Using Reinforcement Learning for Recommendation Systems

Matheus Oliveira

Objetivos

- Estudar e re-implementar modelos de RL utilizados para sistemas de recomendação de filmes;
- Validar implementação para um conjunto de dados de teste aleatório;
- Analisar métricas de avaliação do agente;
- Entender prós e contras de RL para sistemas de recomendação;

Frameworks existentes

- RECSIM
- RecNN
- R4LRS
- Recsys-RL

Método utilizado

- Segundo Liu[1], em geral, sistemas de recomendação são considerados como problema de tomada de decisão sequencial (ambientes contínuos).
- 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).

Definindo o ambiente

Traditional recommendation task can be treated as sequental desicion making problem. Recommender (i.e. agent) interacts with users (i.e. environment) to sequentally suggest set of items. The goal is to maximize clients' satisfaction (i.e. reward). More specifically:

- State is a vector $a \in R^{3 \cdot embedding\_dim}$* computed using the user embedding and the embeddings of N latest positive interactions.
- Action is a vector $a \in R^{embedding\_dim}$. To get ranking score we took dot product of the action and the item embedding.

*An embedding is a relatively low-dimensional space into which you can translate high-dimensional vectors.

Definindo o ambiente

- 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 (depende de N) anteriormente ao usuário;

Definindo o ambiente

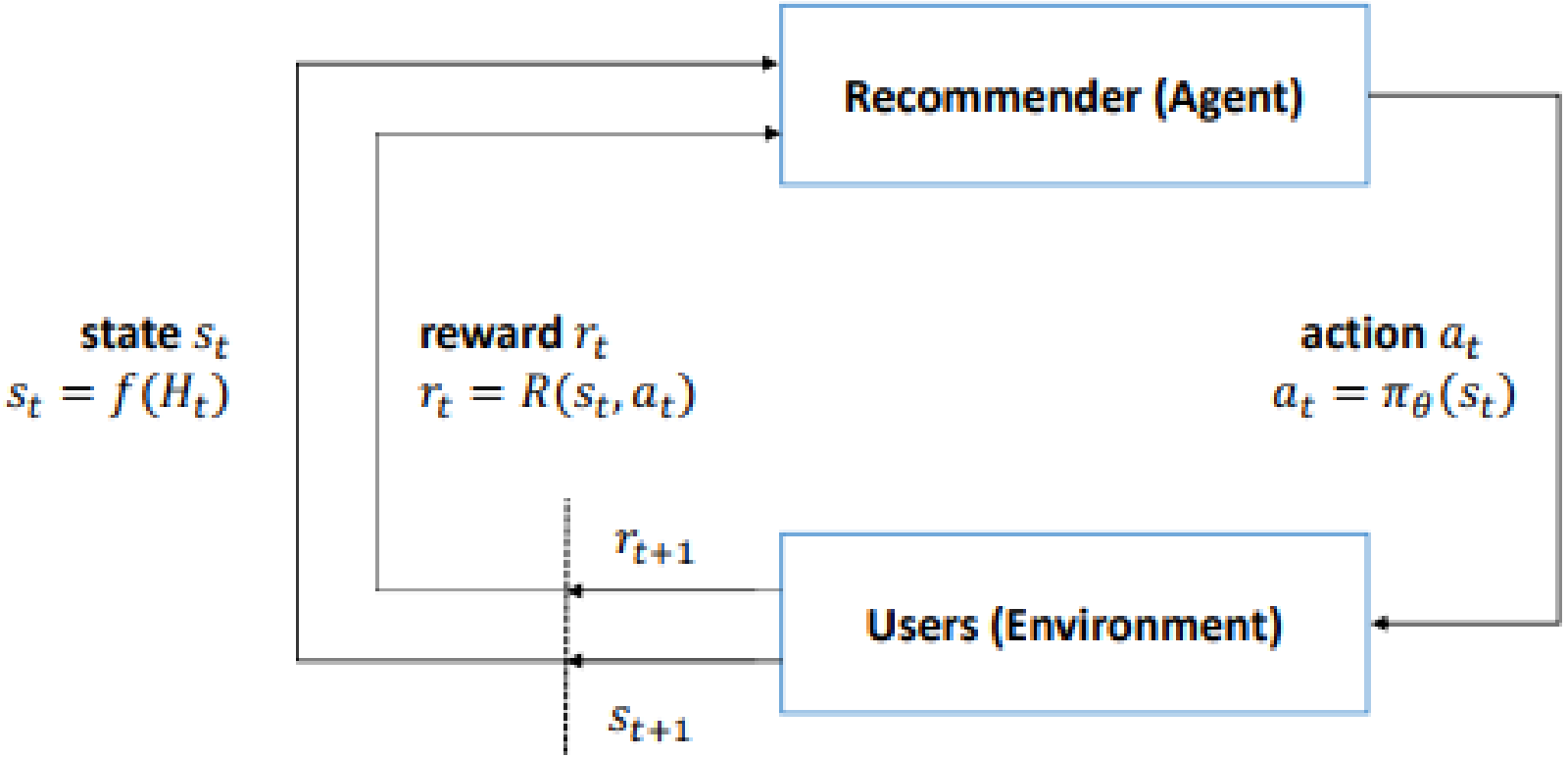
Função de recompensa:

