Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona

Práctica 2 - Mètodes d'exploració: icm i rnd

Aprenentatge per Reforçament Avançat

Fuster Palà, Llum

1 Introducció

En aquest treball ens proposem investigar tècniques d'exploració avançades per a entorns de reforç amb recompensa escassa. El nostre objectiu és implementar dos mètodes d'estat de l'art, Random Network Distillation (RND) i Intrinsic Curiosity Module (ICM), per tal de generar recompenses intrínseques que afavoreixin una exploració més eficient quan la recompensa externa és pràcticament inexistent.

Per convertir l'entorn MountainCar-v0 en un problema de recompensa realment escassa, s'ha utilitzat una nova funció de reforç que retorna 200 únicament quan l'agent assoleix l'estat objectiu i 0 en qualsevol altra transició. En la implementació per defecte, la funció de recompensa original assigna -1 a cada pas fins que l'episodi finalitza. Tot i que aquesta definició sembla en primera instància molt escassa, l'ús d'aproximadors de funció de valor com DQN acaba creant un efecte de "shaping" no intencionat: els estats visitats amb més freqüència acumulen estimacions de valor progressivament més elevades que els no visitats, cosa que guia l'agent cap a certes regions de l'espai d'estats i proporciona, implícitament, un senyal d'exploració. Això fa que la recompensa de –1 no sigui tan poc informativa com podria semblar.

Al llarg del document detallarem primer la implementació de l'ICM i, a continuació, la de RND[3][2]. Finalment, compararem aquests dos mètodes amb l'algorisme Hindsight Experience Replay (HER), adaptat al nou entorn escàs per determinar com influeix cada estratègia en l'eficiència i l'èxit de l'agent[1].

2 Mètodes d'exploració

En aquesta secció presentem les dues tècniques d'exploració d'estat de l'art que hem implementat per afrontar problemes de recompensa escassa: l'Intrinsic Curiosity Module (ICM) i el Random Network Distillation (RND). Cadascun d'aquests mètodes defineix una recompensa intrínseca que incentiva l'agent a explorar estats poc coneguts, superant les limitacions de les estratègies clàssiques basades en ϵ-greedy. A continuació, descrivim la implementació detallada de cada mòdul.

2.1 ICM

La implementació de l'Intrinsic Curiosity Module (ICM) es basa en la classe `ICMNetwork`, que hereta de `IntrinsicRewardModule` i defineix tres components principals: una xarxa de "features" (`feature`) per codificar l'observació en un espai de dimensió reduïda, un model de dinàmica inversa (`inverse_dynamics`) que prediu l'acció realitzada a partir de les codificacions de l'estat actual i el següent, i un model de dinàmica directa (`forward_dynamics`) que, a partir de la codificació de l'estat actual i l'acció, prediu la codificació de l'estat següent.

Per a implementar el mètode `calculate_loss`, primer es construeix la codificació $\phi(s)$ i $\phi(s')$ aplicant la xarxa de "features" a `obs` i `next_obs`. A continuació, el model invers utilitza la concatenació de $\phi(s)$ i $\phi(s')$ per predir la distribució de probabilitats de les accions, la qual es compara amb el vector one-hot de l'acció real mitjançant la pèrdua de cross-entropy. Paral·lelament, el model forward pren $\phi(s)$ (desapegada de la gràfica per no afectar la xarxa de "features") i el vector one-hot de l'acció per predir $\phi(s')$, calculant la pèrdua MSE entre la predicció i la codificació real de l'estat següent i multiplicant-la per 0.5. Finalment, la pèrdua total és una combinació convexa de les dues, ponderada pels hiperparàmetres β i $1-\beta$.

Per calcular la recompensa intrínseca, el mètode `calculate_reward` torna a codificar dins un bloc `with torch.no_grad()` per evitar gradients. A partir de $\phi(s)$ i $\phi(s')$, el model de dinàmica directa prediu $\phi(s')$ i es mesura la discrepància amb la codificació real mitjançant l'error absolut mitjà (MAE). Aquest error es multiplica pel factor d'escala α i s'ajusta amb `unsqueeze(-1)` per garantir la compatibilitat amb la interfície d'entrada del següent mòdul. D'aquesta manera, l'agent rep un senyal d'exploració proporcional a la incertesa en la predicció de la dinàmica del seu entorn.

2.2 RND

La implementació de Random Network Distillation (RND) es basa en la classe `RNDNetwork`, que hereta de `IntrinsicRewardModule` i defineix dues xarxes neuronals amb la mateixa arquitectura: una xarxa *target* fixa i una xarxa *predictor* entrenable. Durant l'entrenament, els pesos de la xarxa *target* es mantenen constants per garantir que la sortida sigui determinista però aparentment aleatòria per cada estat.

El mètode `calculate_loss` computa la pèrdua com l'error quadràtic mitjà (MSE) entre la sortida de la *predictor* i la sortida de la *target* sobre l'observació actual, evitant la retropropagació sobre la xarxa *target* mitjançant un bloc `with torch.no_grad()`.

Per generar la recompensa intrínseca, el mètode `calculate_reward` compara, de manera similar, la predicció del *predictor* i la sortida de la *target* però aplicada a l'observació següent (`next_obs`). Això mesura la "sorpresa" de l'agent en topar-se amb un estat poc vist. L'error MSE es computa sense agregació (`reduction='none'`), es fa la mitjana sobre la dimensió de sortida i es multiplica per un factor d'escala α per controlar la magnitud de la recompensa intrínseca. Finalment, es fa un `torch.clamp(...)` per assegurar que el valor quedi dins l'interval $[0, 1]$ i s'afegeix una dimensió final amb `unsqueeze(-1)` perquè el format sigui compatible amb la resta del pipeline de reforç. Aquesta estructura permet que l'agent rebi un senyal coherent d'exploració, modulant la recompensa segons el desconeixement de cada estat.

