theo thể loại và áp dụng MCTS.
3. **Mô hình hóa hàm phần thưởng:** Có hai cách làm chính là thiết kế các hàm phần thưởng đơn giản dạng số như: [18] +5 cho thêm 1 sản phẩm vào kế hoạch du lịch, +1 cho việc xem kết quả khuyến nghị trên page và 0 cho các trường hợp còn lại. Các cách thiết kế hàm phần

thưởng khác phức tạp hơn dựa trên trích xuất thông tin từ trạng thái như sử dụng khoảng cách Jaccard đo độ tương đồng giữa hai trạng thái [14].

4. **Xây dựng môi trường:** Cách làm hiệu quả và tiết kiệm chi phí tính toán đó là xây dựng môi trường kiểu offline tức huấn luyện và đánh giá mô hình RL-based RS thông qua một bộ dữ liệu đã có sẵn. Hai bộ dữ liệu thường được sử dụng là MovieLen và Million Song [19]. Một cách xây dựng môi trường kiểu khác là mô phỏng simulation. Ví dụ, [18] đã huấn luyện mô hình theo kiểu học có giám sát để xây dựng user behavior model và áp dụng vào trong hệ thống khuyến nghị du lịch NutKing. Cách làm cuối cùng là hiệu quả nhất nhưng tốn kém nhất là xây dựng mô trường online. Một ví dụ tiêu biểu là hệ thống khuyến nghị sách cho cửa hàng online của [20]. Mô hình được đánh giá độ hiểu quả thông qua dữ liệu được cập nhật thường xuyên trong 2 năm.

6 Tổng hợp về tích hợp GAN cho RL-based RS

Mạng GAN được tích hợp trong RL-based RS với mục đích tăng tính khám phá của mô hình giúp chính sách tối ưu đem lại sự hài lòng cho khách hàng. Trong báo cáo này, nhóm tác giả sẽ tiến hành tổng hợp dựa trên các bài báo [21], [3], [1], [2].

Kết hợp quá trình huấn luyện đối nghịch tạo sinh sẽ đem lại việc khuyến nghị các sản phẩm mà khách hàng chưa từng thích từ trước hoặc các sản phẩm mà ban đầu khách hàng không thích nhưng sau đó lại thích [21] dựa trên đặc tính sample ra các mẫu negative của mạng GAN. Mạng GAN sẽ được áp dụng để xây dựng các hệ thống RL-based RS theo kiểu model-based [3], mạng sinh (generator) chịu trách nhiệm sinh ra các khuyến nghị và mô phỏng hành vi khách hàng, mạng phân biệt (discriminator) được dùng để phân biệt so với phần thưởng mà khách hàng nhận được trên thực tế. Mạng GAN trong bài [1] được dùng để sinh dữ liệu từ việc bắt chước các đặc trưng trong sở thích của người dùng cũng như mô phỏng tương tác giữa người dùng và hệ thống. Mạng GAN trong bài [1] giúp tạo ra mô hình tương tác giữa người dùng và hệ thống thông qua dữ liệu ngoại tuyến, từ đó thông qua việc huấn luyện để triển khai policy tối ưu trên trên nền tảng trực tuyến.

6.1 Nghiên cứu của Shi và cộng sự [1]

Các nghiên cứu trong báo cáo: sử dụng học tăng cường để cho kết quả tìm kiếm tốt hơn trên trang bán hàng Taobao.

1. Thay vì huấn luyện học tăng cường trực tiếp trên Taobao, tác giả xây dựng môi trường Taobao ảo để tiết kiệm chi phí khi tạo mẫu huấn luyện mô hình. (*Virtual-Taobao*), giảm thiểu chi phí khi sử dụng mẫu huấn luyện. Taobao ảo được huấn luyện từ hàng trăm triệu bản ghi của người dùng Taobao thực tế.

2. Để cải thiện độ chính xác trên mô hình ảo, tác giả đề xuất mô hình GAN-SD (GAN for Simulating Distributions) là bộ tạo sinh phân phối xác suất tương tự các phân phối xác suất của các đặc trưng của khách hàng trên thực tế.

3. Sử dụng MAIL (Multi-agent Adversarial Imitation Learning) để tạo sinh các hành động khách hàng tốt hơn.

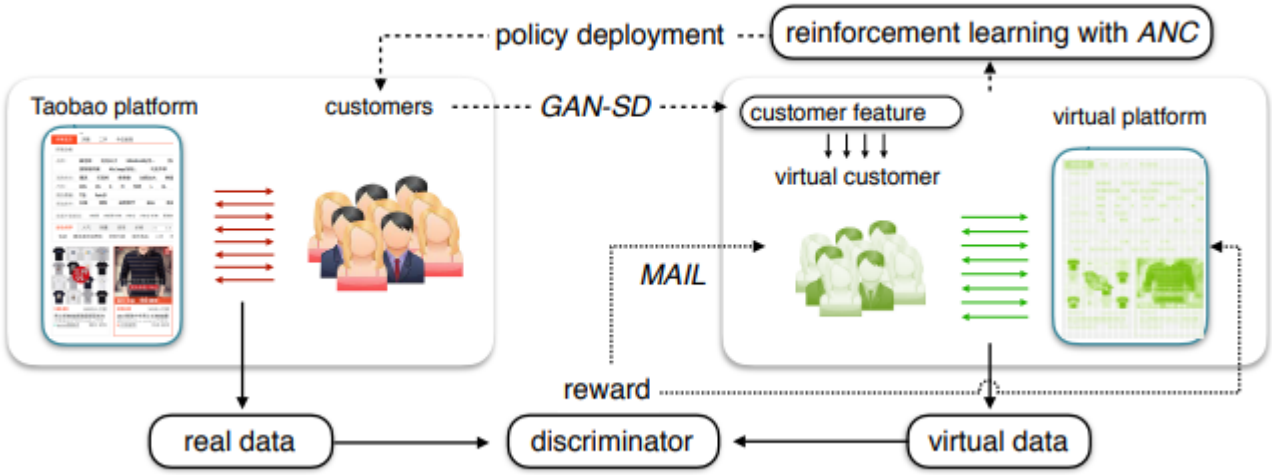
4. Sử dụng ANC(Action Norm Constraint) để chuẩn hóa policy của mô hình.

Luồng hoạt động của mô hình Giải thích về kiến trúc của Taobao ảo được biểu diễn trong hình 10 và được biểu diễn theo các bước như sau:

1. Đầu tiên GAN-SD học cách tạo sinh các đặc trưng của khách hàng.

2. Với các đặc trưng này, các hành vi của khách hàng được học thông qua phương pháp MAIL.

3. Sau khi huấn luyện môi trường Taobao ảo, nền tảng policy được huấn luyện trên Taobao ảo với ANC.



Hình 7: Kiến trúc của Taobao ảo

4. Cuối cùng, nền tảng policy được triển khai trực tiếp trên Taobao.

Kiến thức cơ sở:

- Học tăng cường: Phương pháp học tăng cường giải quyết bài toán chuỗi quyết định thông qua các phép thử và báo lỗi. Ta xem xét cách hình thành mô hình RL cơ bản dựa trên chuỗi quyết định Markov (*Markov Decision Process* MDP).
- Phương pháp Trust Region Policy Optimization, đây là một trong các phương pháp gradient policy hiện đại nhất, được dùng để giải quyết các bài toán biến thiên policy có hạn chế, cụ thể, ta cực đại hóa:

$$\text{maximize}_\theta \, E_t \left[\frac{\pi_\theta \left(a_t \mid s_t \right)}{\pi_{\theta_{old}}} A_t \right]$$

  phụ thuộc vào

$$E_t \left[\text{KL} \left[\pi_{\theta_{\text{otd}}} , \pi_\theta \right] \right] \leq \delta$$

  với $\Pi_{\Theta_{\text{old}}}$ là chính sách trước khi được cập nhật. Quá trình cực tiểu hóa được giải quyết thông qua thuật toán conjugate gradient, sau khi xấp xỉ hàm mục tiêu thông qua hàm tuyến tính và xấp xỉ bậc hai ràng buộc hạn chế.

Học bắt chước

Thay vì cho mô hình học policy thông qua các dữ liệu thô như trong RL truyền thống, sẽ hiệu quả hơn nếu cho mô hình học thông qua việc sao chép các hành động theo các mô hình đã được thử nghiệm. Có hai cách tiếp cận truyền thống là sao chép hành vi và học tăng cường đảo ngược (*Inverse Reinforcement Learning* , IRL)

- Sao chép hành vi (*clone behavior*) là phương pháp mạnh mẽ trong việc bắt chước một bước, nhưng cần bộ dữ liệu lớn để thực hiện trong những tác vụ không tầm thường. Thường nhạy cảm và gặp thất bại khi vị trí của agent quá xa so với hành trình đã mô tả ban đầu.
- IRL tìm hàm phần thưởng bằng cách tối ưu bởi chuyên gia. Thuật toán IRL thường tốn kém khi chạy bởi vì cần mô hình học tăng cường trong mỗi vòng lặp.

Phương pháp cải thiện: sử dụng mô hình GAN trong việc học bắt chước. Phương pháp được đề xuất giúp khắc phục tính nhạy cảm trong mô hình sao chép hành vi và sự tốn kém khi sử dụng IRL với mô hình GAN. GAIL cho phép agent có thể tương tác với môi trường và học policy bằng phương pháp RL trong khi hàm phần thưởng vẫn được cải thiện thông qua quá trình Training. Mô hình GAIL được chứng tỏ đã đạt được những kết quả lý thuyết và thực nghiệm hiệu quả tương tự IRL.

6.1.1 Xây dựng Taobao ảo

Mô tả vấn đề

Cơ chế hoạt động của Taobao được mô tả như sau: Khi khách hàng truy cập và gửi một yêu cầu tìm kiếm tới công cụ, khi đó công cụ tìm kiếm sắp xếp các sản phẩm liên quan và hiển thị trên một trang phản hồi phù hợp (*page view*, PV) . Khách hàng phản hồi lại kết quả hiển thị, ví dụ như mua một sản phẩm nào đó, chuyển trang, hoặc rời Taobao. Quyết định của khách hàng dựa trên các PV cũng như ý định ban đầu của người dùng. Công cụ tìm kiếm tiếp nhận các phản hồi và học hỏi để ra quyết định cho các yêu cầu PV tiếp theo. Mục tiêu kinh doanh của Taobao là tối đa hóa lợi nhuận thông qua chiến lược hiển thị PV hiệu quả.

Hành vi của người dùng ảnh hưởng bởi các PV đã xem trong lịch sử, do đó sẽ hợp lý hơn khi xem mô hình là bài toán quyết định nhiều bước thay vì bài toán học một bước.

Xem công cụ tìm kiếm là agent, hành vi của người dùng là phản hồi từ môi trường. Giả sử người dùng chỉ nhớ và bị ảnh hưởng bởi m PV đã xem cuối trong lịch sử xem.

Ở mặt khác, nếu ta xem khách hàng là agent và công cụ tìm kiếm là môi trường, quá trình mua sắm cho người dùng có thể xem xét như là chuỗi quyết định. Khách hàng phản hồi thông qua hành động tới các sản phẩm đã được xếp hạng, ví dụ như thông qua hành động tìm kiếm. Hành vi của khách hàng cũng được ảnh hưởng từ m PV cuối cùng, được tạo bởi công cụ tìm kiếm. Hành vi của khách hàng có tính chất Markov, do đó cần quan tâm tới việc xây dựng policy mua sắm của khách hàng thông qua sở thích mua sắm của họ trên Taobao.

6.1.2 GAN-SD: tạo sinh các đặc trưng của người dùng

Cách tiếp cận thông qua GAN thông thường có thể tạo ra sản phẩm mới gần giống với bộ dữ liệu ban đầu. Tuy nhiên discriminator của GAN truyền thống lại không có khả năng nắm bắt được cấu trúc phân phối của khách hàng. Để tạo ra phân phối thay vì tạo sinh mẫu đơn giản, tác giả đề xuất mô hình GAN-SD, được mô tả trong Thuật toán 8. Tương tự mạng GAN, GAN-SD duy trì

hai mô hình G và D đối nghịch nhau. Trong khi D tối đa hóa hàm mục tiêu

$$E_{p_{x\sim\mathcal{D}}}[\log D(x)] + E_{p_{x\sim\}}}[\log(1 - D(G(z)))]$$

thì hàm G tối đa hóa hàm

$$E_{p_{\mathcal{D}},p_{\Phi}}[D(G(z)) + \alpha\mathcal{H}(V(G(z))) - \beta KL(V(x)\|V(G(z)))]$$

điểm khác biệt là trong hàm tối đa hóa của G có hàm entropy $\mathcal{H}(V(G(z))$ và $KL(V(x)\|V(G(z)))$ là KL-phân kỳ giữa các biến ngẫu nhiên trên tập huán luyện và tập tạo sinh. Thông qua việc hạn chế bằng KL-phân kỳ và entropy, GAN-SD học các tạo sinh với nhiều thông tin tạo nên dữ liệu thực tế hơn, có thể tạo ra phân phối tốt hơn so với GAN truyền thống.

