REINFORCEMENT LEARNING АШИГЛАН МОГОЙ ТОГЛОХ НЬ

Гантулга Гантөмөр, Луубаатар Бадарч

[1]*Шинжлэх Ухаан Технологийн Их сургууль*
Мэдээлэл, холбооны технологийн сургууль

limited.tulgaa@gmail.com, luubaatar@must.edu.mn

Хураангуй

Энэ ажлын хүрээнд олны танил Могой-Snake тоглоомыг хэд хэдэн аргаар хэрэгжүүлж, тэдгээрийн үр дүнг харьцуулах зорилго тавьсан. Үүнд:

- *Энгийн буюу өөрсдийн зохиосон алгоритм;*
- *Reinforcement-Learning-ийн MDP арга;*
- *Reinforcement-Learning-ийн Q-Learning арга;*
- *Reinforcement-Learning-ийн Deep Q-Network арга;*

Эхний 2 арга нь урьдчилон тодорхойлж өгсөн нөхцлийн дагуу гүйцэтгэдэг аргууд бол сүүлийн 2 арга нь тоглолтын талбай, төлөвийг өөрөө оролдлого хийж, алдаа гаргах замаар үнэлэн автоматаар тогложсурдаг аргууд юм. Туршилтын үр дүнд Q-Learning болон Deep Q-Network аргууд нь өрсөлдөхүйц үр дүн үзүүлж байсан. Сургалтын төлөвийн элемэнтүүдийг нарийн тодорхойлж, суралцах хугацааг уртасгах юм бол агент буюу могойны тоглолтын чадвар улам өсөн нэмэгдэх боломжтой нь харагдаж байна.

Түлхүүр үг: *Q-Learning, Reinforcement Learning, Markov Decision Process, Deep Q-Network.*

Удиртгал

Орчин үед технологи хөгжихийн хэрээр автоматжуулалт, робот систем, хиймэл оюуны салбаруудад томоохон дэвшлүүд гарч байна. Хүчирхэг тооцоолох чадамж өндөр компьютеруудын ачаар сүүлийн жилүүдэд өндөр үр дүн үзүүлж байгаа салбар бол Reinforcement Learning [1] юм. Амжилттай хэрэгжсэн жишээнүүд нь DeepMind–AlpaGo, StartCraftII [2], OpenAI–Dota2 OpenAI Five [3] зэрэг юм. Энэхүү ажлын хүрээнд бид RL-ыг програмчлалд хэрхэн энгийн байдлаар буулган ашиглаж байгааг харуулахыг зорьсон бөгөөд могой тоглоомыг тоглуулж болох хэд хэдэн аргуудыг туршиж өөр өөр аргуудын давуу болон сул талыг харуулахыг зорьсон болно. Тоглоомын код болон бүх алгоритмууд Python хэл дээр бичигдсэн бөгөөд Deep Q-Network [4] сургахдаа Tensorflow [5] (Машин сургалтад зориулагдсан сан) Python хэлний open-source сан ашигласан.