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

Onde rate é definido como a nota dada por um usuário i a um item j recomendado pelo agente.
Este rate é um número discreto de 0 a 5, e é obtido em uma tabela que mapeia a avaliação de diferentes usuários para diversos filmes.

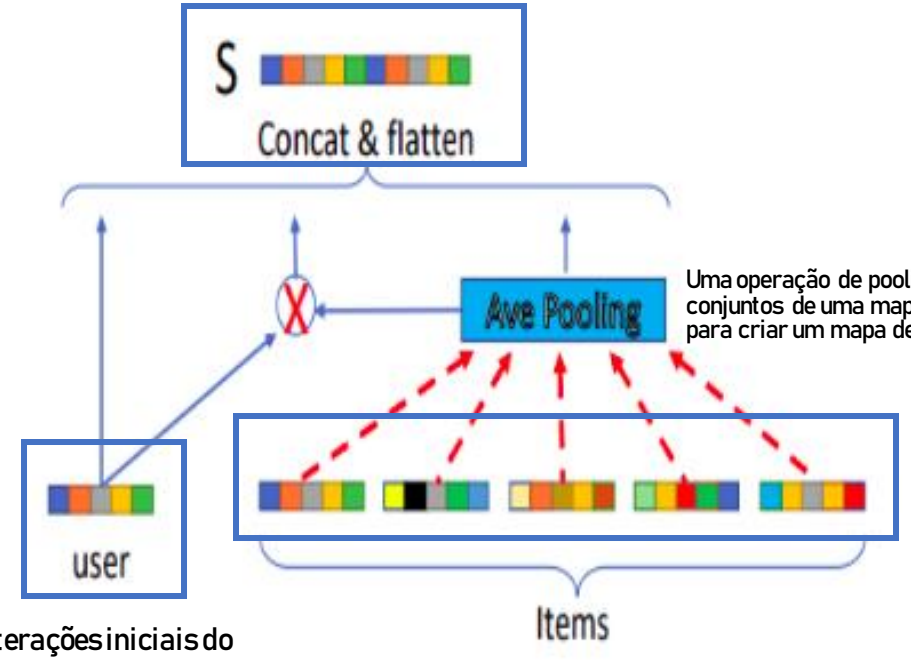
Definindo o ambiente

Interação agente-ambiente:

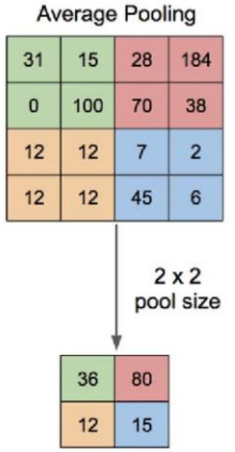


Representando o estado

Estado é representado como a multiplicação de Hadarmad (item a item), concatenando o vetor resultado e o reduzindo dimensionalmente para dimensão 3.



Uma operação de pooling que calcula o valor médio para conjuntos de uma mapa de características e o utiliza para criar um mapa de características pooled.



Interações iniciais do usuário com o ambiente

Filmes ainda não recomendados ao usuário

Algoritmo utilizado:

Deep Deterministic Gradient Policy

São projetados para aprender políticas determinísticas em tarefas de controle contínuo. Ao contrário das políticas estocásticas, que geram uma distribuição de probabilidade sobre ações, as políticas determinísticas mapeiam diretamente estados para ações específicas.