6.1.3 Tạo sinh tương tác (MAIL: Generating Interactions)

Ý nghĩa cốt lõi của MAIL là mô hình có thể tạo tương tác giữa khách hàng và nền tảng Taobao thông qua mô phỏng policy người dùng. Tác giả đề xuất phương pháp MAIL (*Multi-agent Adversarial Imitation Learning*) dựa trên ý tưởng của GAIL. GAIL cho phép agent tương tác với môi trường trong quá trình huấn luyện trong khi hàm phần thưởng được tối ưu hóa. Mô hình huấn luyện policy khách hàng xem môi trường là công cụ tìm kiếm, đây là yếu tố chưa biết hoặc thay đổi thường xuyên.

Khác với GAIL chỉ huấn luyện agent trong môi trường tĩnh, MAIL huấn luyện policy người dùng đồng thời với policy của công cụ tìm kiếm, giúp agent policy có thể tổng quát hóa các policy của công cụ tìm kiếm khác nhau. Khi MAIL huấn luyện hai mô hình policy đồng thời, agent và môi trường, chỉ cần các thông tin trong lịch sử mà không cần tới dữ liệu cập nhật theo thời gian thực.

Việc học agent và môi trường một cách tuần tự có thể dẫn tới các tác vụ trong không gian rất lớn, do đó mà kết quả cho ra hiệu năng thấp. Thay vào đó ta ó thể tối ưu cả hai yếu tố đồng thời. Trong MAIL, để bắt chước policy agent khách hàng, ta tham số hóa policy khách hàng π^c_κ bởi κ, policy của công cụ tìm kiếm π_σ bởi σ, và hàm phần thưởng $\mathcal{R}^c_\theta$ bởi θ. Ta cũng ký hiệu $\mathcal{T}^c$ là $\mathcal{T}^c_\sigma$. Trạng thái của khách hàng $s^c = \langle s, a, n\rangle$ phụ thuộc vào các hành động của a, ta có:

$$\pi^c\left(s^c, a^c\right) = \pi^c\left(\langle s, a, n\rangle, a^c\right) = \pi^c\left(\langle s, \pi(s, \cdot), n\rangle, a^c\right)$$

Điều này mang ý nghĩa rằng với các tham số của agent tìm kiếm, agent khách hàng có thể trực tiếp ra quyết định mà không cần thông tin của PV. Các hình thành policy chung cung cấp cách thức mô phỏng Π và Π^c cùng nhau.

Thuật toán trong hình 9 mô tả thủ tục của MAIL. Để chạy được, cần lịch sử τ_e và phân phối của khách hàng $\mathcal{P}^x$. Trong bài báo cáo này, $\mathcal{P}^c$ được học từ GAN-SD. Trong thuật toán, ở mỗi vòng lặp, ta thu thập các hành trình tương tác giữa agent người dùng và môi trường. Sau đó tạo mẫu từ các hành trình tương tác và tối ưu hàm phần thưởng bằng phương pháp gradient. Cuối cùng, MAIL trả về policy dành cho agent người dùng là Π^c

Algorithm 1 GAN-SD

Input: Real data distribution $\mathcal{P}_{\mathcal{D}}$
Initialize training variables θ_D, θ_G
for $i = 0, 1, 2...$ **do**
  for k steps **do**
    Sample minibatch from $p_{\mathcal{G}}$
    Sample minibatch from $p_{\mathcal{D}}$
    Update the generator by gradient:

$$\nabla_{\theta_G} E_{p_{\mathcal{G}}, p_{\mathcal{D}}}[D(G(z)) + \alpha \mathcal{H}(G(z)) - \beta KL(x||G(z))]$$

  end for
  Sample minibatch from $p_{\mathcal{G}}$
  Sample minibatch from $p_{\mathcal{D}}$
  Update the discriminator by gradient:

$$\nabla_{\theta_D} E_{p_{x\sim\mathcal{D}}}[\log D(x)] + E_{p_{x\sim\mathcal{G}}}[\log(1 - D(G(z)))]$$

end for
Output: Customer generator G

Hình 8: Cấu trúc của GAN-SD.

6.1.4 ANC: giảm thiểu overfitting cho Taobao ảo

Nếu triển khai policy tương tự như policy trong lịch sử, Taobao ảo có xu hướng hành động giống thực tế hơn so với triển khai hai policy khác nhau. Tuy nhiên các policy tương tự đồng nghĩa với việc cải thiện hạn chế. Các thuật toán RL mạnh như TRPO, có thể dễ dàng huấn luyện Taobao ảo overfit, nghĩa là có thể cho hiệu quả tốt đối với môi trường ảo nhưng hiệu quả thấp trên môi trường thật. Do đó, cần kết hợp chuyển đổi giữa độ chính xác đồng thời với sự cải thiện để cho ra mô hình phù hợp. Ta sử dụng chiến thuật ANC để kiểm soát chuẩn của các hành động. Mô hình hàm phần thưởng thay đổi như sau:

$$r'(s, a) = \frac{r(s, a)}{1 + \rho \max\{\|a\| - \mu, 0\}}$$

Khi chuẩn của hành động lớn hơn chuẩn hành động trong lịch sử, ta cho agent một hình phạt.

6.2 Nghiên cứu của Xueying Bai và cộng sự [2]

Học tăng cường là phương pháp hiệu quả dùng để tối ưu policy cho các hệ thống gợi ý. Hầu hết các giải pháp hiện tại đều chú trong vào cách tiếp cận model-free, điều này yêu cầu tần suất tương tác thường xuyên với môi trường thực, do đó gây tốn kém cho mô hình học. Phương pháp đánh giá ngoại tuyến, chẳng hạn như tạo mẫu quan trọng, có thể giải quyết một phần các hạn chế, tuy nhiên tại yêu cầu lượng lớn dữ liệu đã được ghi và không hoạt động tốt khi không gian hành vi

Algorithm 2 MAIL

Input: Expert trajectories τ_e, customer distribution $\mathcal{P}^c$
Initialize variables κ, σ, θ
for $i = 0, 1, 2..., I$ **do**
  for $j = 0, 1, 2..., J$ **do**
    $\tau_j = \emptyset$, $s \sim \mathcal{P}^c, a \sim \pi_\sigma(s, \cdot), s^c = \langle s, a \rangle$
    while NOT TERMINATED **do**
      sample $a^c \sim \pi(s^c, \cdot)$,
      add (s^c, a^c) to τ_j,
      generate $s^c \sim \mathcal{T}_\sigma^c(s^c, a^c | \mathcal{P}^c)$
    end while
  end for
  Sample trajectories τ_g from $\tau_{0 \sim J}$
  Update θ to in the direction to minimize

$$E_{\tau_g}[\log(\mathcal{R}_\theta^c(s, a))] + E_{\tau_e}[\log(1 - \mathcal{R}_\theta^c(s, a))]$$

  Update κ, σ by optimizing $\pi_{\kappa,\sigma}^c$ with RL in $\mathcal{M}^c$
end for
Output: The customer agent policy π^c

Hình 9: Cấu trúc của MAIL.

người dùng lớn. .Trong nghiên cứu này, tác giả đề xuất giải pháp là một mô hình cơ sở học tăng cường, mô hình hóa tương tác giữa agent và người dùng cho việc học policy ngoại tuyến thông qua mạng đối nghịch tạo sinh (*generative adversarial network*, GAN). Tác giả dựa trên policy gradient với cơ sở RL, ước lượng đồng thời mô hình hồi quy của người dùng cùng với môi trường một cách tin cậy để cập nhật chính sách của người dùng.

Các kiến thức cơ sở:

1. Deep RL for recommendation: là phương pháp tận dụng mô hình học sâu tăng cường trong việc gợi ý các sản phẩm online, ví dụ như tin tức, âm nhạc, video,... Tuy nhiên, hầu hết các phương pháp hiện tại đều dùng phương pháp free-model, do đó không mô hình hóa rõ ràng được tương giác giữa agent và người dùng. Trong các free-model, cách tiếp cận dựa trên cơ sở giá trị (*value-based*), ví dụ như Q-learning, thể hiện các lợi thế riêng ví dụ như học ngoại tuyến liên tục, nhưng lại thiếu sự ổn định khi xấp xỉ hàm. Sự hội tụ trong các thuật toán cũng chưa được nghiên cứu rõ. Ngược lại thì phương pháp dựa trên policy (policy-based), ví dụ như policy gradient duy trì sự ổn định, tuy nhiên cũng có hạn chế về sai lệch dữ liệu không kiểm soát trên thời gian thực, do có hạn chế về quá trình học cũng như cơ sở hạ tầng học. Thông thường, cách tạo mẫu quan trọng (importance sampling) được áp dụng để giải quyết sự sai khác nhưng thay vào đó lại dẫn tới kết quả có phương sai lớn.

2. Offline evaluation: Bài toán học policy ngoại tuyến và đánh giá policy ngoại tuyến là vấn đề phổ biến rộng rãi, vẫn còn là một thách thức đối với RL nói chung, và hệ gợi ý nói riêng. Khi một policy được tối ưu, các phân phối cũng như giá trị kỳ vọng của các gradient cũng được tính toán lại. Đặc biệt nếu đặt trong bối cảnh hệ thống gợi ý, khi danh mục sản phẩm và hành

vi người dùng thay đổi nhanh chóng, các chính sách cũng thay đổi đáng kể, và do đó không khả thi nếu tiếp cận theo cách truyền thống khi hạn chế sự thay đổi trên policy trước dữ liệu mới được cập nhật. Có nhiều cách ước lượng policy ngoại tuyến tận dụng inverse-propensity. Giới hạn inverse-propensity scores và các phương pháp kiểm soát phương sai khác nhau đã được nghiên cứu trong lĩnh vực này. (*inverse-propensity scores*), giới hạn điểm xu hướng nghịch đảo và nhiều phương pháp kiểm soát biến khác nhau đã được phát triển cho mục tiêu ban đầu.

3. RL với mạng đối nghịch tạo sinh: Yu và các cộng sự đã đề xuất SeqGan để mở rộng GAN cho vấn đề tạo sinh chuỗi, với các phần thưởng được cung cấp bởi discriminator mỗi khi kết thúc một giai đoạn thông qua hướng tiếp cận Monte Carlo. Generator tiếp nhận chuỗi hành động và học policy sử dụng ước lượng tích lũy phần thưởng. Trong báo cáo, generator chứa hai thành phần, gồm agent gợi ý và mô hình hành vi người dùng, và tác giả mô hình hóa quá trình tương tác thông qua quá trình huấn luyện đối nghịch và policy gradient. Điều này khác với các nghiên cứu về tác vụ tạo sinh chuỗi trước đây ở chỗ trước đây chỉ hướng tới độ giống nhau giữa chuỗi tạo sinh với quan sát, trong khi mô hình mới tận dụng quá trình huấn luyện đối nghịch để giúp giảm thiểu sự khác nhau giữa mô hình người dùng và từ đó giúp giảm thiểu sự biến thiên trong quá trình huấn luyện agent.

6.2.1 Phát biểu bài toán

Bài toán đặt ra cần giải quyết là việc học policy từ dữ liệu ngoại tuyến sao cho khi triển khai trực tuyến, policy có thể tối đa hóa hàm phần thưởng tích lũy từ các tương tác với người dùng. Trước khi giải quyết, ta đặt ra các giả thuyết dựa trên góc nhìn thực tiễn:

- Mỗi lần người dùng phải click vào một sản phẩm trong danh sách gợi ý
- Sản phẩm không được chọn sẽ không ảnh hưởng tới hành vi người dùng về sau.
- Phần thưởng chỉ liên quan tới các sản phẩm đã chọn. Ví dụ, khi lấy các sản phẩm đã thanh toán của người dùng làm phần thưởng, các sản phẩm thanh toán chỉ được công nhận khi click vào sản phẩm.

Tiếp theo ta tiến hành mô hình hóa bài toán dưới dạng mô hình học tăng cường, với hành vi người dùng được biểu diễn cụ thể thông qua bộ dữ liệu

Phát biểu bài toán

Hệ gợi ý được hình thành từ agent và các hành vi được tạo ra từ policy, với mỗi hành vi cung cấp cho hệ gợi ý một danh sách k sản phẩm. Mỗi thời điểm, thông qua sự tương tác giữa agent và mô trường, tập Ω bao gồm n thành phần $\Omega = \{\tau_1, ..., \tau_n\}$ được ghi lại, với τ_i lưu giá trị bao gồm hành động thứ i của agent , hành vi người dùng và phần thưởng: $\tau_i = \{(a_0^i, c_0^i, r_0^i), (a_1^i, c_1^i, r_1^i), \ldots, (a_t^i, c_t^i, r_t^i)\}$. Dựa trên chuỗi quan sát, ta cần tìm policy tối ưu để tối đa hóa kỳ vọng hàm phần thưởng tích lũy $\mathbb{E}_{T\sim\pi}\left[\sum_{t=0}^{T} r_t\right]$.