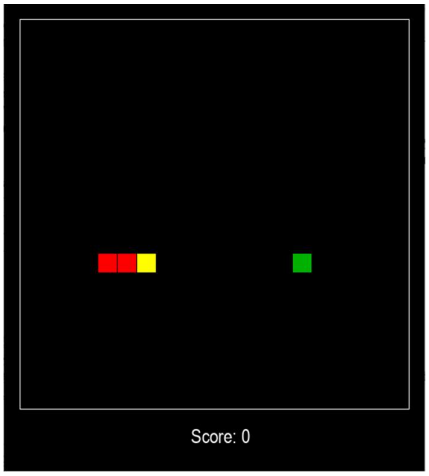
1. RL-ын үндсэн ойлголтууд

RL-ийн агент нь өөрийн хүрээлэн буй орчинд тодорхой нэг үйлдэл(action-a) хийх ба үр дүнд агентын нөхцөл байдлаас хамааран тодорхой хэмжээний шагнал (reward-r) авна. r ямар байхаас хамааран тухайн нөхцөл байдалд (state-s) ямар үйлдэл хийгээд ямар нөхцөл байдалд(state'-s') орсноос хамааран дүгнэх замаар агентыг сургадаг. Ийм бүтцээр задалж үзсэнийг MDP (Markov Decision Problem) [6] гэж авч үздэг. RL нь одоогийн нөхцөл байдлаас дараа дараачийн нөхцөлүүдийг зөв үнэлж хугацааны төгсгөлд reward хамгийн их байхаар тухайн орчинд үйлдэл хийх чадамжтай агентыг сургах зорилготой ба үүнийг үнэлэх тооцоолох арга нь MDP юм. RL-ыг хэрэгжүүлдэг олон аргууд байдаг ба

ерөнхийдөө On-Policy [7] ба Off-Policy [8] гэж хоёр ангилдаг. Бидний хэрэгжүүлсэн арга нь “Off-Policy” төрлийн арга бөгөөд орчны тухай мэдээлэлгүйгээр суралцах боломжийг олгодог арга юм.

2. Могой

Могой тоглоомын дүрэм нь энгийн бөгөөд биеэ мөргөх юм уу тоглоомын талбараас хэтэрсэн тохиолдолд тоглоом дууссан гэж үзэх бөгөөд RL-ын хувьд энэ нь "Terminal state" буюу эцсийн цэг болох юм. Могой нь зөвхөн харж буй чиглэлдээ чигээрээ явах, баруун эргэх болон зүүн эргэх гэх гурван үйлдэл хийх ба агент нь эдгээр үйлдлүүдээс сонгон явна.



Зураг 1. Могой тоглоомын талбар

Могой тоглоомоос state болгон ашиглаж болох мэдээллүүд нь:

- Тоглоомын талбар
- Хоолны байршил
- Могойны биеийн байршил
- Талбарт харагдаж байгаа зүйлстэй хамааралтай бүх мэдээлэл

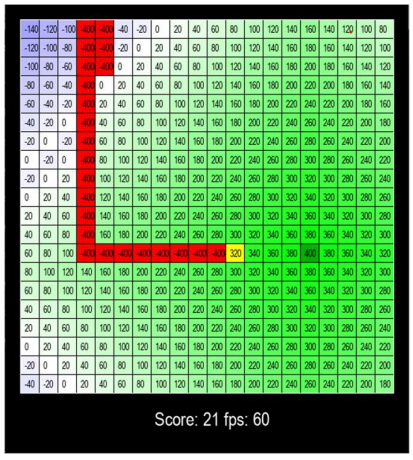
Бүх state-ыг $s_t \, \epsilon \, S = \{s_0, \ s_1, \ s_{2,} \ldots\}$
Хийж болох үйлдлүүд $a \, \epsilon \, A = \{0, 1, 2\}$
$0 =$ чигээрээ; $1 =$ баруун; $2 =$ зүүн;

Тоглоомын явцад ямар нэг алгоритм ашиглан хийх үйлдлээ сонгох ба алгоритмыг тоглоомын state-ээс хамаарсан функц гэж үзээд $\pi_{(s)}$ гэж үзье.

Snake game Basic game flow
Initialize Snake game
while game running
    Get game state s from S
    while game is not over
        Get action a using $\pi_{(s)}$
        Execute action a in game
        Get agent’s observation s'
        Assign s' to s
    end while
end while

3. MDP (Value Iteration)

Markov Decision Problem (MDP) нь тухайн орчны state хоорондын хамаарлыг орчны динамикаас хамааруулан тооцоолох арга юм. Value Iteration [9] нь MDP-г тооцоолох арга ба Offline Learning алгоритм гэж үздэг. Энэ нь тоглоомын мэдэгдэж буй утгууд дээр тооцоолол хийн дүгнэдэг буюу тоглоомыг тоглох шаардлагагүйгээр суралцах бус тооцоолох замаар үнэлдэг арга юм. Могой тоглоомын хувьд тоглоомын талбарын нүд бүрийг state $s = (i, j) \quad (i - \text{мөр}, j - \text{багана})$ гэж үзээд Value Iteration ашиглан state бүрийн утгыг хоолны байршлаас хамааран тооцоолох юм.



