

Tutorial: RecoGym

Patricio Cerda Mardini


IIC3633
Sistemas Recomendadores
2019-2



Contenidos tutorial

- 1 Material suplementario
- 2 Conceptos
- 3 Recomendación clásica vs. moderna
- 4 RecoGym
- 5 Notebook
- 6 Referencias



- **Notebook de hoy:** link en el aviso siding
- **Ambiente:** Python3
- **Referencia adicional:**  /bamine/recsys-summer-school tiene slides + notebooks adicionales!

Agradecimientos: Amine Benhalloum y Flavian Vasile de Criteo AI Lab por compartir su contenido.



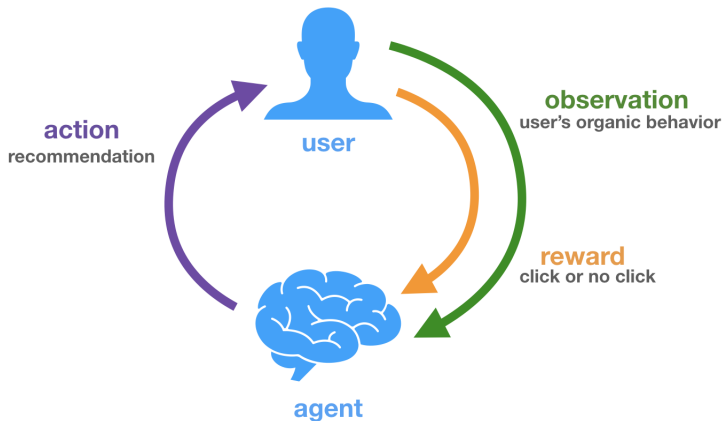
Contenidos tutorial

- 1 Material suplementario
- 2 Conceptos**
- 3 Recomendación clásica vs. moderna
- 4 RecoGym
- 5 Notebook
- 6 Referencias



Conceptos: Reinforcement Learning

“Determinar qué acciones debe escoger un agente de software en un entorno con el fin de maximizar una recompensa o premio acumulado.”



Organic

user views different
items on website

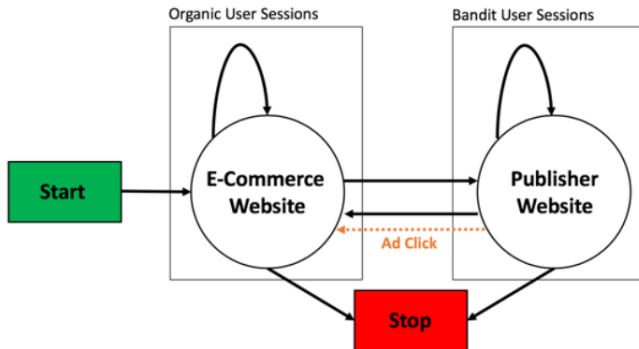


Bandit

agent recommends
items to user



Conceptos: User session



1 Offline Learning

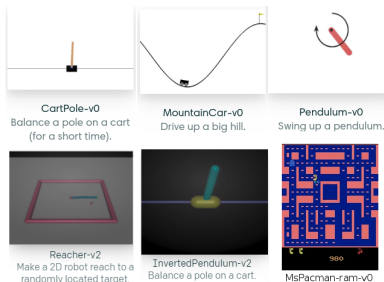
fixed policy chooses **action**,
environment reveals **observation** and **reward**
agent learns from data

2 Online Learning

agent chooses **action**,
environment reveals **observation** and **reward**
agent continues learning



Conceptos: OpenAI Gym



```
import gym
env = gym.make("CartPole-v1")
observation = env.reset()
for _ in range(1000):
    env.render()
    action = env.action_space.sample()
    observation, reward, done, info = env.step(action)

    if done:
        observation = env.reset()
env.close()
```



Extensiones: VizDoom Gym

Link: vizdoom.cs.put.edu.pl/research

Doom

Doom environments based on VizDoom.