Learning framework

Trong quá trình quyết định Markov, môi trường chứa tập các trạng thái S, tập hành động A, tập phân phối của trạng thái chuyển: $P : S \times A \times S$ và hàm phần thưởng $f_r : S \times A \to \mathbb{R}$ được ánh xạ từ cặp trạng thái - hành động. Trong báo cáo, môi trường được mô hình hóa là mô hình hành vi người dùng $\mathcal{U}$, và được học từ dữ liệu ngoại tuyến. S phản ánh lịch sử tương tác trước thời điểm t, và P ký hiệu cho sự chuyển đổi hành vi người dùng. Trong khi đó, dựa trên các giả thiết đã nêu ở trên, tại mỗi thời điểm t, môi trường tạo ra hành động click c_t trên một sản phẩm được gợi ý dựa vào $\mathcal{A}$, phụ thuộc phân phối xác suất đối với trạng thái hiện tại, và hàm phần thưởng f_r tạo ra phần thưởng r_t từ sản phẩm đã click.

Policy gợi ý được học từ cả dữ liệu ngoại tuyến và các mẫu tạo sinh từ các mô hình hành vi người dùng đã học. Tác giả kết hợp việc huấn luyện đối nghịch trong mô hình huấn luyện Policy để:

- Cải thiện mô hình người dùng để đảm bảo các dữ liệu được tạo mẫu gần giống với phân phối dữ liệu thực tế.
- Tận dụng discriminator để mở rộng phần thưởng từ chuỗi được tạo sinh, giảm sự sai khác trong việc ước lượng giá trị.

Phương pháp đề xuất chứa mô hình tương tác được xây dựng bởi $\mathcal{U}$ và $\mathcal{A}$, và phương pháp học chính sách đối nghịch. Đề xuất này được đặt tên là Interactive Recommender GAN (IRecGAN)

6.2.2 Mô hình tương tác cho hệ gợi ý

Phần sau đây sẽ trình bày về mô hình tương tác của hệ gợi ý, chứa hai thành phần:

- $\mathcal{U}$ là mô hình hành vi người dùng, tạo sinh các hành động click dựa trên các sản phẩm được gợi ý, với các phần thưởng tương ứng.
- Agent $\mathcal{A}$ là mô hình tạo sinh hệ gơij ý dựa trên policy của chính nó.

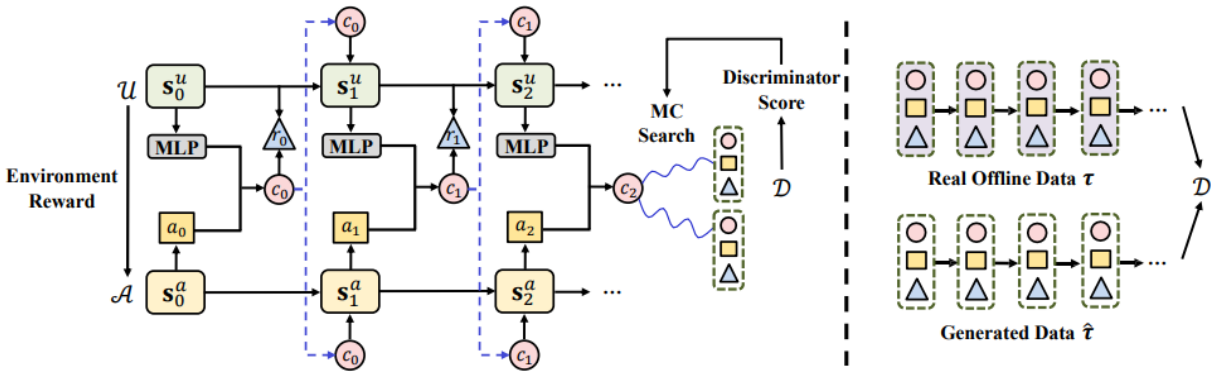
$\mathcal{U}$ và $\mathcal{A}$ tương tác với nhau để tạo ra chuỗi hành vi người dùng cho việc học policy đối nghịch.

Mô hình hành vi người dùng

Tổng quan mô hình IRecGAN. $\mathcal{A}$, $\mathcal{U}$ và $\mathcal{D}$ ký hiệu tương ứng cho mô hình agent, mô hình hành vi người dùng và mô hình discriminator. Trong IRecGAN, $\mathcal{A}$ và $\mathcal{U}$ tương tác với nhau để tạo ra chuỗi gợi ý gần với phân phối của dữ liệu thực, giảm thiểu bias trong $\mathcal{U}$ và cải thiện chất lượng gợi ý trong $\mathcal{A}$. Với tập hành động $a_t = \left\{a_{t(1)}, \dots a_{t(k)}\right\}$, tốp k gợi ý tại thời gian t, ta tính toán xác suất click giữa sản phẩm gợi ý thông qua softmax function:

$$\mathbf{V}^c = \left(\mathbf{W}^c \mathbf{s}_t^u + \mathbf{b}^c\right)^\top \mathbf{E}_t^u, \quad p\left(c_t \mid \mathbf{s}_t^u, a_t\right) = \exp\left(\mathbf{V}_i^c\right) / \sum_{j=1}^{|a_t|} \exp\left(\mathbf{V}_j^c\right)$$

với $V^c \in \mathbb{R}^k$ là véc tơ chuyển chỉ chất lượng được đánh giá của mỗi sản phẩm $a_{t(i)}$ trong trạng thái s_t^u, E_t^u là ma trận nhứng của các sản phẩm được gợi ý, W^x là ma trận trọng số click, và b^c tương



Hình 10: Tổng quan mô hình IRecGAN

ứng với thuật ngữ bias. Với giả thiết rằng mọi phần thưởng mục tiểu chỉ liên quan tới các sản phẩm được click, phần thưởng r_t cho $(s_t^u,\, a_t)$ được tính toán bởi:

$$r_t\left(\mathbf{s}_t^u, a_t\right) = f_r\left(\left(\mathbf{W}^r \mathbf{s}_t^u + \mathbf{b}^r\right)^\top \mathbf{e}_t^u\right) \tag{5}$$

với W_r là ma trận trọng số phần thưởng, b^r là thuật ngữ chỉ bias tương ứng, và f_r là hàm ánh xạ phần thưởng và có thể được điều chỉnh theo định nghĩa phần thưởng trong các hệ gợi ý cụ thể. Lấy ví dụ, nếu ta lấy r_t là thanh toán của sản phẩm đã click c_t, với $r_t = 1$ nếu nó sản phẩm được thanh toán hoặc r_t trong trường hợp ngược lại, f_r có thể được suy ra bởi hàm Sigmoid với đầu ra dạng nhị phân.

Dựa trên phương trình 1 và 5, , lấy phần các phần thưởng phân loại, mô hình hành vi người dùng $\mathcal{U}$ có thể được ước lượng từ dữ liệu ngoại tuyến Ω thông qua ước lượng hợp lý cực đại:

$$\max \sum_{\tau_i \in \Omega} \sum_{t=0}^{T_i} \log p\left(c_t^i \mid \mathbf{s}_t^{u_i}, a_t^i\right) + \lambda_p \log p\left(r_t^i \mid \mathbf{s}_t^{u_i}, c_t^i\right) \tag{6}$$

với λ_p là tham số cân bằng mất mát giữa dự đoán click và dự đoán phần thưởng, và T_i là độ dài của chuỗi quan sát τ_i. Với mô hình hành vi người dùng đã học, người dùng click vào sản phẩm và trả thưởng trên danh sách gợi ý có thể được tạo mẫu dựa trên phương trình 1 và 5.

Agent

Agent thường tạo các hành động dựa trên trạng thái của mô trường cung cấp. Tuy nhiên, trên thực tế, trạng thái của người dùng không thể quan sát được trong hệ gợi ý. Bên cạnh đó, trạng thái của agent khi tạo hành động có thể khác so cới người dùng khi tạo hành động click và trả thưởng. Do đó, ta xây dựng các mô hình trạng thái khác nhau ở bên agent $\mathcal{A}$ để học trạng thái. Tương tự, bên phía người dùng, với các véc tơ chiếu

$$\left\{\mathbf{e}_0^a, \mathbf{e}_2^a, \ldots \mathbf{e}_{t-1}^a\right\},$$

ta mô hình hóa trạng thái người dùng bởi

$$\mathbf{s}_t^a = h^a\left(\mathbf{s}_{t-1}^a, \mathbf{e}_{t-1}^a\right),$$

với s_t^a ký hiệu cho trạn thái được duy trì bởi agent tại thời điểm t.

Xác suất của sản phẩm i chứa trong a_t trên policy π là:

$$\pi\left(i \in a_t \mid \mathbf{s}_t^a\right) = \frac{\exp\left(\mathbf{W}_i^a \mathbf{s}_t^a + \mathbf{b}_i^a\right)}{\sum_{j=1}^{|C|} \exp\left(\mathbf{W}_j^a \mathbf{s}_t^a + \mathbf{b}_j^a\right)} \tag{7}$$

với W_i^a là hàng thứ i của ma trận trọng số hành độ W^a, C là tập toàn bộ sản phẩm trong hệ thống, và b_i^a tương ứng với bias. Không giống như 6, ta không xem xét ảnh hưởng tổ hợp giữa k sản phẩm bằng việc giả sử người dùng sẽ đánh giá chúng độc lập.

6.2.3 Mô hình học policy đối nghịch

Ta sử dụng phương pháp sử dụng policy gradient để cho policy của agent học dựa trên cả dữ liệu ngoại tuyến và dữ liệu tạo sinh. Khi tạo sinh một bộ $\hat{\tau}_{0:t} = \{(\hat{a}_0, \hat{c}_0, \hat{r}_0), \ldots, (\hat{a}_t, \hat{c}_t, \hat{r}_t)\}$ với $t > 0$, ta thu được $\hat{a}_t = \mathcal{A}\left(\hat{\tau}^c_{0:t-1}\right)$, $\hat{c}_t = \mathcal{U}_c\left(\hat{\tau}^c_{0:t-1}, \hat{a}_t\right)$ và $\hat{r}_t = \mathcal{U}_r\left(\hat{\tau}^c_{0:t-1}, \hat{c}_t\right)$ thông qua phương trình 1. Chuỗi tạo sinh kết thúc tại thời điểm t nếu $\hat{c}_t = c_{\text{end}}$, với c_{end} là ký hiệu kết thúc. Phân phối của dữ liệu tạo sinh và ngoại tuyến được ký hiệu tương ứng là g và *data*. Tiếp theo ta bắt đầu huấn luyện mô hình $\mathcal{U}$ từ dữ liệu ngoại tuyến, để có được bias thừa kế thông qua các quan sát và lựa chọn mô hình cụ thể. Bias ảnh hưởng tới chuỗi tạo sinh và do đó có thể gây ra các sai sót trong việc ước lượng giá trị. Để giảm thiểu ảnh hưởng từ bias, ta áp dụng mô hình đối nghịch để tạo ra sự kiểm soát giữa hai mô hình $\mathcal{U}$ và $\mathcal{A}$. Discriminator cũng được sử dụng để chỉnh lại hàm phần thươgr tạo sinh $\hat{r}$ từ việc học policy. Do đó, việc học của agent $\mathcal{A}$ sẽ thông qua chuỗi tạo sinh và phần thưởng.

Policy Learning

Bởi vì việc huấn luyện đối nghịch khuyến khích mô hình IRecGAN tạo sinh sự kiện click và phần thưởng với mẫu tương tự như dữ liệu ngoại tuyến, và giả sử rằng phần thưởng chỉ liên quan tới các sản phẩm được click, ta sử dụng dữ liệu ngoại tuyến như là dữ liệu tạo sinh cho quá trình học policy. Với dữ liệu đã cho $\tau_{0:T} = \{(a_0, c_0, r_0), \ldots, (a_T, c_T, r_T)\}$, bao gồm cả hai dữ liệu offline và dữ liệu tạo sinh, mục tiêu của agent là tối đa hóa phần thưởng tích lũy, với $R_T = \sum_{t=0}^{T}$. Trong dữ liệu tạo sinh, để phân biệt giữa phân phối của dữ liệu tạo sinh và chuỗi ngoại tuyến, phần thưởng tạo sinh trong phương trình 5 có thể bị thay đổi. Để giảm thiểu sự thay đổi này, ta tận dụng chuỗi điểm tạo sinh để căn chỉnh lại phần thưởng tạo sinh: $r_t^s = q_{\mathcal{D}}(\tau_{0:t}\hat{r}_t)$, và xem nó như phần thưởng cho dữ liệu tạo sinh. Gradient tổng quát của hàm mục tiêu được ước lượng bởi:

$$\mathbb{E}_{\tau \sim \{g,\ \text{data}\ \}}\left[\sum_{t=0}^{T} R_t \nabla_{\Theta_a} \log \pi_{\Theta_a}\left(c_t \in a_t \mid \mathbf{s}_t^a\right)\right], \quad R_t = \sum_{t'=t}^{T} \gamma^{t'-t} q_{\mathcal{D}}\left(\tau_{0:t}\right) r_t \tag{8}$$

R_t là xấp xỉ của R_T với yếu tố chiết khấu γ. Nhìn chung, mô hình hành vi người dùng $\mathcal{U}$ chỉ được cập nhật thông qua chuỗi tạo sinh được định nghĩa trong phương trình 1 trên cả dữ liệu ngoại tuyến và dữ liệu tạo sinh. Tuy nhiên agent $\mathcal{A}$ được cập nhật bởi cả chuỗi tạo sinh và phần thưởng. Do đó, phần thưởng chung của $\mathcal{A}$ trong thời gian t là $q_{\mathcal{D}}(\tau_{0:t})(1 + \lambda_r r_t)$, với λ_r là trọng số của hàm

phần thưởng mục tiêu tích lũy. Gradient cho $\mathcal{A}$ là:

$$\mathbb{E}_{\tau\sim\{g,\text{ data }\}}\left[\sum_{t=0}^{T} R_t^a \nabla_{\Theta_a}\log\pi_{\Theta_a}\left(c_t\in a_t\mid \mathbf{s}_t^a\right)\right],\quad R_t^a=\sum_{t'=t}^{T}\gamma^{t'-t}q_{\mathcal{D}}\left(\tau_{0:t}\right)\left(1+\lambda_r r_t\right) \tag{9}$$

6.2.4 Phân tích lý thuyết

Với mỗi một vòng lặp trong thuật toán học policy trong IRecGAN, đầu tiên ta huấn luyện $\mathcal{D}$ với dữ liệu ngoại tuyến, $\mathcal{D}$ được tạo sinh bởi các unknown logging policy và từ $\mathcal{D}$ sẽ sinh ra dữ liệu , và dữ liệu được tạo sinh có được, $\mathcal{D}$ tối ưu là $\mathcal{D}^*(\tau)=\frac{P_{data}(\tau)}{P_{data}(\tau)+P_g(\tau)}$.