Зураг 2. MDP planning with Value Iteration

Value Iteration ашиглан MDP тооцоолоход

$$V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a) * (R(s,a,s') + \gamma V_k(s')) \quad (1)$$

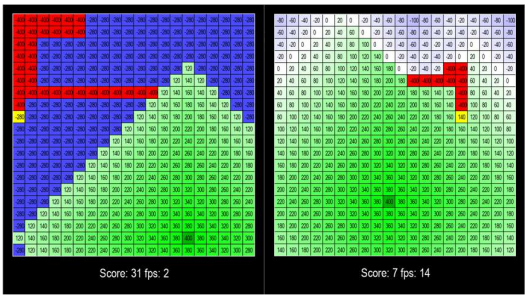
(1) томьёог ашиглах ба γ, $P(s'|s,a)$, $R(s,a,s')$ зэрэг утгууд мэдэгдэж байх ёстой. Могойны хувьд орчны динамик бүх state-д ижил байх ба агент зорьсон state дээ заавал буудаг буюу тухайн сонгогдсон үйлдэл бүрийн магадлал тогтмол 1 гэж үзэж болно. R буюу reward функцийн утгуудыг

$$R = \begin{cases} if\ s' == food \rightarrow r = 400 \\ if\ s' == over \rightarrow r = -4000 \\ else\ r = -20 \end{cases} \quad (2)$$

дүрмээр тогтоож өгсөн. Тооцооллыг хялбар байлгах үүднээс γ параметрийг 1 гэж үзсэн болно. (1) томьёог дээрх нөхцөлүүдийг харгалзан хялбарчилбал

$$V_{k+1}(s) = \max_{a \epsilon A}(R(s,a) + V_k(s')) \quad (3)$$

(3) томьёогоор тоглоомыг дүрслэх бүрд state болгон дээр бодогдох ба нэг frame дүрслэхэд нэг ба түүнээс дээш удаа тооцоолол хийж болно.



Зураг 3. 1 frame-д 1 удаа (зүүн гар тал), 20 удаа тооцоолох (баруун гар тал)

Програмд хийх нийт давталтын тоо $4km^2$ (m^2 талбарын хэмжээ) болно. MDP ашинлан тооцоолоход програмын нийт тооцоолох хугацаа $O(n)$ байх ба тоглоомын талбараас хамааран $O(n^2)$ байна. Агент нь алхам бүртээ могойн толгойг тойрсон дөрвөн нүдний хамгийн их утгатай зүгт хөдлөх ба зөвшөөрөгдсөн нүдний утгууд ижил байвал A-ын боломжит хамгийн бага индекс дээрх үйлдлийг сонгож явах ба сул үйлдэл хийсэнд тооцно. Тоглоомын талбарыг бүрэн бодож дуусах хугацаа бага байх тусам могой нь сул үйлдэл хийх нь багасна. Зураг 3-т харуулж буйгаар агент маань үйл тооцооллоо хийх ба шууд хоолруу чиглэсэн үйлдэл хийх нь урт хугацаанд үр дүнгүй болох тоглуулах явцад харагдсан. Өөрөөр хэлбэл хэт greedy үйлдэл хийх нь могой тоглоомыг тоглох арга биш гэдгийг харуулж байна.

