Insper

Using Reinforcement Learning for Recommendation Systems

Matheus Silva Melo de Oliveira

Professor: Fabrício Barth

São Paulo, 2023

Contexto e Objetivo

Este projeto tem por objetivo utilizar Reinforcement Learning (aprendizagem por reforço) para sistemas de recomendação. No escopo específico deste projeto foi construído um agente que recomende uma dada quantidade de filmes depois de uma maior quantidade inicial de filmes escolhidos pelo usuário.

Dessa maneira, este projeto se baseou no estudo e re-implementação de agentes de Reinforcement Learning existentes como *Recsim, RecNN, R4LRS e Recsys-RL.* Foram analisadas as técnicas utilizadas para criação dos agentes, e após treinamento, criou-se um arquivo que permite ao agente recomendar filmes dada a escolha inicial de filmes do gosto do usuário.

Destarte, este projeto consiste na análise e estudo de técnicas de Reinforcement Learning para sistemas de recomendação, analisando a eficácia de execução do agente para diferentes filmes, métricas de eficiência, prós e contras.

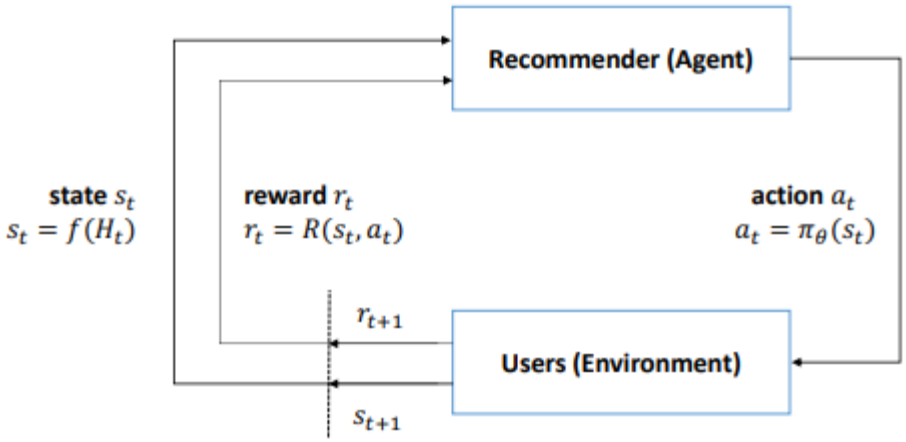
Método Utilizado

Segundo Liu[1], em geral, sistemas de recomendação são considerados como problema de tomada de decisão sequencial. Em geral, um agente irá recomendar para um *enviroment* (usuário que deseja recomendação) para sugerir uma quantidade sequencial de filmes após a inteiração inicial deste. O objetivo desse modelo se torna então maximizar a satisfação ou chance de o usuário assistir ao filme (conceito traduzido tecnicamente como *reward)*. Nesse contexto, define-se:

- O estado é um vetor tridimensional que usa as últimas escolhas dadas pelo usuário (no início da inteiração) e pelas últimas N recomendações dadas pelo agente;
- A ação trata-se de um vetor unidimensional;
- O score de avaliação atrelado a um filme advém do produto escalar entre a ação tomada pelo agente e avaliação intrínseca dos filmes na plataforma;
- A recompensa é armazenada na matriz de avaliação-usuário, sendo de valor 1 se a classificação retornada da matriz é maior que três e zero se menor que isso);

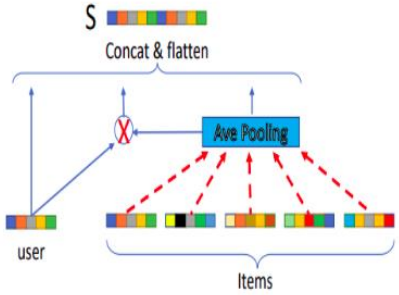
Já o ambiente, tem por definição e suas características:

- Espaço observacional: Vetor tridimensional de todas as últimas N recomendações dadas ao usuário;
- Espaço de ação: Uma ação é o vetor contínuo denotado com todos os itens (filmes) ainda não recomendados anteriormente ao usuário;



Dessa forma, dado o estado representado obtemos a avaliação relativa à ação, calculamos o produto escalar entre esta avaliação e as avaliações de todos os itens no espaço de ação, seleciona-se 1 item de classificação superior, calcula-se a recompensa, atualizamos os itens visualizados e a memória e armazenamos a transição no buffer-relay.