Quá trình tạo sinh chuỗi

Cả $\mathcal{A}$ và $\mathcal{U}$ đều góp phần tạo nên chuỗi tạo sinh trong IRecGAN. $\mathcal{U}$ được cập nhật bởi gradient trong phương trình 7 để tối ưu chuỗi tạo sinh mục tiêu. Tại thời điểm t, chuỗi tạo sinh phần thưởng cho $\mathcal{A}$ trên dữ liệu tạo sinh là: $\mathbb{E}_{\tau\sim g}\left[V_g\right]=\mathbb{E}_{\tau\sim g}\left[\sum_{t=0}^{T} q_{\mathcal{D}}\left(\tau_{0:t}\right)\right]=\sum_{t=0}^{T}\mathbb{E}_{\tau_{0:t}\sim g}\left[\mathcal{D}\left(\tau_{0:T}\mid\tau_{0:t}\right)\right]$. Với giá trị D^* tối ưu, chuỗi tạo giá trị tạo sinh có thể viết lại như sau:

$$\mathbb{E}_{\tau\sim g}\left[V_g\right]=\sum_{t=0}^{T}\mathbb{E}_{\tau_{0:t}\sim g}\left[\frac{P_{data}\left(\tau_{0:T}\mid\tau_{0:t}\right)}{P_{data}\left(\tau_{0:T}\mid\tau_{0:t}\right)+P_g\left(\tau_{0:T}\mid\tau_{0:t}\right)}\right] \tag{10}$$

Cực đại hóa mỗi phần trong phương trình 10 là mục tiêu của bộ tạo sinh tại thời điểm t trong GAN. Lời giải tối ưu cho các phần trên là: $P_g\left(\tau_{0:T}\mid s_0\right)=P_{\text{data}}\left(\tau_{0:T}\mid s_0\right)$. Điều này nghĩa rằng $\mathbb{A}$ có thể cực đại hóa chuỗi giá trị tạo sinh, giúp tọ ra phân phối giống với dữ liệu thực. Bên cạnh việc tối ưu toàn cục, phương trình 10 có thể khuyến khcihs $\mathbb{A}$ có thể trả thưởng cho mỗi $P_g\left(\tau_{0:T}\mid\tau_{0:t}\right)=P_{data}\left(\tau_{0:T}\mid\tau_{0:t}\right)$, mặc dù $\tau_{0:t}$ ít có khả năng tạo ra từ P_g. Điều này ngăn chặn việc IRecGAN chỉ gợi ý các sản phẩm dựa trên sở thích tức thời của người dùng.

Ước lượng giá trị

Agent $\mathcal{A}$ phải được cập nhật để cực đại hóa giá trị kỳ vọng của phần thưởng mục tiêu V_a. Để đạt được điều này, ta sử dụng discriminator $\mathcal{D}$ để chỉnh lại ước lượng của V_a trên chuỗi tạo sinh, ta cũng kết hợp dữ liệu ngoại tuyến để đánh giá V_a cho policy Π_{Θ_α}:

$$\mathbb{E}_{\tau\sim\pi_{\Theta_a}}\left[V_a\right]=\lambda_1\sum_{t=0}^{T}\mathbb{E}_{\tau_{0:t}\sim g}\frac{P_{data}\left(\tau_{0:t}\right)}{P_{data}\left(\tau_{0:t}\right)+P_g\left(\tau_{0:t}\right)}\hat{r}_t+\lambda_2\sum_{t=0}^{T}\mathbb{E}_{\tau_{0:t}\sim\text{ data }} r_t \tag{11}$$

với $\hat{r}_t$ là phần thưởng tạo sinh bởi $\mathcal{U}$ tại thời điểm t và r_t là phần thưởng thực tế. Chỉ số λ_1 và λ_2 biểu diễn tỷ số giữa dữ liệu tạo sinh và dữ liệu ngoại tuyến trong suốt quá trình huấn luyện mô hình, với giả thiết $\lambda_1+\lambda_2=1$. Với ký hiệu $P\left(\tau_{0:t}\right)$ thông qua $P\left(\tau_{0:T}\mid\tau_{0:t}\right)$. Từ đó, ta thấy rằng có 3 bias trong công thức ước lượng giá trị:

$$\Delta=\hat{r}_t-r_t,\quad \delta_1=1-P_{\pi_{\Theta_a}}\left(\tau_{0:t}\right)/P_g\left(\tau_{0:t}\right),\quad \delta_2=1-P_{\pi_{\Theta_a}}\left(\tau_{0:t}\right)/P_{\text{data}}\left(\tau_{0:t}\right). \tag{12}$$

Dựa trên các nguồn bias khác nhau, ước lượng giá trị kỳ vọng của phương trình 11 là:

$$\begin{aligned}\mathbb{E}_{\tau\sim\pi_{\Theta_a}}[V_a] &= \lambda_1 \sum_{t=0}^{T} \mathbb{E}_{\tau_{0:t}\sim g} \frac{P_{\pi_{\Theta_a}}(\tau_{0:t})}{P_g(\tau_{0:t})} \frac{\Delta + r_t}{2-(\delta_1+\delta_2)} + \lambda_2 \sum_{t=0}^{T} \mathbb{E}_{\tau_{0:t}\sim \text{ data}} \left(\frac{P_{\pi_{\Theta_a}}(\tau_{0:t})}{P_{\text{data}}(\tau_{0:t})} + \delta_2 \right) r_t \\ &= V_a^{\pi_{\Theta_a}} + \sum_{t=0}^{T} \mathbb{E}_{\tau_{0:t}\sim\pi_{\Theta_a}} w_t \Delta + \sum_{t=0}^{T} \mathbb{E}_{\tau_{0:t}\sim \text{ data}} \lambda_2 \delta_2 r_t - \sum_{t=0}^{T} \mathbb{E}_{\tau_{0:t}\sim\pi_{\Theta_a}} (\lambda_1 - w_t) r_t,\end{aligned} \tag{13}$$

với $w_t = \frac{\lambda_1}{2-(\delta_1+\delta_2)}$. Δ và δ_1 được lấy từ bias của mô hình hành vi người dùng $\mathcal{U}$. Bởi vì quá trình huấn luyện đối nghịch giúp cải thiện $\mathcal{U}$ để nắm bắt được mẫu dữ liệu thực tế, mô hình lại giảm giá trị Δ và δ_2. Ta có thể điều chỉnh giá trị tỷ số λ_1 để giảm w_t, giá trị $w_t\Delta$ có thể nhỏ tùy ý. Chuỗi phần thưởng tạo sinh cho agent $\mathcal{A}$ khuyến khích phân phối g gần với phân phối thật của dữ liệu. Bởi vì $\delta_2 = 1 - \frac{P_{\pi_\Theta}(\tau_{0:t})}{P_g(\tau_{0:t})} \cdot \frac{P_g(\tau_{0:t})}{P_{data}(\tau_{0:t})}$, bias δ_2 có thể giảm tùy ý. Điều này chứng tỏ rằng phương pháp có thể được kiểm soát hiệu quả thông qua các bias.

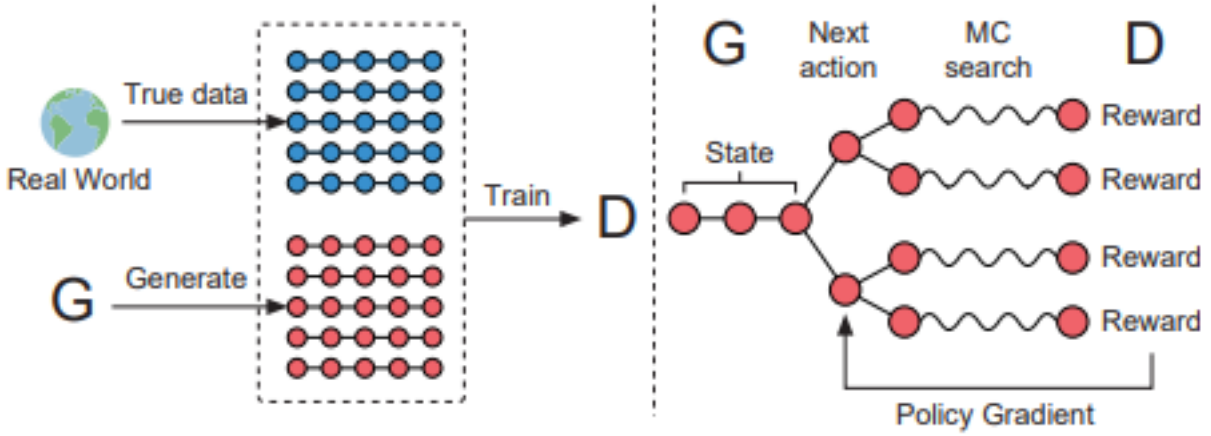
6.3 Nghiên cứu của Lantao Yu và cộng sự [3]

Generative Adversarial Nets (Gan) là một cách mới để đào tạo Generative models, sử dụng discriminative model để hướng dẫn đào tạo Generative models đạt được đáng kể trong việc tạo ra dữ liệu có giá trị thực.Tuy nhiên, lại có hạn chế khi mục đích tạo ra một chuỗi tokens. Discriminative model chỉ có thể đánh giá một chuỗi hoàn chỉnh. Bài báo này đề xuất một framework tạo chuỗi được gọi là SeqGan, để giải quyết các vấn đề. Mô hình hóa trình tạo dữ liệu như một chính sách ngẫu nhiên trong RL, SeqGAN bỏ qua sự khác biệt trình tạo bằng cách trực tiếp cập nhật policy gradient. Tính hiệu phần thưởng RL đến từ bộ phân biệt GAN được đánh gia trên một chuỗi hoàn chỉnh và được chuyển trở lại các bước hành động trạng thái trung gian bằng cách sử dụng tìm kiếm Monte Carlo [22]. Trong bài báo này, sử dụng recurrent neural networks (RNNs) [23] là Generative model và convolutional neural network (CNN) [24] là Discriminator.

6.3.1 Sequence Generative Adversarial Nets

Bài toán tạo chuỗi được biểu thị như sau. Từ một tập dữ liệu gồm các chuỗi có cấu trúc trong thế thực được đào tạo Generative model tham số θ (G_θ) để tạo ra một chuỗi $Y_{1:T} = (y_1, y_2, y_3, ..., y_t, ..., y_T), y_t \in \mathcal{Y}$ trong đó $\mathcal{Y}$ là lực lượng của tokens ứng cử viên. Tại bước thời gian t, trạng thái s hiện tại tạo bởi token $(y_1, y_2, y_3, ..., y_{t-1})$ và hành động a tiếp theo y_t chọn. Do đó, mô hình chính sách $G_\theta(y_t|Y_{1:t})$ là ngẫu nhiên, trong khi quá trình chuyển đổi trạng thía xác định sau khi một hành động đã được chọn.

Ngoài ra, đào tạo discriminative model với tham số ϕ (D_ϕ) (Goodfellow and others 2014) để cải tiến G_θ. $D_\phi(Y_{1:T})$ là xác suất một chuỗi $Y_{1:T}$ là chuỗi dữ liệu thực. Trong hình 1, discriminative model D_ϕ được đào tạo từ các chuỗi dữ liệu thực và chuỗi dữ liệu được tạp ra từ Generative model G_θ . Đồng thời, Generative model G_θ được cập nhật bằng cách sử dụng Policy Gradient và tìm kiếm Monte Carlo (MC). Phần thưởng được ước tính theo khả năng nó đánh lừa discriminative model D_ϕ.