Snake game with Value Iteration

Initialize Snake game
Initialize state value table $V_{(s)}$ by zeros
Set value table update rate
Now our $\pi_{(s)}$ becomes $V_{(s)}$
while game running
    Reinitialize state value table $V_{(s)}$ by zeros
    Get game state $s \epsilon S$
    while game is not over
        Choose action a using $V_{(s)}$
        Execute action a in game
        Get agent's observation s'
        Update $V_{(s)}$ by update rate
        Assign s' to s
    end while
end while

4. Q-Learning

Greedy үйлдэл хийх нь хамгийн сайн арга биш гэдгийг бид MDP ашиглаж харууллаа. Иймээс нэг шат ахин RL ашиглан Online Learning буюу тоглоомыг тоглуулах замаар агентыг сургасан. Q-Learning нь орчны state үүдээс sample хийх замаар буюу тухайн орчныхоо state үүдээс хамааран үйлдэл хийх замаар state үүдийн үйлдэл бүрт харгалзах утгыг тооцоолдог арга юм. Value Iteration хэрэгжүүлж байхад тоглоомын талбарын нүд бүрийг state гэж үзсэн ба reward-ын тархалтаас агентын хийх үйлдлийг тогтоож байсан бол Q-Learning аргаар сургахдаа агентын state нь могойны толгойтой харьцангуй хоолны байрлах чиг, могойн толгойг тойрсон 4 нүдний комбинацууд байхаар тодорхойлсон.

Q-Learning нь state бүрд харгалзах үйлдлүүдийг үнэлсэн хүснэгт(Q-Table) байх ба тэдгээр үнэлгээнүүдийг агентын авж буй reward-аас хамааруулан өөрчилж тааруулан хугацааны төгсгөлд хамгийн их reward авах арга барилд суралцахад оршдог.

Q-Learning нь тоглоомыг бүрэн үнэлэхийн тулд бүх state үүдийг нэг буюу түүнээс олон удаа тооцоондоо оруулсан байх шаардлагатай байдаг. Хэрвээ агент ямар нэг байдлаар тодорхой хэмжээний эерэг reward өгдөг арга олбол тоглоом нэг л аргаар хэрэгжиж сурсан байх магадлалтай. Өөрөөр хэлбэл агентыг илүү сайн арга барилд сургахын тулд өөр өөр аргуудыг туршиж үздэг байх буюу рандомчлал хэрэгтэй болно. Үүнийг RL-д Exploration and Exploitation [10] буюу рандомчлал болон сурсан чадварыг хэрхэн ашиглахыг тодорхойлно.

$$Q_{(s,a)} = Q_{(s,a)} + \alpha(R_{(s,a)} + \gamma \max_{a'} Q_{(s',a')} - Q_{(s,a)}) \quad (4)$$

томъёог ашиглах ба параметрүүдийг:

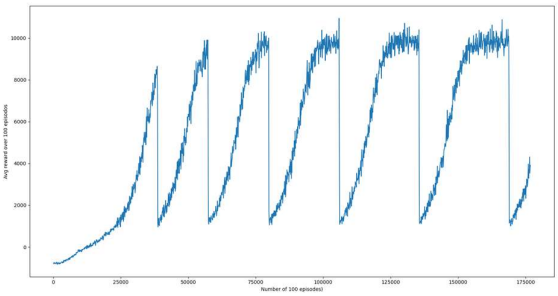
- $\alpha = 0.01$
- $\gamma = 0.999$
- $\varepsilon = 1$
- $\varepsilon_{decay\_rate} = 0.999999$
- $\varepsilon_{min} = 0.01$

$$R = \begin{cases} if\ s' == food \rightarrow r = 400 \\ if\ s' == over \rightarrow r = -600 \\ else\ r = 0 \end{cases} \quad (5)$$

Snake game with Q-Learning

Initialize Snake game
Initialize Q table $Q_{(s,a)}$ by zeros
Now our $\pi_{(s)}$ becomes $Q_{(s,a)}$
while game running
  Get game state $s\ \epsilon\ S$
  while game is not over
    Choose action a using
      $epsilon\ greedy\ with\ argmaxQ_{(s)}$
    Execute action a in game
    Get agent's observation s', r
    Update $Q_{(s,a)}$ by equation 4
    Assign s' to s
  end while
end while