DoomBasic-v0 (experimental)



DoomCorridor-v0
(experimental)



DoomDefendCenter-v0
(experimental)



DoomDefendLine-v0
(experimental)



DoomHealthGathering-v0
(experimental)



DoomMyWayHome-v0
(experimental)

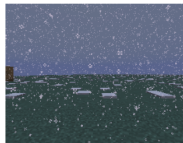


Extensiones: Minecraft Gym

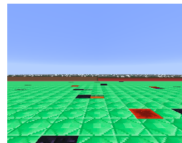
Link: github.com/microsoft/malmo



CliffWalking1-v0



MinecraftDefaultFlat1-v0



MinecraftTrickyArena1-v0



Eating1-v0



MinecraftDefaultWorld1-v0



MinecraftBasic-v0



Contenidos tutorial

- 1 Material suplementario
- 2 Conceptos
- 3 Recomendación clásica vs. moderna**
- 4 RecoGym
- 5 Notebook
- 6 Referencias



Recomendación como un problema de “auto-completar”:

- Uso predominante de información orgánica
- Definición del problema:
 - 1 Missing Link Prediction (e.g. Factorización Matricial)
 - 2 Next-item prediction (e.g. GRU4Rec)



Recomendación clásica

Ventajas:

- Framework establecido
- Datasets y métricas estándar
- Más fácil contribuir y comparar

En el mundo real:

- Datos surgen antes de tener un S.R.
- Métodos eficientes y comprobados
- En general, muy buen punto de partida



Recomendación clásica: un supuesto

Sean V_i items vistos orgánicamente, A_i items recomendados, C_i si el usuario clickea la recomendación. Si:

$$P(V_n = a \mid V_1 = v_1 \dots V_{n-1} = v_{n-1}) > P(V_n = b \mid V_1 = v_1 \dots V_{n-1} = v_{n-1})$$

Se asume:

$$P(C_n = 1 \mid A_n = a, V_1 = v_1 \dots V_{n-1} = v_{n-1})$$

>

$$P(C_n = 1 \mid A_n = b, V_1 = v_1 \dots V_{n-1} = v_{n-1})$$

Pero esto no necesariamente se cumple.



Limitación

Operamos bajo el supuesto de que la mejor política de recomendación es predecir de manera óptima el comportamiento natural del usuario

En general, lo anterior constituye una buena política inicial, sobre todo ante *cold starts*.

Sin embargo, eventualmente este objetivo ingenuo a optimizar podrá diverger del objetivo de negocios (e.g. maximizar Click-Through Rate).

Métricas *offline* (Recall, Precision, Hit Rate @ K) no evalúan la calidad de la recomendación, sino la capacidad de predecir el siguiente ítem.



Recomendación como una política de *intervención* sobre el comportamiento natural del usuario:

- Uso predominante de información tipo *bandit* (e.g. click en un anuncio)
- Definición del problema:
 - 1 De tipo *contextual bandit*
 - 2 De tipo *policy learning*
- Métricas online: evalúan calidad?
 - 1 AB Test: si, pero es caro
 - 2 Estimador “Inverse Propensity Score” de la CTR: sí, pero es ruidoso



Se requiere un registro de recomendaciones, indicando cuándo fueron exitosas (el usuario consumió la recomendación) para poder evaluar la calidad de la salida del sistema.

Casi ningún dataset contemporáneo nos permite medir la calidad de las recomendaciones:

- MovieLens: No
- Netflix Prize: No
- RSC15: No
- 30 Music: No
- Yahoo News Feed Dataset: Sí
- Criteo Counterfactual Dataset: Sí



¿Qué es un registro en este contexto?

u	t	$z_{u,t}$	$v_{u,t}$	$a_{u,t}$	$c_{u,t}$
.
10	0	organic	104	NA	NA
10	1	organic	52	NA	NA
10	2	organic	71	NA	NA
10	3	bandit	NA	42	0
10	4	bandit	NA	52	1
10	5	organic	52	NA	NA
.

Table 2: Example Data



Comparación de políticas

Sean A_t un item recomendado, V_t un item visitado orgánicamente, y π_1, π_2 políticas (que recomendarán dado el pasado del usuario). Supongamos que disponemos de un *log* de π_1 .

Si tenemos:

$$\pi_1(A_2 = \text{piña} \mid V_1 = \text{pizza}) = 0.01$$

$$\pi_2(A_2 = \text{piña} \mid V_1 = \text{pizza}) = 1$$

La política π_2 recomendará piña $\frac{1}{0.01} = 100x$ veces más respecto a π_1 .

Por otro lado, sea c_n el feedback de la n -ésima recomendación: 1 si hubo click, 0 si no.



Inverse Propensity Score

Podemos ponderar cada click según cuánto más probable es que la recomendación aparezca si usamos la nueva política π_2 comparado a la política π_1 , obteniendo un estimador de la *click through rate*:

$$\text{CTR} \approx \frac{1}{N} \sum_n^N c_n \frac{\pi_2}{\pi_1}$$

Como tenemos registro previo de π_1 , se habla de Offline IPS, pero también se puede calcular en línea.

IPS busca responder una pregunta “counter-factual”: qué hubiese pasado?

...usualmente es ruidoso.