Hình 11: Hình minh họa SeqGan

6.3.2 SeqGan via Policy Gradient

Khi không có phần thưởng trung gian, mục tiêu của Generator model $G_\theta(y_t|Y_{1:t-1})$ là một chuỗi từ trạng thái bắt đầu s_0 để tối đa hóa phần thưởng kỳ vọng:

$$J_\theta = \mathbb{E}[R_T|s_0, \theta] = \sum_{y_1 \in \mathcal{Y}} G_\theta(y_1|s_0).Q_{D_\phi}^{G_\theta}(s_0, y_1)$$

Tính hợp lý của hàm mục tiêu đối với một chuỗi bắt đầu từ một trạng thái ban đầu nhất định, mục tiêu của Generator là tạo ra một chuỗi mà Discriminator coi là thật.

Trong bài báo này sử dụng thuật toán REIFORCE [7] lấy và xem xét xác suất ước tính có là thật bởi discriminator $D_\phi(Y_{1:T})$ là phần thưởng:

$$Q_{D_\phi}^{G_\theta}(a = y_T, s = Y_{1:T-1}) = D_\phi(Y_{1:T})$$

Discriminator chỉ cung cấp giá trị thưởng cho một trình tự đã hoàn thành. Để đánh giá giá trị hành động trạng thái trung gian, Lantao Yu và cộng sự đã áp dụng tìm kiếm Monte Carlo với chính sách triển khai G_β để lấy các tokens T-t cuối cùng chưa biết. Bài báo biểu diễn sau N lần tiềm kiếm Monte Carlo như sau:

$$\{Y_{1:T}^1, Y_{1:T}^2, ..., Y_{1:T}^N\} = MC^{G_\beta}(Y_{1:t}, N)$$

Trong đó $Y_{1:t} = (y_1, ..., y_t)$ là trạng thái hiện tại, $Y_{t+1:T}^n$ là mẫu được lấy ra dựa trên G_β. Trong bài báo G_β được lấy (Silver et al. 2016). Để giảm phương sai và có được đánh giá chính xác hơn về giá trị hành động, bài báo chạy chính sách triển khai bắt đầu từ trạng thai hiện tại cho đến khi kết

thúc chuỗi trong N lần để nhận mẫu đầu ra. Có công thức:

$$Q_{D_\phi}^{G_\theta}(s = Y_{1:t-1}, a = y_t) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} D_\phi\left(Y_{1:T}^n\right), Y_{1:T}^n \in \mathrm{MC}^{G_\beta}\left(Y_{1:t}; N\right) & \text{với } t < T \\ D_\phi\left(Y_{1:t}\right) & \text{với } t = T, \end{cases} \tag{14}$$

Việc sử dụng discriminator D_ϕ như một hàm phần thưởng giúp cập nhật động để cải thiện Generative model lặp đi lặp lại. Khi có một tập hợp chuỗi được tạo thực tế hơn, thì Discrimintor model được đào tạo như sau:

$$\min_\phi -\mathbb{E}_{Y\sim p_{\text{data}}}\left[\log D_\phi(Y)\right] - \mathbb{E}_{Y\sim G_\theta}\left[\log\left(1 - D_\phi(Y)\right)\right]$$

Mỗi khi có được mô hình phân biệt mới thì sẽ cập nhật lại Generator. Phương pháp dựa trên chính sách được đề xuất dựa vào việc tối ưu hóa tham số chính sách để trực tiếp tối da hóa phần thưởng dài hạn. Gradient của hàm mục tiêu $J(\theta)$ với các tham số θ của Generator được suy ra:

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_{Y_{1:t-1}\sim G_\theta}\left[\sum_{y_t\in\mathcal{Y}} \nabla_\theta G_\theta\left(y_t \mid Y_{1:t-1}\right) \cdot Q_{D_\phi}^{G_\theta}\left(Y_{1:t-1}, y_t\right)\right] \\ &\simeq \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{y_1\sim G_\theta(y_t|Y_{1:t-1})}\left[\nabla_\theta \log G_\theta\left(y_t \mid Y_{1:t-1}\right) \cdot Q_{D_\phi}^{G_\theta}\left(Y_{1:t-1}, y_t\right)\right] \end{aligned} \tag{15}$$

Trong đó $Y_{1:t-1}$ là trạng thái trung gian lấy từ G_θ. Kỳ vọng $E[.]$ được tính bằng phương pháp lấy mẫu, cập nhật các tham số θ của Generator như sau:

$$\theta \leftarrow \theta + \alpha_h \nabla_\theta J(\theta)$$

trong đó $\alpha_h \in \mathbb{R}^+$ là tốc độ học tập ở bước thứ h.

Thuật toán Sequence Generative Adversarial Nets như sau:

Algorithm 1 Sequence Generative Adversarial Nets

Require: generator policy G_θ; roll-out policy G_β; discriminator D_ϕ; a sequence dataset $\mathcal{S} = \{X_{1:T}\}$

Initialize G_θ, D_ϕ with random weights θ, ϕ.
Pre-train G_θ using MLE on $\mathcal{S}$
$\beta \leftarrow \theta$
Generate negative samples using G_θ for training D_ϕ
Pre-train D_ϕ via minimizing the cross entropy
repeat
  for g-steps **do**
    Generate a sequence $Y_{1:T} = (y_1, \ldots, y_T) \sim G_\theta$
    for t in $1 : T$ **do**
      Compute $Q(a = y_t; s = Y_{1:t-1})$ by Eq. (4)
    end for
    Update generator parameters via policy gradient Eq. (8)
  end for
  for d-steps **do**
    Use current G_θ to generate negative examples and combine with given positive examples $\mathcal{S}$
    Train discriminator D_ϕ for k epochs by Eq. (5)
  end for
  $\beta \leftarrow \theta$
until SeqGAN converges

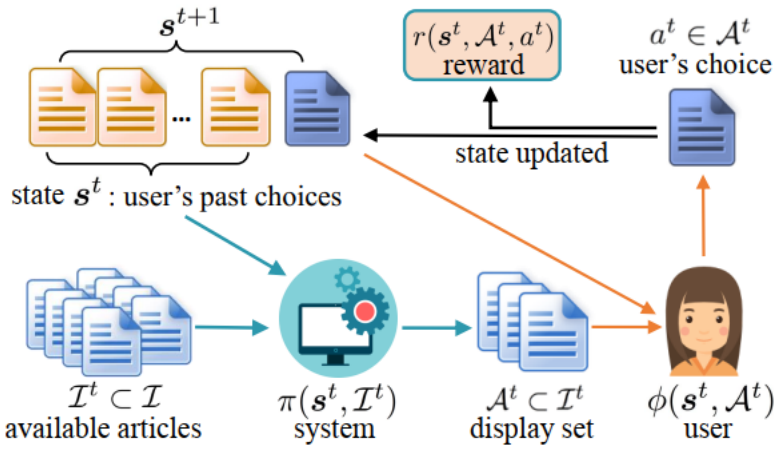
Hình 12: Thuật toán Sequence Generative Adversarial Nets

7 Phương pháp luận

Trong phần này, nhóm tác giả sẽ trình bày nội dung của bài báo [21]. Để xây dựng hệ thống khuyến nghị, các tác giả đã thích hợp mô hình GAN trong việc mô hình hóa hành vi người dùng và thiết kế quá trình luyện bằng hàm đối nghịch tạo sinh giúp quá trình khai phá exploration được mở rộng. Đồng thời các tác giả cũng xây dựng mạng Cascading Q-network để phục vụ quá trình huấn luyện tìm chính sách tối ưu khi không gian các hành động của hệ thống quá lớn. Mạng Cascading Q-network sẽ "trồng"nhiều hàm giá trị hành động để chọn ra các hành động tối ưu trong không gian.

7.1 Mô hình hóa bài toán

Hệ thống khuyến nghị sẽ được cài đặt như sau: RS hiển thị k items (có thể là video, bài hát,...) cho người dùng chọn lựa. Người dùng cung cấp feedback bằng cách chọn k items trên hoặc không chọn cái nào sau đó hệ thống khuyến nghị k items khác trên một trang mới.



Hình 13: Hệ thống khuyến nghị

Mục tiêu chính của RS là khuyến nghị các mặt hàng tăng sự hài lòng cho người dùng, điều này phù hợp với mục tiêu chính của các mô thức RL cho RS là đem lại lợi nhuận lâu dài (sự hài lòng) cho người dùng . Các hệ thống khuyến nghị dựa trên RL sẽ đi tìm một $\pi(s, \mathcal{I})$ được chọn từ tập các items $\mathcal{I}$ ứng với một trạng thái của khách hàng *user state* s, sao cho phần thưởng kỳ vọng tích lũy lớn nhất,

$$\pi^* = \arg\max_{\pi(\boldsymbol{s}^t, \mathcal{I}^t)} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r\left(\boldsymbol{s}^t, a^t\right)\right],$$

với $\boldsymbol{s}^0 \sim p^0, \mathcal{A}^t \sim \pi\left(\boldsymbol{s}^t, \mathcal{I}^t\right), \boldsymbol{s}^{t+1} \sim P\left(\cdot \mid \boldsymbol{s}^t, \mathcal{A}^t\right)$, $a^t \in \mathcal{A}^t$. Mô hình được minh họa trong Hình 13. Các đặc trưng của RL được mô hình hóa như sau:

1. Môi trường: tương ứng với một khách hàng, người sẽ chọn một (hoặc không chọn) trong k items được hệ khuyến nghi·.

2. Trạng thái $s^t \in \mathcal{S}$: tương ứng một chuỗi có sắp thứ tự chứa thông tin lịch sử chọn của khách hàng.

3. Hành động $\mathcal{A}^t \in \begin{pmatrix} \mathcal{I}^t \\ k \end{pmatrix}$ của hệ thống: tương ứng tập con gồm k items được chọn bởi hệ thống từ $\mathcal{I}^t$ để khuyến nghị ra cho người dùng. $\begin{pmatrix} \mathcal{I}^t \\ k \end{pmatrix}$ nghĩa là tất cả các tập con gồm k items được chọn ra từ $\mathcal{I}^t \subset \mathcal{I}$ là các items có sẵn tại thời điểm t.

4. Xác suất chuyển $P\left(\cdot \mid s^t, \mathcal{A}^t\right) : \mathcal{S} \times \begin{pmatrix} \mathcal{I} \\ k \end{pmatrix} \mapsto \mathcal{P}(\mathcal{S})$ sẽ được mô phỏng bằng mô hình hành vi người dùng *user behavior* sẽ cho biết xác suất chuyển từ $\boldsymbol{s}^t$ sang $\boldsymbol{s}^{t+1}$ khi có tập hiển thị $\mathcal{A}^t$. Xác suất này sẽ tương đương với $\phi\left(\boldsymbol{s}^t, \mathcal{A}^t\right)$ được định nghĩa ở phần 7.2.

5. Hàm phần thưởng $r\left(\boldsymbol{s}^t, \mathcal{A}^t, a^t\right) : \mathcal{S} \times \begin{pmatrix} \mathcal{I} \\ k \end{pmatrix} \times \mathcal{I} \mapsto \mathbb{R}$: sẽ tương đương với ma trận utility biểu thị cho độ hài lòng của khách hàng khi chọn $a^t \in \mathcal{A}^t$ ở trạng thái s^t.

6. Chính sách $\mathcal{A}^t \sim \pi\left(s^t, \mathcal{I}^t\right) : \mathcal{S} \times 2^{\mathcal{I}} \mapsto \mathcal{P}\left(\begin{pmatrix} \mathcal{I} \\ k \end{pmatrix}\right)$: biểu thị cho xác suất hiển thị tập $\mathcal{A}^t$ của $\mathcal{I}^t$ tại trạng thái s^t.

Chú ý: Trong mô hình trên thì môi trường, trạng thái và xác suất chuyển trạng thái sẽ là các đặc trưng tương ứng với khách hàng. Hành động và chính sách sẽ là cắc đặc trưng tương ứng với hệ khuyến nghi·, còn phần thưởng sẽ là đặc trưng của cả hệ thống và khách hàng. Ký hiệu $r\left(\boldsymbol{s}^t, \mathcal{A}^t, a^t\right)$ được sử dụng để nhấn mạnh rằng phần thưởng được tính thông qua các tập items mà hệ khuyến nghi·. Tuy nhiên, giá trị của hàm phần thưởng được định nghĩa thông qua trạng thái của khách hàng và item mà khách hạng chọn nên $r\left(s^t, \mathcal{A}^t, a^t\right) = r\left(s^t, a^t\right) \cdot \mathbf{1}\left(a^t \in \mathcal{A}^t\right)$. Trong các phần sau, ký hiệu $r\left(\boldsymbol{s}^t, a^t\right)$ sẽ được giản lược thành $r\left(\boldsymbol{s}^t, \mathcal{A}^t, a^t\right)$ và giả sử $a^t \in \mathcal{A}^t$ là đúng.