Сургалтыг хэд хэдэн янзаар хэрэгжүүлж үзсэн ба хамгийн сайн сурсан агент нь



Зураг 4. Нийт сургалтын 100 үе бүрийн дундаж reward

Зураг 4. суралцсан ба MDP-ээс ялгаатай байдлаар суралцаж greedy бус үйлдлүүдийг хийж тодорхой арга барилын дагуу тоглож байгаа нь харагдсан. Гэвч могой тоглоомыг төгс тоглодог байхын тулд дөрөвхөн нүдний мэдээлэл хангалтгүй болох нь харагдсан.

5. Deep Q-Network

Энэ арга нь Q-Learning-ын $Q_{(s,a)}$ функцийг Neural Network болгон сольсон арга бөгөөд давуу тал нь Q функцийн оролт тодорхой хязгаарт хязгаарлагдахгүй, агентын харж байгаагүй state д ойролцоо state үүдээс хамааруулан ойролцоолчилох боломжтой. Мөн Network архитектураас хамааран агент өөр өөр байдлаас суралцах боломжтой. DQN-ыг хэрэгжүүлэхэд гарах нэг ялгаа нь q-table-ыг тэгээр дүүргэж байсан бол Network дээр бүх weight үүдийг тэгээр өгч болдоггүй учир санамсаргүй тоонуудыг олгодог. Network санамсаргүй учир Network-ын алдааг бодох нь асуудалтай болно. Network-ын гаралт мөн тодорхой бус байна. Тиймээс Network-ын зорилтот утгыг боддог байх тогтвортой Network байх хэрэгтэй. Үүний тулд үндсэн Network-ыг хэсэг хугацаанд өөрчлөлтгүй байлгах шаардлагатай ба үндсэн Network-ын weight үүдийг ижил архитектуртай Target Network-д хуулж үлдээдэг [4]. Ингэснээр үндсэн Network-ын алдаа тогтвортой болно. Мөн Network-ыг SGD байдлаар сургах нь үр дүн муутай байдаг ба Mini Batch Gradient Descent аргаар сургах нь илүү үр дүнтэй байдаг. MBGD аргаар

сургахын тулд бэлэн өгөгдлүүд шаардлагатай бөгөөд Replay Buffer [4] ашиглаж тоглоомын тухайн агшиндах s, a, s', r ын утгуудыг хадгалж авна.
DQN аргаар сургахад:

$$y_i = R_{(s,a)} + \gamma \max_{a'} Q_{i-C}(s', \theta_{i-C}) \quad (6)$$

$$L_i(\theta_i) = (y_i - Q_i(s, a, \theta_i))^2 \quad (7)$$

$$\theta_i \coloneqq \theta_i + \alpha \nabla_{\theta_i} L_i(\theta_i) \quad (8)$$

(6), (7), (8) томьёоны дагуу хэрэгжих бөгөөд параметрүүдийг:

- $\alpha = 0.001$
- $\gamma = 0.99$
- $\varepsilon = 1$
- $\varepsilon_{decay\_rate} = 0.99999$
- $\varepsilon_{min} = 0.01$
- $batch\ size = 256$
- $C = 5$
- $Buffer\ size = 20000$

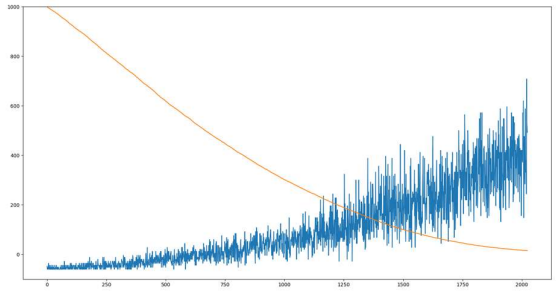
$$R = \begin{cases} if\ s' == food \rightarrow r = 20 \\ if\ s' == over \rightarrow r = -20 \\ else\ r = 0 \end{cases} \quad (5)$$