3 Resultats

Quan utilitzem la recompensa original $R_a(s, s') = -1$ fins a un màxim de 200 passos o fins a arribar al goal, l'agent aconsegueix arribar a l'objectiu en alguns casos sense cap mètode avançat d'exploració. Això és degut a que la funció de reward, encara que esparsa, té un caràcter informatiu. El fet que cada pas retorni -1 crea un gradient de valor que distingeix clarament estats més propers al goal (amb menor nombre de passos restants fins al límit) d'estats més llunyans. D'aquesta manera, el DQN pot aprendre estimacions de valor ascendent cap a les regions que condueixen al final de l'episodi, guiant l'agent cap a la solució.

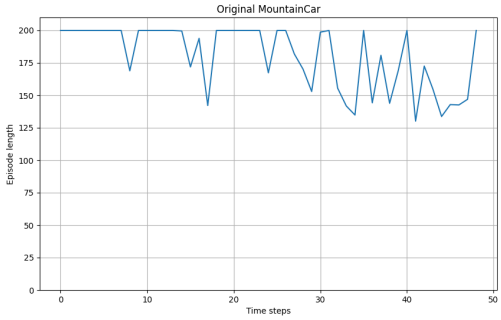
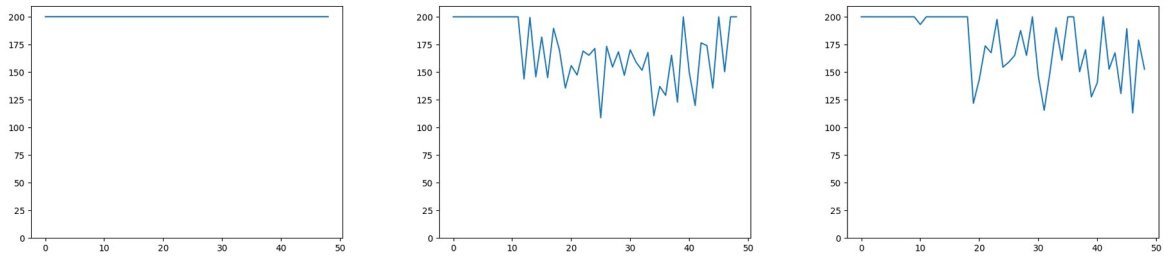


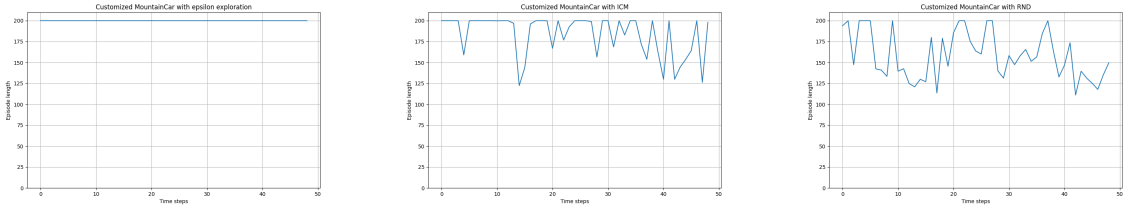
Figure 1: Resultat obtingut amb ϵ-greedy i la recompensa original -1 per pas.

Així doncs, per poder analitzar els resultats i destacar el potencial dels mètodes avançats d'exploració, s'ha utilitzat la recompensa modificada, que obté una línia plana per al mètode epsilon greedy, mostrant que mai aconsegueix obtenir bons resultats, com es mostra en les següents figures. Veurem els resultats esperats i els resultats aconseguits per aquesta nova funció de reforç esparsa i no informativa en l'entorn personalitzat.



(a) Epsilon greedy (b) ICM (c) RND

Figure 2: Resultats esperats per a l'entorn personalitzat.



(a) Epsilon greedy (b) ICM (c) RND

Figure 3: Resultats obtinguts en l'entorn personalitzat

Comparant la Figura 2 dels resultats esperats amb els resultats obtinguts a la Figura 3, podem veure una tendència similar. Per al mètode Epsilon Greedy no aconseguim resoldre el problema en cap iteració i amb els altres dos, encara que no amb una corba massa exacta, aconseguim que pugui aprendre i aconseguir arribar a l'objectiu en molts dels casos.

Cal mencionar que ICM ha aconseguit reproduir el comportament esperat des de la primera execució gràcies a la seva recompensa intrínseca basada en l'error de predicció de la dinàmica, que guia l'agent ràpidament cap a estats nous. En canvi, RND ha demanat fins a la tercera execució per obtenir una corba amb mostra d'aprenentatge comparable a l'esperada. Aquesta variabilitat inicial en RND es pot explicar per la naturalesa aleatòria de la xarxa *target* i per l'estocàstic entrenament de la xarxa *predictor*, que poden generar recompenses intrínseques massa baixes o massa altes en les primeres iteracions, retardant la convergència de l'agents a un patró d'exploració òptim.

Bibliografia

[1] Marcin Andrychowicz et al. "Hindsight Experience Replay". In: *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*. 2017, pp. 5048–5058. URL: https://arxiv.org/pdf/1707.01495.pdf.

[2] Yuri Burda et al. "Exploration by random network distillation". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=H1lJJnR5Ym.

[3] Deepak Pathak et al. *Curiosity-driven Exploration by Self-supervised Prediction*. 2017. arXiv: 1705.05363 [cs.LG]. URL: https://arxiv.org/abs/1705.05363.