7.2 Mô hình hóa hành vi người tích hợp mạng GAN

7.2.1 Xây dựng user behavior tối đa hóa phần thưởng

Chúng ta cần xây dựng mô hình user behavior dựa trên sự lựa chọn và tâm lý khách hàng nên mô hình sẽ được xây dựng dựa trên hai giả thiết sau:

- Khách hàng sẽ luôn lựa chọn các item giúp tối đa hóa sự thỏa mãn của họ. Khách hàng có thể chọn 1 trong các item được khuyến nghị hoặc không chọn item nào và khách hàng vẫn sẽ nhận được phần thưởng tương ứng với việc không chọn item nào.
- Hàm phần thưởng cần được thiết kế dựa vào cả lịch sử chọn của khách hàng và item mà khách hàng vừa chọn khi hệ khuyến nghi·. Ví dụ, một người ban đầu có thể không thích nhạc của Sơn Tùng MTP nhưng họ có thể thích sau khi được khuyến nghị nghe lại các bài nhạc đó

cũng như khách hàng có thể chán nghe nhạc của Taylor Swift nếu được khuyến nghị nghe quá nhiều.

Để mô hình hóa được mô hình ta cần đưa yếu tố lịch sử chọn của khách hàng biểu thị qua trạng thái s^t của khách hàng và items mà khách hàng vừa chọn a^t làm đầu vào cho hàm phần thưởng $r\left(s^t, a^t\right)$. Giả sử ở thời điểm t, khách hàng được khuyến nghị k items $\mathcal{A}^t = \{a_1, \cdots, a_k\}$ và features vector lưu trữ thông tin của các items đó tương ứng là $\{f_1^t, \cdots, f_k^t\}$ (ví dụ, items là hình ảnh thì feature vector được trích xuất qua các lớp CNN). Khách hàng sẽ thực hiện hành động $a^t \in \mathcal{A}^t$ theo user behavior ϕ^* tối ưu sự thỏa mãn của khách hàng. Cụ thể, ϕ^* chính là phân phối xác suất chọn các items trong $\mathcal{A}^t$, và là kết quả của (16).

$$\phi^*\left(s^t, \mathcal{A}^t\right) = \arg\max_{\phi \in \Delta^{k-1}} \mathbb{E}_\phi\left[r\left(s^t, a^t\right)\right] - R(\phi)/\eta \tag{16}$$

với Δ^{k-1} là đơn hình xác suất *probability simplex* và $R(\phi)$ là một hàm regularization lồi để tăng tính khám phá của quá trình luyện và η là một tham số điều khiển mức độ regularization. Trong paper [21], họ sử dụng *negative Shannon entropy* để làm hàm regularization.

Bổ đề 7.1 *Khi thiết lập hàm regularization ở (16) là $R(\phi) = \sum_{i=1}^k \phi_i \log \phi_i$ thì nghiệm tối ưu ϕ^* cho (16) có dạng là:*

$$\phi^*\left(\boldsymbol{s}^t, \mathcal{A}^t\right)_i = \exp\left(\eta r\left(\boldsymbol{s}^t, a_i\right)\right) / \sum_{a_j \in \mathcal{A}^t} \exp\left(\eta r\left(s^t, a_j\right)\right)$$

Chứng minh: Từ công thức tại (16), ta có:

$$\phi^*\left(\boldsymbol{s}^t, \mathcal{A}^t\right) = \arg\max_{\phi \in \Delta^{k-1}} \mathbb{E}_\phi\left[r\left(\boldsymbol{s}^t, a^t\right)\right] - \frac{1}{\eta} R(\phi).$$

Ký hiệu $\phi^t = \phi\left(s^t, \mathcal{A}^t\right)$ và ϕ là tất cả các phân phối xác suất trong Δ^{k-1}. Tập các hành động $\mathcal{A}^t = \{a_1, \cdots, a_k\}$, do đó công thức tính kỳ vọng với các $a^t \in \mathcal{A}^t$ tương đương như sau:

$$\mathbb{E}_\phi\left[r\left(s^t, a^t\right)\right] - \frac{1}{\eta} R(\phi) = \sum_{i=1}^k \phi_i^t r\left(s^t, a_i\right) - \frac{1}{\eta} \sum_{i=1}^k \phi_i^t \log \phi_i^t \tag{17}$$

Tiến hành đạo hàm (17) theo từng ϕ_i^t và cho đạo hàm bằng 0 để tìm điểm cực trị ta có:

$$r(s^t, a_i) - \frac{1}{\eta}(\log \phi_i^t + 1) = 0$$

Từ đó, $\phi_i^t = \exp \eta r(s^t, a_i) - 1$ kết hợp với $\sum_{i=1}^k \phi_i^t = 1 \Longrightarrow \exp = \sum_{j=1}^k \exp\left(\eta r\left(\boldsymbol{s}^t, a_j\right)\right)$. Do đó giá trị $\phi^{t*} \in \Delta^{k-1}$ tối ưu (17) là:

$$\phi_i^{t*} = \frac{\exp\left(\eta r\left(\boldsymbol{s}^t, a_i\right)\right)}{\sum_{j=1}^k \exp\left(\eta r\left(\boldsymbol{s}^t, a_j\right)\right)},$$

Từ đó được điều phải chứng minh.

Từ đây, ta có nhận xét rằng hàm $\phi(s^t, \mathcal{A}^t)$ có ý nghĩa biểu diễn quá trình chọn lựa của khách hàng theo ý nghĩa "tham lam". Khách hàng sẽ chọn những item đem lại cảm giác hài lòng nhất cho khách hàng.

Chú ý:

- Các dạng hàm regularization $R(\phi)$ khác cũng có thể được cài đặt và từ đó sẽ thay đổi lại hàm user behavior và có thể nghiệm tối ưu $\phi^\star$ sẽ không có công thức.
- Trường hợp khách hàng không chọn lựa sản phẩm nào luôn được mô hình hóa như một feature vector 0 hoặc phần thưởng tương ứng với việc không chọn sẽ được coi như 1 hằng số và có thể học.

7.2.2 Tham số hóa mô hình

Đầu tiên chúng ta sẽ định nghĩa trạng thái của khách hàng như sau

$$s^t := h\left(\boldsymbol{F}_*^{1:t-1} := \left[\boldsymbol{f}_*^1, \cdots, \boldsymbol{f}_*^{t-1}\right]\right),$$

với mỗi $\boldsymbol{f}_*^\tau \in \mathbb{R}^d$ là feature vector ứng với các item đã chọn ở thời điểm τ và $h(\cdot)$ là 1 hàm embedding. Chúng ta cũng có thể biểu diễn chuỗi M bước như sau:

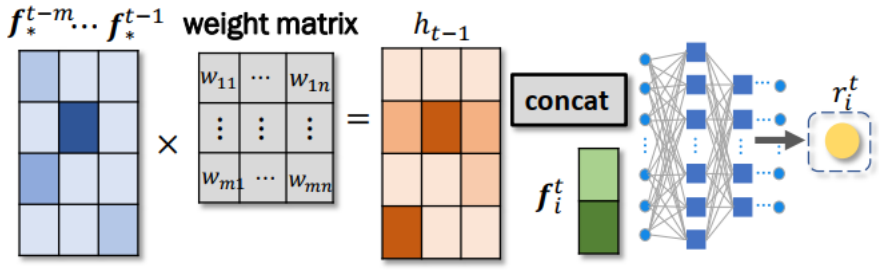
$$\boldsymbol{F}_*^{t-m:t-1} := \left[\boldsymbol{f}_*^{t-m}, \cdots, \boldsymbol{f}_*^{t-1}\right].$$

Chúng ta sử dụng các hàm embedding $h(\cdot)$ ở dạng đơn giản là sự kết hợp của các hàm trọng số tuyến tính và hàm kích hoạt phi tuyến như ReLU hay ELU. Đặt $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ là một ma trận có m tương ứng với số lần chọn của lịch sử và mỗi cột tương ứng với tập các trọng số quan trọng Hàm embedding $h \in \mathbb{R}^{dn \times 1}$ sẽ được thiết kế như sau:

$$\boldsymbol{s}^t = h\left(\boldsymbol{F}_*^{t-m:t-1}\right) := \text{vec}\left[\sigma\left(\boldsymbol{F}_*^{t-m:t-1}\boldsymbol{W} + \boldsymbol{B}\right)\right], \tag{18}$$

với $\boldsymbol{B} \in \mathbb{R}^{d \times n}$ là ma trận hệ số bias và $\sigma(\cdot)$ là các hàm phi tuyến như ReLU hay ELU, và vec$[\cdot]$ sẽ chuyển ma trận thành vector dài bằng việc nối các cột của ma trận với nhau. Hơn nữa, mạng LSTM hoàn toàn có thể được dùng để trích xuất thông tin từ chuỗi item đã chọn trong quá khứ nhưng sẽ tăng chi phí luyện mô hình do có nhiều tham số cần được học hơn.

Tiếp theo, chúng ta định nghĩa hàm phần thưởng và mô hình user behavior model. Một sự lựa chọn $a^t \in \mathcal{A}^t$ của khách hàng sẽ ứng với một item có feature vector $f_{a^t}^t$ do đó ta tham số hóa hàm phần thưởng $r(s^t, a^t)$ và $\phi(s, \mathcal{A}^t)$ bằng mạng neural như sau:



Hình 14: Mạng neural biểu diễn hàm phần thưởng

Tương ứng ta có công thức biểu diễn của r và ϕ như sau:

$$r\left(\boldsymbol{s}^t, a^t\right) := \boldsymbol{v}^\top \sigma\left(\boldsymbol{V}\left[\left(\boldsymbol{s}^t\right)^\top, \left(\boldsymbol{f}_{a^t}^t\right)^\top\right]^\top + \boldsymbol{b}\right) \text{ and}$$
$$\phi\left(\boldsymbol{s}, \mathcal{A}^t\right) \propto \exp\left(\boldsymbol{v}^\top \sigma\left(\boldsymbol{V}'\left[\left(\boldsymbol{s}^t\right)^\top, \left(\boldsymbol{f}_{a^t}^t\right)^\top\right]^\top + \boldsymbol{b}'\right)\right),$$

với $\boldsymbol{V}, \boldsymbol{V}' \in \mathbb{R}^{\ell \times (dn+d)}$ là các ma trận trọng số, $\boldsymbol{b}, \boldsymbol{b}' \in \mathbb{R}^{1\times(dn+d)}$ là các bias vectors và $\boldsymbol{v}, \boldsymbol{v}' \in \mathbb{R}^\ell$ là tham số hồi quy. Để giản lược ký hiệu, ta ký hiệu tập các tham số của hàm phần thưởng θ và tập các tham số của mô hình user behavior là α tương ứng với 2 ký hiệu r_θ và ϕ_α.

7.2.3 Huấn luyện theo phương pháp đối nghịch tạo sinh

Cả hàm phần thưởng $r\left(\boldsymbol{s}^t, a^t\right)$ và mô hình user behavior $\phi\left(s^t, \mathcal{A}^t\right)$ đều sẽ được ước lượng từ dữ liệu. Mô hình user behavior ϕ cố gắng bắt chước chuỗi hành động của người dùng để tối đa hóa phần thưởng r thu được. Từ ý tưởng của mạng đối nghịch tạo sinh ta sẽ thiết kế cách luyện sao cho:

- ϕ hoạt động như mạng sinh sẽ sinh ra hành động tiếp theo của khách hàng dựa trên lịch sử đã chọn của họ.
- r hoạt động như một mạng đối nghịch có tác dụng phân biệt hành động thật sự của khách hàng với hành động được sinh ra từ mạng sinh.

Dựa trên thiết kế của mạng GAN ta xấp xỉ ϕ và r thông qua hàm mini-max (19). Cụ thể, với chuỗi lịch sử chọn T với các hành động như sau: $\left\{a_{\text{true}}^1, a_{\text{true}}^2, \ldots, a_{\text{true}}^T\right\}$ tương ứng với các features vector $\left\{\boldsymbol{f}_*^1, \boldsymbol{f}_*^2, \ldots, \boldsymbol{f}_*^T\right\}$, ta sẽ học r và ϕ qua bài toán tối ưu mini-max như sau:

$$\min_\theta \max_\alpha \left(\mathbb{E}_{\phi_\alpha}\left[\sum_{t=1}^T r_\theta\left(s_{\text{true}}^t, a^t\right)\right] - R\left(\phi_\alpha\right)/\eta\right) - \sum_{t=1}^T r_\theta\left(s_{\text{true}}^t, a_{\text{true}}^t\right), \tag{19}$$

với s_{true}^t là trạng thái được trích xuất từ dữ liệu của khách hàng. Từ công thức (19) trên có thể nhận xét rằng hàm phần thưởng r_θ sẽ được xây dựng từ cả những hành động a^t của mô hình tạo ra và những hàng động a_{true}^t từ dữ liệu và cố gắng phân biệt giá trị của các phần thưởng đó. Ngược lại,

mô hình user behavior ϕ_α sẽ cố gắng tạo ra các hành động mô phỏng lại hành động chọn của khách hàng sao cho giá trị của các phần thưởng do mô hình tạo ra và phần thưởng khách hàng thực tế nhận được là giống nhau nhất. Điều này sẽ tăng tính khai phá của mô hình, giúp hệ thống học được chính sách tốt nhất cho mọi trường hợp.