O estado então é representado seguindo o seguinte esquema:



No qual tem-se concatena-se e agrupa-se o produto de Hadamard das recomendações iniciais ou padrões do usuário com os itens já avaliados pelo agente calculando a média para cada patch do mapa de recursos via camada de *Average Pooling* de uma rede neural convolucional, gerando assim o vetor tridimensional descrito anteriormente.

Com a representação do estado em mãos, o modelo de processamento opera em um algoritmo de Deep Deterministic Policy Gradient,escolhido aqui pois este foi projetado para resolver problemas de controle contínuo, onde as ações são representadas por um espaço contínuo, como avaliações. Foram comparadas metodologias com e sem o processo de Ornstein-Uhlenbeck.

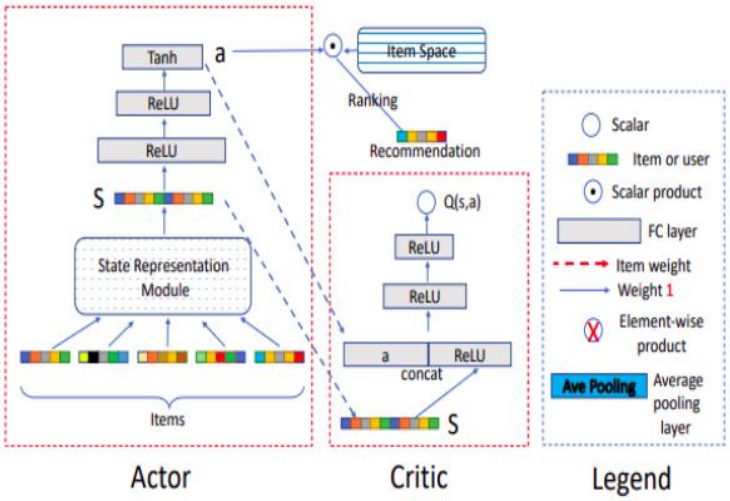
O modelo DDPG[2] se trata de um modelo off-policy e online, e utiliza cerca de quatro redes neurais para conseguir processar com maior estabilidade e menores resíduos de oscilação, visto que os valores de atualização da rede, são interdependentes dos valores calculados na própria rede:

This depends Q function itself (at the moment it is being optimized)

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a) + \gamma max Q(s',a') - Q(s,a)]$$

Essas redes, seguem a arquitetura Ator e crítico, estrutura comumente usada em RL, onde um agente aprende a melhor política (Actor) e também avalia o valor dessa política (Critic). O ator é responsável por selecionar ações com base nas observações do ambiente, enquanto o crítico avalia o quão bom é o estado atual e as ações tomadas pelo ator. No DDPG, essa arquitetura é ampliada com redes neurais profundas para representar o ator e o crítico. O ator é uma rede neural que mapeia observações de entrada para ações contínuas. Essa rede neural é treinada para maximizar o valor esperado dos retornos cumulativos ao longo do tempo. O crítico também é uma rede neural que recebe como entrada o estado e a ação tomada pelo ator e retorna uma estimativa do valor esperado desse estado-ação. Por usar Deep-learning, utiliza-se a técnica de replay-buffer para atualizar os pesos da rede em batches ao invés de momentâneos.

Em específico, a arquitetura utilizada nessa implementação [1], utiliza três layers para a rede Actor e duas principais para a Critc, obtém-se a recomendação para um dado estado e um espaço de itens para o usuário, advindo da ação da rede Actor, enquanto a rede Critic (uma Deep-Q-Network) analisa o mérito,isto é, a "validade" da ação escolhida pela Actor no atual Estado representado, atualizando de acordo com os valores calculados os pesos da rede Actor, melhorando a tomada de ações.