Os algoritmos DPG visam encontrar uma política determinística ótima que maximize o valor de retorno esperado, utilizando redes neurais para este processo.

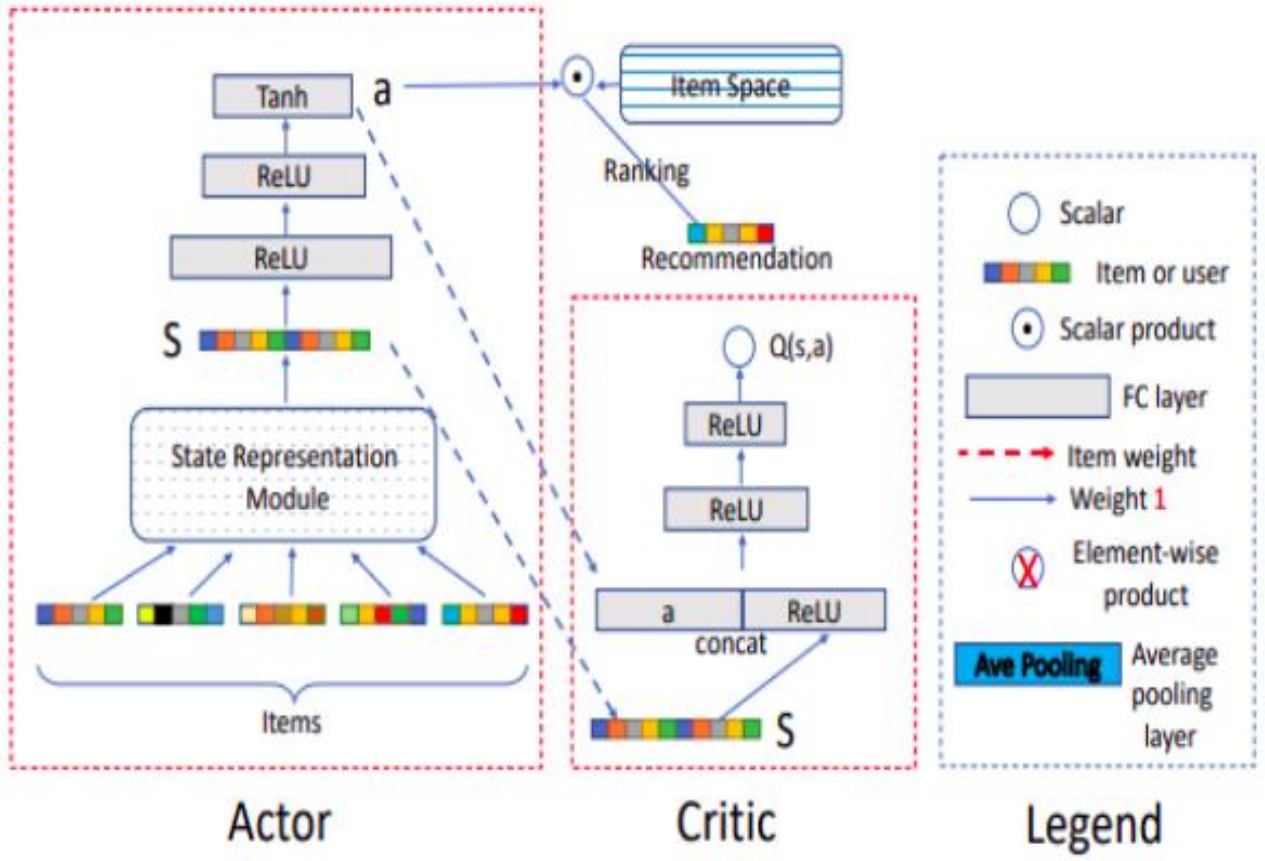
Ações são representadas por um espaço contínuo, como avaliações.

Arquitetura do algoritmo

Actor/Policy Network:

Para um dado usuário (env), a rede estima uma ação (recomendação de filme) de acordo com estado representado atualmente.

Isto é, para as ações iniciais de input do usuário em conjunto aos filmes ainda não recomendados ao usuário, estima-se qual a melhor ação a ser tomada, procurando a maximização de chance do usuário gostar da recomendação (maior rating) e portanto, maior recompensa.