Tùy vào cách thiết kế hàm regularization $R(\phi_\alpha)$ mà phương trình (19) có thể có công thức nghiệm tối ưu hoặc không có do tính không lồi của hàm tối ưu. Khi thiết kế hàm regularization theo kiểu negative Shannon entropy ta có kết quả như sau:

Bổ đề 7.2 *Nếu hàm regularization trong (19) được định nghĩa là* $R(\phi) = \sum_{i=1}^{k} \phi_i \log \phi_i$ *và* Φ *bao gồm tất cả các phân phối xác suất* $\mathcal{S} \times \begin{pmatrix} \mathcal{I} \\ k \end{pmatrix}$ *đến* Δ^{k-1}*, thì bài toán trong (19) sẽ tương đương bài toán ước lượng hợp lý cực đại sau:*

$$\max_{\theta \in \Theta} \prod_{t=1}^{T} \frac{\exp\left(\eta r_\theta\left(s_{true}^t, a_{true}^t\right)\right)}{\sum_{a^t \in \mathcal{A}^t} \exp\left(\eta r_\theta\left(s_{true}^t, a^t\right)\right)}.$$

Chứng minh: Kết quả của bổ đề (7.2) được suy ra trực tiếp từ bổ đề (**??**). Bài toán tối ưu là:

$$\min_{\theta \in \Theta} \left(\max_{\phi \in \Phi} \mathbb{E}_\phi \left[\sum_{t=1}^{T} r_\theta\left(\boldsymbol{s}_{\text{true}}^t, a^t\right) \right] - \frac{1}{\eta} R(\phi) \right) - \sum_{t=1}^{T} r_\theta\left(\boldsymbol{s}_{\text{true}}^t, a_{\text{true}}^t\right)$$

Chúng ta giả sử không có cặp (s_{true}^t, a^t) nào bị trùng lặp giá trị trong (19). Điều này hoàn toàn hợp lý do s_{true}^t được cập nhật qua mọi bước lặp và a^t được biểu diễn thông qua các giá trị feature vector $\boldsymbol{f}_{a^t}^t$, trong không gian $\mathbb{R}^d$. Với giả sử này, ϕ ánh xạ mỗi cặp (s_{true}^t, a^t) sang nghiệm tối ưu ϕ^{t*} cực đại $r_\theta\left(s_{\text{true}}^t, a^t\right) - \frac{1}{\eta} R\left(\phi^t\right)$ vì không có các giá trị trùng lặp. Sử dụng kết quả của bổ đề (16), ta có:

$$\max_{\phi \in \Phi} \mathbb{E}_\phi \left[\sum_{t=1}^{T} r_\theta\left(s_{\text{true}}^t, a^t\right) \right] - \frac{1}{\eta} R(\phi) = \max_{\phi \in \Phi} \sum_{t=1}^{T} \mathbb{E}_\phi \left[r_\theta\left(s_{\text{true}}^t, a^t\right) \right] - \frac{1}{\eta} R(\phi)$$
$$= \sum_{t=1}^{T} \left(\sum_{i=1}^{k} \phi_i^{t*} r\left(s^t, a_i\right) - \frac{1}{\eta} \sum_{i=1}^{k} \phi_i^{t*} \log \phi_i^{t*} \right) = \sum_{t=1}^{T} \frac{1}{\eta} \log \left(\sum_{i=1}^{k} \exp\left(\eta r_\theta\left(s_{\text{true}}^t, a_i\right)\right) \right)$$

Phương trình (19) tương đương với:

$$\min_{\theta \in \Theta} \sum_{t=1}^{T} \frac{1}{\eta} \log \left(\sum_{i=1}^{k} \exp\left(\eta r_\theta\left(s_{\text{true}}^t, a_i\right)\right) \right) - \sum_{t=1}^{T} r_\theta\left(s_{\text{true}}^t, a_{\text{true}}^t\right)$$

là dạng hàm negative log-likelihood tương đương.

7.3 Xây dựng cascading policy để luyện mô hình

Khi đã xây dựng được mô hình user behavior và hàm phần thưởng tương ứng ta sẽ sử dụng các phương pháp cải thiện chính sách của RL để tạo ra chính sách tối ưu cho RS. Do cách mô hình

hóa không gian trạng thái dưới dạng tổ hợp $\begin{pmatrix} \mathcal{I} \\ k \end{pmatrix}$, nghĩa là ta cần chọn ra k item từ tập $\mathcal{I}$. Chúng ta cần thiết kế mạng Q giá trị hành động sao cho đem lại hiệu quả tính toán khi duyệt qua toàn bộ không gian trạng thái, trong bài báo [21], họ đã thiết kế ra mạng Cascading Q.

7.3.1 Cascading Q-network

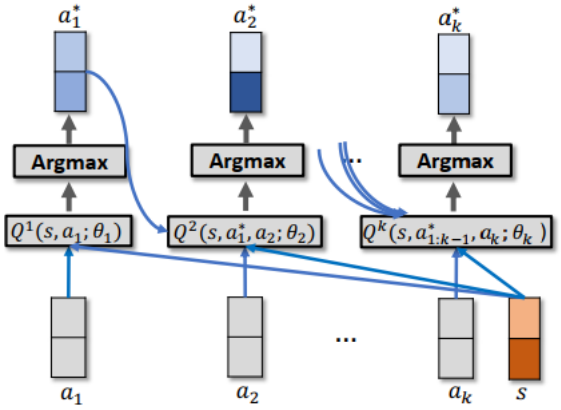
Sử dụng ý tưởng của mô thức Q-learning trong việc tìm giá trị tối ưu cho hàm action-value $Q^*(s, \mathcal{A})$ sẽ được học thông qua phương trình $Q^*\left(s^t, \mathcal{A}^t\right) = \mathbb{E}\left[r\left(s^t, \mathcal{A}^t, a^t\right) + \gamma \max_{\mathcal{A}' \subset \mathcal{I}} Q^*\left(\boldsymbol{s}^{t+1}, \mathcal{A}'\right)\right]$, $a^t \in \mathcal{A}^t$. Khi các hàm action-value đã được học, chính sách tối ưu tương ứng là:

$$\pi^*\left(s^t, \mathcal{I}^t\right) = \arg\max_{\mathcal{A}^t \subset \mathcal{I}^t} Q^*\left(s^t, \mathcal{A}^t\right), \tag{20}$$

với $\mathcal{I}^t \subset \mathcal{I}$ là tập các items có sẵn tại thời điểm t. Không gian các hành động có dạng $\begin{pmatrix} K \\ k \end{pmatrix}$ có thể lớn vô cùng với giá trị của K (ví dụ 1000) và k (ví dụ 5). Hơn nữa, một item đặt trong tập hành động khác nhau có thể có xác suất được chọn khác nhau, do đó chính sách trong (20) có chi phí tính toán rất lớn. Do đó mạng cascading Q-network sẽ gồm k hàm Q-functions liên kết với nhau để tìm nghiệm tối ưu cho (20).

Ký hiệu các hành động do hệ khuyến nghị là $\mathcal{A} = \{a_{1:k}\} \subset \mathcal{I}$ và tập các hành động tối ưu là $\mathcal{A}^* = \{a^*_{1:k}\} = \arg\max_{\mathcal{A}} Q^*(s, \mathcal{A})$. Mạng cascading Q-networks được thiết kế dựa trên ý tưởng:

$$\max_{a_{1:k}} Q^*\left(s, a_{1:k}\right) = \max_{a_1}\left(\max_{a_{2:k}} Q^*\left(s, a_{1:k}\right)\right).$$



Hình 15: Cascading Q-network

Chúng ta sẽ thiết kế các Q-function thành phần $Q^{1*}, \ldots, Q^{k*}$ và tìm các hành động tối ưu a_j^*

tương ứng là:

$$
\begin{aligned}
a_1^* &= \arg\max_{a_1} \left\{ Q^{1*}\left(s, a_1\right) := \max_{a_{2:k}} Q^*\left(s, a_{1:k}\right) \right\} \\
a_2^* &= \arg\max_{a_2} \left\{ Q^{2*}\left(s, a_1^*, a_2\right) := \max_{a_{3:k}} Q^*\left(s, a_{1:k}\right) \right\} \\
&\cdots \\
a_k^* &= \arg\max_{a_k} \left\{ Q^{k*}\left(s, a_{1:k-1}^*, a_k\right) := Q^*\left(s, a_{1:k}\right) \right\}
\end{aligned}
$$

Thuật toán tìm các hành động a_j^* tối ưu như sau:

Algorithm 1 Recommend using Q^j Cascades

Let $\mathcal{A}^* = \emptyset$ be empty, remove clicked items $\mathcal{I} = \mathcal{A} \setminus \boldsymbol{s}$
For $j = 1$ *to* k **do**
  $a_j^* = \arg\max_{a_j \in \mathcal{I} \setminus \mathcal{A}^*} Q^j(\boldsymbol{s}, a_{1:j-1}^*, a_j; \Theta_j)$
  Update $\mathcal{A}^* = \mathcal{A}^* \cup \{a_j^*\}$
return $\mathcal{A}^* = (a_1^*, \cdots, a_k^*)$

Độ phức tạp tính toán của mạng cascading Q-network là $O(k|\mathcal{I}|)$. Tuy nhiên, các hàm Q^{j*} thường phải được ước lượng từ dữ liệu.

7.3.2 Tham số hóa và xấp xỉ

Mỗi hàm Q^{j*} được tham số hóa bằng một mạng neural như sau:

$$
\boldsymbol{q}_j^\top \sigma\left(\boldsymbol{L}_j \left[\boldsymbol{s}^\top, \boldsymbol{f}_{a_1^*}^\top, \ldots, \boldsymbol{f}_{a_{j-1}^*}^\top, \boldsymbol{f}_{a_j}^\top\right]^\top + \boldsymbol{c}_j\right), \forall j,
$$

với $\boldsymbol{L}_j \in \mathbb{R}^{\ell \times (dn+dj)}, \boldsymbol{c}_j \in \mathbb{R}^\ell$ và $\boldsymbol{q}_j \in \mathbb{R}^\ell$ gọi chung là tập Θ_j và các trạng thái s được định nghĩa như ở (18). Sau đó ta cần huấn luyện mô hình để tìm được bộ tham số tối ưu Θ_j sao cho các hàm Q^{j*} thỏa mãn các điều kiện ràng buộc sau:

$$
Q^{j*}\left(s, a_1^*, \cdots, a_j^*\right) = Q^*\left(s, a_1^*, \cdots, a_k^*\right), \quad \forall j. \tag{21}
$$

Việc tìm nghiệm tối ưu thỏa mãn (21) là tương đối khó do đó ta sẽ đi tìm nghiệm của bài toán tối ưu nới lỏng sau:

$$
\begin{aligned}
&\min \left(y - Q^j\right)^2 \\
&\text{v.đ.k } y = r\left(s^t, \mathcal{A}^t, a^t\right) + \gamma Q^k\left(\boldsymbol{s}^{t+1}, a_{1:k}^*; \Theta_k\right), \forall j.
\end{aligned}
$$

Các hàm Q^j đều được tối ưu với cùng hàm y do đó bộ tham số Θ_k có thể được cập nhật thông qua gradient descent. Cách cập nhật được trình bày chi tiết trong thuật toán sau

Algorithm 2 cascading deep Q-learning (CDQN) with Experience Replay

Initialize replay memory $\mathcal{M}$ to capacity N

Initialize parameter Θ_j of $\widehat{Q^j}$ with random weights for each $1 \leq j \leq k$ **For** *iteration* $i = 1$ *to* L **do**

  Sample a batch of users $\mathcal{U}$ from training set Initialize the states $\boldsymbol{s}^0$ to a zero vector for each $u \in \mathcal{U}$ **For** $t = 1$ *to* T **do**

    For *each user* $u \in \mathcal{U}$ *simultaneously* **do**

      With probability ε select a random subset $\mathcal{A}^t$ of size k

      Otherwise, $\mathcal{A}^t = \text{ARGMAX\_Q}(\boldsymbol{s}_u^t, \mathcal{I}^t, \Theta_1, \cdots, \Theta_k)$

      Recommend $\mathcal{A}^t$ to user u, observe user action $a^t \sim \phi(\boldsymbol{s}^t, \mathcal{A}^t)$ and update user state $\boldsymbol{s}^{t+1}$

      Add tuple $\left(\boldsymbol{s}^t, \mathcal{A}^t, r(\boldsymbol{s}^t, a^t), \boldsymbol{s}^{t+1}\right)$ to $\mathcal{M}$

    Sample random minibatch $B \overset{\text{iid.}}{\sim} \mathcal{M}$

    For each j, update Θ_j by SGD over the loss $\left(y - \widehat{Q^j}(\boldsymbol{s}^t, A_{1:j}^t; \Theta_j)\right)^2$ for B

return $\Theta_1, \cdots, \Theta_k$

8 Kết quả thực nghiệm

Trong phần này, nhóm tác giả sẽ xây dựng một số hệ thống khuyến nghị dựa trên code của các paper nghiên cứu về RS và tiến hành so sánh kết quả của các mô hình đó với mô hình [21] chính. Các mô hình sẽ được cài đặt bằng ngôn ngữ Python và sử dụng bộ dữ liệu MovieLen 100K để huấn luyện các mô hình.