Assim, a função de recompensa baseia-se no score (valor discreto entre 0 e 5) dado por um usuário ao item recomendado pelo agente (obtido pelos dados) e é definida a seguir:

$$R(s,a) = \frac{1}{2}(rate_{i,j} - 3)$$

Dessa forma, em termos gerais, a implementação seguiu o seguinte pseudo-algoritmo:

input : Actor learning rate η_a, Critic learning rate η_c, discount factor γ, batch size N, state window size n and reward function R

Randomly initialize the Actor π_θ and the Critic Q_ω with parameters θ and ω

Initialize the target network π' and Q' with weights $\theta' \leftarrow \theta$ and $\omega' \leftarrow \omega$

Initialize replay buffer D

for *session = 1, M* **do**

  Observe the initial state s_0 according to the offline log

  for *t = 1, T* **do**

    Observe current state $s_t = f(H_t)$, where $H_t = \{i_1, ..., i_n\}$

    Find action $a_t = \pi_\theta(s_t)$ according to the current policy with ε-greedy exploration

    Recommended item i_t according to action a_t by Eq. (2)

    Calculate reward $r_t = R(s_t, a_t)$ based on the feedback of the user

    Observe new state $s_{t+1} = f(H_{t+1})$, where $H_{t+1} = \{i_2, ..., i_n, i_t\}$ if r_t is positive, otherwise, $H_{t+1} = H_t$

    Store transition (s_t, a_t, r_t, s_{t+1}) in D

    Sample a minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) in D with prioritized experience replay sampling technique

    Set $y_i = r_i + \gamma Q_{\omega'}(s_{i+1}, \pi_{\theta'}(s_{i+1}))$

    Update the Critic network by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q_\omega(s_i, a_i))^2$

    Update the Actor network using the sampled policy gradient:
    $\nabla_\theta J(\pi_\theta) \approx \frac{1}{N}\sum_t \nabla_a Q_\omega(s, a)|_{s=s_t, a=\pi_\theta(s_t)} \nabla_\theta \pi_\theta(s)|_{s=s_t}$

    Update the target networks:
    $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
    $\omega' \leftarrow \tau\omega + (1-\tau)\omega'$

return θ and ω

Resultados

Após implementação, foram testadas duas métricas:

- Métrica de hit, proporção de recomendações em que um ou mais itens relevantes estão presentes na lista de recomendação fornecida pelo sistema
- Discount Cumulative Gain, soma dos valores de relevância classificados de todos os resultados em uma lista de resultados recomendados.

Foram obtidos respectivamente, 50% de hit-cache e 28% de DCG, o que indica que apesar do agente conseguir recomendar com um certo discernimento, uma base de treinamento melhor e táticas de implementação mais robustas podem melhorar a eficácia deste.

Considerações finais

Conforme afirma-se no artigo referência desta re-implementação e também na implementação original, analisa-se que apesar de técnicas de Reinforcement Learning conseguirem se adequar para conjuntos de dados desbalanceados e até mesmo não variados, ele ainda carece de técnicas mais robustas para criar agentes e modelos que consigam resultados mais próximos de recomendações adequadas para usuários. Assim, nota-se que apesar de promissora, o uso deste tipo de técnicas

ainda não é totalmente indicada no momento para construção de modelos de recomendação robustos.

Referências

[1] LIU, F. et al. **Deep Reinforcement Learning based Recommendation with Explicit User-Item Interactions Modeling**. [s.l: s.n.]. Disponível em: <https://arxiv.org/pdf/1810.12027.pdf>. Acesso em: 24 maio. 2023.

[2] YOON, C. **Deep Deterministic Policy Gradients Explained**. Disponível em: <https://towardsdatascience.com/deep-deterministic-policy-gradients-explained-2d94655a9b7b>.

[3] BAJAJ, A. **Performance Metrics in Machine Learning [Complete Guide]**. Disponível em: <https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide>.