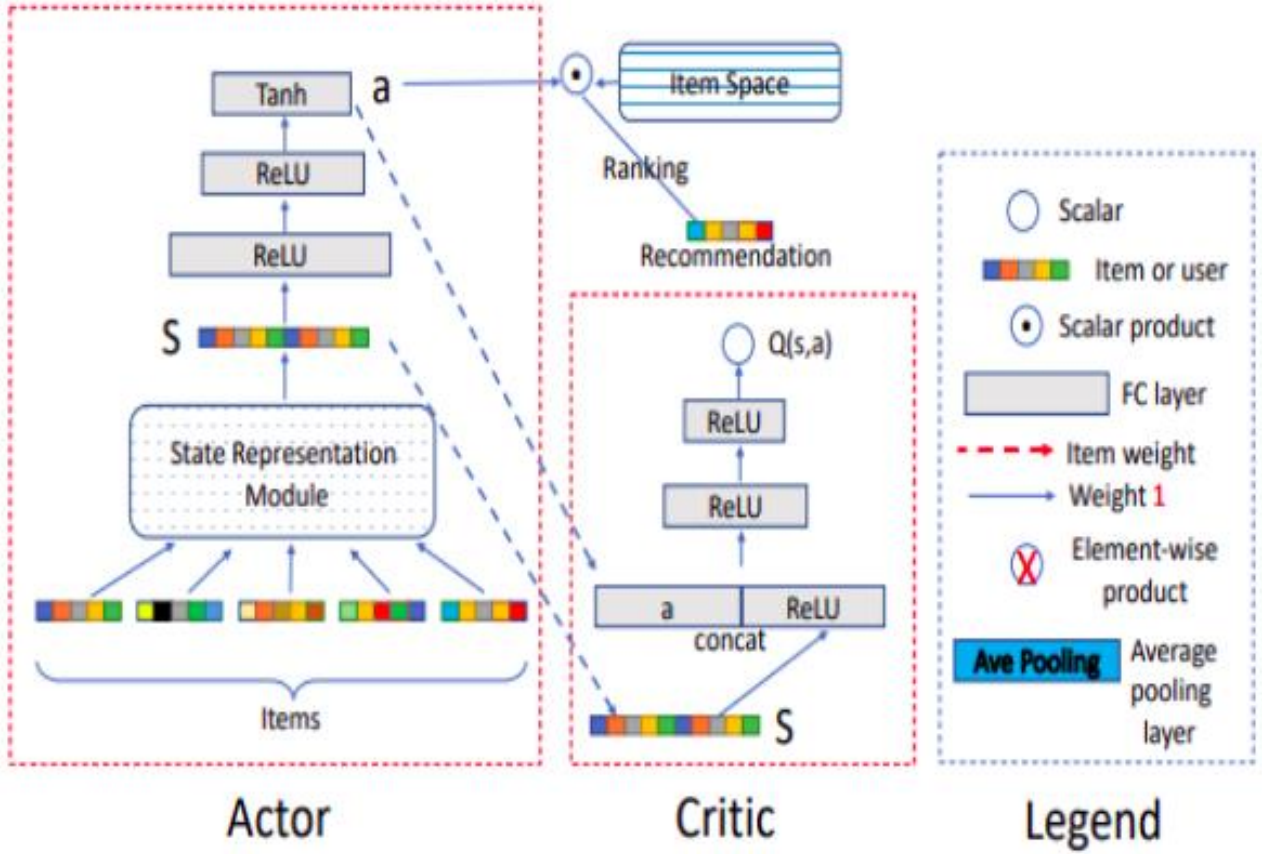
Arquitetura do algoritmo

Critic Network:

Calcula o "mérito" da ação recomendada pela Actor Network em relação ao estado representado e analisa se ação recomendada foi adequada. Essa rede calcula o Q-value resultante desse par ação-ambiente e atualiza-se os pesos da rede Actor após atingir um número de ações minimo (buffer relay)

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

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



Pseudo-código

Algorithm 1: Training Algorithm of DRR Framework

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

Implementação

- Movies dataset (3383 filmes)
- 994k avaliações de usuários sobre estes filmes;
- 4699 inteirações;
- 4:30h de treinamento;

Métricas do agente

Hit: Proporção de recomendações em que um ou mais itens relevantes estão presentes na lista de recomendação fornecida pelo sistema

50%

Discount Cumulative Gain: soma dos valores de relevância classificados de todos os resultados em uma lista de resultados recomendados.

26%

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>.