8.1 Dữ liệu sử dụng

Trong báo cáo này, nhóm tác giả sẽ sử dụng bộ dữ liệu MovieLen 100K (`https://grouplens.org/datasets/movielens/`) gồm 100000 ratings của 600 khách hàng khi xem phim cho 9000 phim khác nhau để làm tập dữ liệu huấn luyện cho các mô hình. Dữ liệu gồm các thông tin của user (người xem phim) và thông tin về ratings cho các bộ phim.

Thông tin về khách hàng

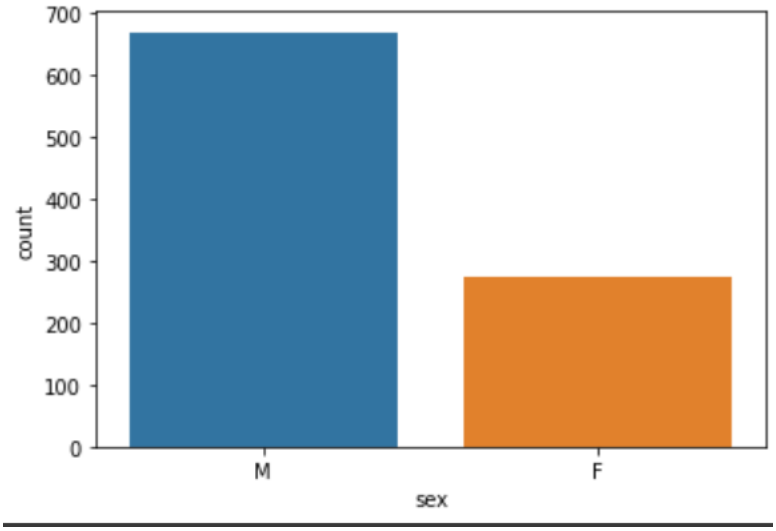
Các trường quan trọng của dữ liệu về User được sử dụng là: user id, tuổi, giới tính, nghề nghiệp và zip code.

	user_id	age	sex	occupation	zip_code
0	0	24	M	technician	85711
1	1	53	F	other	94043
2	2	23	M	writer	32067
3	3	24	M	technician	43537
4	4	33	F	other	15213
...	...	...	...	...	...
938	938	26	F	student	33319
939	939	32	M	administrator	02215
940	940	20	M	student	97229
941	941	48	F	librarian	78209
942	942	22	M	student	77841

943 rows × 5 columns

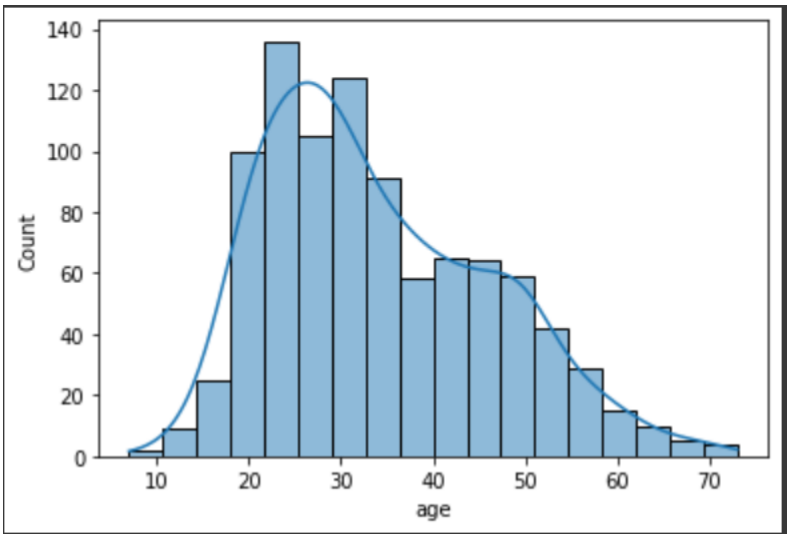
Hình 16: Thông tin về User

Tiến hành khai phá dữ liệu bằng một vài bước đơn giản, ta thấy tỷ lệ giới tính của khách hàng chủ yếu là nam với 670 khách hàng trên tổng số 900 khách hàng.



Hình 17: Tỷ lệ giới tính

Lứa tuổi tham gia đánh giá đa dạng nhưng chủ yếu nằm trong độ tuổi về 20-30 điều này sẽ ảnh hưởng đến xu hướng đánh giá phim của bộ dữ liệu do thị hiếu của khách hàng trong độ tuổi này sẽ khác so với các khách hàng lớn tuổi hơn.



Hình 18: Phân phối tuổi của khách hàng

Thông tin về rating phim

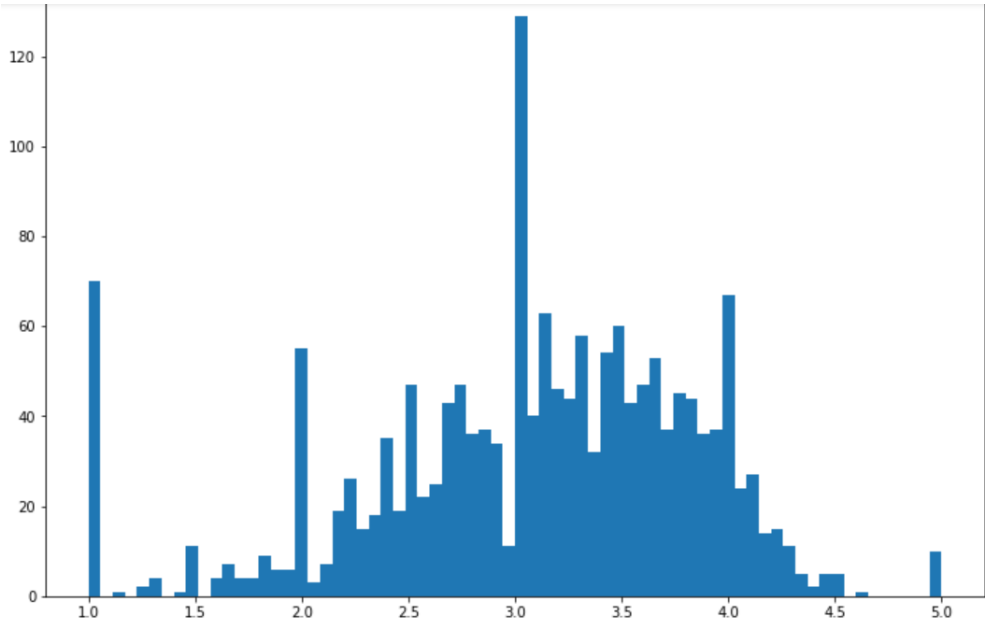
Một số trường chính của dữ liệu về các rating của khách hàng là user id, movie id, rating và timestamp (mốc thời gian đánh giá đã được embedding)

	user_id	movie_id	rating	unix_timestamp
0	195	241	3.0	881250949
1	185	301	3.0	891717742
2	21	376	1.0	878887116
3	243	50	2.0	880606923
4	165	345	1.0	886397596
...	...	...	...	...
99995	879	475	3.0	880175444
99996	715	203	5.0	879795543
99997	275	1089	1.0	874795795
99998	12	224	2.0	882399156
99999	11	202	3.0	879959583

100000 rows × 4 columns

Hình 19: Dữ liệu rating của khách hàng

Phân phối về ratings cho các bộ phim được miêu tả ở hình dưới, tỷ lệ đánh giá phim ở mức điểm 3-3.5 cao nhất



Hình 20: Phân phối cho Rating

8.2 Kết quả chạy mô hình

Kết quả thực nghiệm và code cài đặt mô hình được upload tại link `https://github.com/nhatt1k26/GANRL`. Trong báo cáo này, nhóm tác giả cài đặt mô hình chính GAN-PW và GAN-LSTM của [21] và tiến hành so sánh kết quả với các mô hình sau: Mô hình 2 Tower đơn giản do các tác giả tự xây dựng dựa trên [25], mô hình Bilateral Variational Autoencoder (BiVaE) [26], mô hình RS Hybrid dựa trên package LightFM [27], mô hình Neural Collaborative Filtering (NCF) [28] và mô hình Bayesian Personalized Ranking (BPR) [29].

Ta sử dụng Top-k precision (Prec k) là độ đo đánh giá. Độ đo này đo tỷ lệ top-k items được xếp hạng cao nhất từ mô hình ở mỗi page view so với các items mà khách hàng thật sự chọn, sau đó lấy trung bình trên toàn bộ tập dữ liệu và page views. Kết quả được tổng hợp trong bảng dưới: Ta

MovieLens	
Model	prec(%)@10
2 Tower	8.57 (±1.9)
NCF	18.253 (±1.9)
LightFM	13.16 (±1.9)
BPR	35.98 (±1.9)
BiVaE	40.81(±2.0)
GAN-PW	80.6 (±0.7)
GAN-LSTM	**87.4** (±0.5)

Bảng 3: Kết quả chạy mô hình

có thể thấy mô hình GAN-PW và GAN-LSTM cho kết quả vượt trội hơn hẳn so với các mô hình còn lại.

9 Kết luận

Trong báo cáo trên, nhóm tác giả đã đạt được một số kết quả sau:

- Trình bày được khái quát lý thuyết của hệ thống khuyến nghị, mạng GAN và trình bày đầy đủ các nội dung lý thuyết cần có của Học tăng cường.
- Tiến hành khảo sát về ứng dụng của RL cho việc xây dựng RS.
- Tổng hợp một số bài báo tích hợp mạng GAN cho RL-based RS.
- Tiến hành cài đặt, đánh giá thử nghiệm các mô hình.

Tài liệu tham khảo

[1] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4902–4909, 2019.

[2] Xueying Bai, Jian Guan, and Hongning Wang. A model-based reinforcement learning with adversarial training for online recommendation. *Advances in Neural Information Processing Systems*, 32, 2019.

[3] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[4] Richard S Sutton, Andrew G Barto, et al. Introduction to reinforcement learning. 1998.

[5] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

[6] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. Citeseer, 1994.

[7] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[8] Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard L Lewis, and Xiaoshi Wang. Deep learning for real-time atari game play using offline monte-carlo tree search planning. *Advances in neural information processing systems*, 27, 2014.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[10] Yuanguo Lin, Yong Liu, Fan Lin, Pengcheng Wu, Wenhua Zeng, and Chunyan Miao. A survey on reinforcement learning for recommender systems. *arXiv preprint arXiv:2109.10665*, 2021.

[11] M Mehdi Afsar, Trafford Crump, and Behrouz Far. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys (CSUR)*, 2021.

[12] Thorsten Joachims, Dayne Freitag, Tom Mitchell, et al. Webwatcher: A tour guide for the world wide web. In *IJCAI (1)*, pages 770–777. Citeseer, 1997.

[13] Yang Zhang, Chenwei Zhang, and Xiaozhong Liu. Dynamic scholarly collaborator recommendation via competitive multi-agent reinforcement learning. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 331–335, 2017.

[14] Sungwoon Choi, Heonseok Ha, Uiwon Hwang, Chanju Kim, Jung-Woo Ha, and Sungroh Yoon. Reinforcement learning based recommender system using biclustering technique. *arXiv preprint arXiv:1801.05532*, 2018.

[15] Omar Moling, Linas Baltrunas, and Francesco Ricci. Optimal radio channel recommendations with explicit and implicit feedback. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 75–82, 2012.

[16] Elad Liebman, Maytal Saar-Tsechansky, and Peter Stone. Dj-mc: A reinforcement-learning agent for music playlist recommendation. *arXiv preprint arXiv:1401.1880*, 2014.

[17] Binbin Hu, Chuan Shi, and Jian Liu. Playlist recommendation based on reinforcement learning. In *International Conference on Intelligence Science*, pages 172–182. Springer, 2017.

[18] Tariq Mahmood and Francesco Ricci. Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 73–82, 2009.

[19] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. 2011.

[20] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. An mdp-based recommender system. *Journal of Machine Learning Research*, 6(9), 2005.

[21] Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, Yuan Qi, and Le Song. Generative adversarial user model for reinforcement learning based recommendation system. In *International Conference on Machine Learning*, pages 1052–1061. PMLR, 2019.

[22] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[23] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[24] Yoon Kim. Convolutional neural networks for sentence classification in: Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp), 1746–1751.. acl. *Association for Computational Linguistics, Doha, Qatar*, 2014.

[25] Chi Zhao and Ivan Blekanov. Two towers collaborative filtering algorithm for movie recommendation. 8(1):397–401, 2021.

[26] Quoc-Tuan Truong, Aghiles Salah, and Hady W Lauw. Bilateral variational autoencoder for collaborative filtering. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 292–300, 2021.

[27] Maciej Kula. Metadata embeddings for user and item cold-start recommendations. In Toine Bogers and Marijn Koolen, editors, *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015.*, volume 1448 of *CEUR Workshop Proceedings*, pages 14–21. CEUR-WS.org, 2015.

[28] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[29] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.