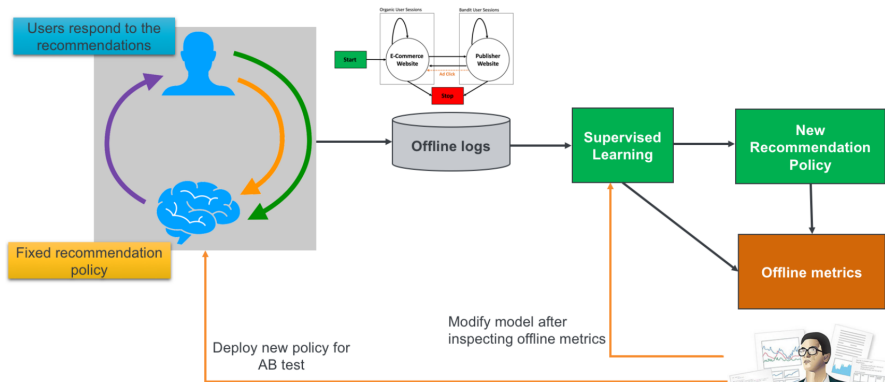
Cómo se mejora un S.R. a gran escala en el mundo real?

- Modelo supervisado aprende de actividad de usuarios *offline*
- Evaluamos con métricas offline
- A/B test
- Si va bien y es escalable, *roll-out*
- Si no, entender la razón y buscar alternativas
- Repetir

¡Esto es aprendizaje reforzado “manual”!



Recommendation: Supervised Learning AB Testing



Experimenter: tunes the model, adjusts according to offline metrics, runs AB tests and changes the production policy



Contenidos tutorial

- 1 Material suplementario
- 2 Conceptos
- 3 Recomendación clásica vs. moderna
- 4 RecoGym**
- 5 Notebook
- 6 Referencias



RecoGym es un *framework* para probar sistemas recomendadores basados en aprendizaje reforzado, tanto *offline* como *online*.

Sigue la API estándar de OpenAI Gym, por lo que proprotipar es fácil y rápido.

RecoGym: A Reinforcement Learning Environment for the Problem of Product Recommendation in Online Advertising

David Rohde
Criteo AI Labs
Paris
d.rohde@criteo.com

Stephen Bonner*
Criteo AI Labs
Paris
st.bonner@criteo.com

Travis Dunlop†
Universitat Pompeu Fabra
Barcelona
dunloptravis@gmail.com

Flavian Vasile
Criteo AI Labs
Paris
f.vasile@criteo.com

Alexandros Karatzoglou
Telefonica Research
Barcelona
alexandros.karatzoglou@gmail.com



- Clase *Agente*: recomendador de productos
- Clase *Environment*: ambiente simulado
 - `reset()`: inicializa usuario sintético aleatorio
 - `step()`: dado el estado actual, retorna:
 - observación: última sesión (completa) del usuario¹
 - *reward*: por la recomendación anterior (click / no click)
 - *done*: terminó la sesión orgánica? De ser así, `reset()`
 - *info*: información adicional del *log* de eventos

¹Si estamos entre eventos bandidos, retorna nulo



RecoGym incluye tres agentes simples como *baselines*:

- 1 **Random:** recomendaciones aleatorias, sin aprendizaje de datos orgánicos
- 2 **Logistic:** al entrenar cuenta la CTR de cada item, y recomienda el máximo. Considera como historia solo al último item orgánico.
- 3 **Prod2Vec supervisado**



Contenidos tutorial

- 1 Material suplementario
- 2 Conceptos
- 3 Recomendación clásica vs. moderna
- 4 RecoGym
- 5 Notebook**
- 6 Referencias



En el notebook asociado, encontrarán un breve ejercicio en RecoGym, donde podrán:

- Instanciar un ambiente
- Definir agentes
- Entrenar y evaluar tanto *offline* como *online*
- Comparar una política simple vs. una *baseline*

Para explorar agentes más sofisticados, ver los notebooks del repositorio oficial!



Contenidos tutorial

- 1 Material suplementario
- 2 Conceptos
- 3 Recomendación clásica vs. moderna
- 4 RecoGym
- 5 Notebook
- 6 Referencias**



- [1] Rohde, D., Bonner, S., Dunlop, T., Vasile, F., Karatzoglou, A. (2018). RecoGym: A Reinforcement Learning Environment for the problem of Product Recommendation in Online Advertising. ArXiv, abs/1808.00720.
- [2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym.
- [3] Amine Benhalloum, Flavien Vasile, David Rohde, Martin Bompaire, Olivier Jeunen. 2019. Modern Recommendation for Advanced Practitioners. RecSys Summer School, Gothenburg, Sweden.