Network архитектур:
Оролтын давхарга: 126
Дундын давхарга: 96, 48
Гаралтын давхарга: 3

Snake game with DQN

Initialize Snake game
Initialize main and target Q network weights
Now our $\pi_{(s)}$ becomes $Q_{(s,a,\theta)}$
while game running
  Get game state $s \epsilon S$
  while game is not over
    Choose action a using
      $epsilon\ greedy\ with\ argmaxQ_{(s,\theta)}$
    Execute action a in game
    Get agent's observation s', r
    Append s, a, s', r into Replay Buffer
    When have enough data to train
      Sample data from Replay buffer
      update main net θ by equation 8
    After C updates update target net
    Assign s' to s
  end while
end while

Deep Learning модель сургахад параметр болон network-ын хэмжээг сайн тааруулж өгөх нь чухал байдаг. Могой тоглоомыг сургах явцад агентын нийт цуглуулах reward тодорхой утгад хүрээд буурах хандлагатай байсан. Үүнийг бид network-ын сонголтоос хамаарч байгаа гэж дүгнэсэн.



Зураг 5. Нийт сургалтын 100 үе бүрийн дундаж reward

Энэхүү алгоритм нь MDP-тэй төстэй байдлаар ажилладаг бөгөөд гол ялгаа нь нүд бүр дээр тооцоолол хийх бус могойн толгойг тойрсон 3 нүдийг шалгадгаараа ялгаатай. Хоолны могойн толгойтой харьцангуй байршлаар хийж болох үйлдлүүдээс нөхцөл шалган шийднэ.

Snake game with Hard coded agent

Initialize Snake game
Initialize agent $\pi_{(s)}$
while game running
  Get game state s from S
  while game is not over
    Get action a using $\pi_{(s)}$
    Execute action a in game
    Get agent's observation s'
    Assign s' to s
  end while
end while

6. Гараар бичсэн алгоритм

Дүгнэлт

Могой тоглоомыг тоглох оролдлогуудаас Q-Learning алгоритм нь хамгийн өндөр үзүүлэлттэй байсан ба могой тоглоомын хувьд төсөөтэй бөгөөд ялгаатай үр дүнд хүрэх state үүд олон тул нэг функц ашиглан могой ний үйлдлийг таамаглах нь жижиг network-ын

хувьд тун хэцүү байж болох юм. Энэ нь DQN арга бусад аргаас тааруу үр дүн үзүүлсэн шалтгаан гэж үзсэн. Мөн DQN сургалтын хугацаа удаан явагдаж байсан тул бид нийт 2000 гаруй үе тоглуулсан бол Q-Learning дээр 300000 гаруй үе тоглуулсан. MDP ашиглан могойн хийх үйлдлийг төлөвлөх нь үр дүнтэй арга ч могой тоглоомыг тоглох ерөнхий модель гэж үзэхэд тооцоолол хийх хугацаа үзүүлж байгаа үйлдлүүдээс харахад тохиромжгүй байгаа юм. MDP, Q-Learning аргууд нь гараар бичсэн алгоритмыг давсан үзүүлэлттэй байсан.

Ном зүй

[1] Sutton, Richard S., and Andrew G. Barto. "Reinforcement learning." Journal of Cognitive Neuroscience 11.1 (1999): 126-134.

[2] Vinyals, Oriol, et al. "Starcraft ii: A new challenge for reinforcement learning." arXiv preprint arXiv:1708.04782 (2017).

[3] Berner, Christopher, et al. "Dota 2 with large scale deep reinforcement learning." arXiv preprint arXiv:1912.06680 (2019).

[4] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

[5] Goldsborough, Peter. "A tour of tensorflow." arXiv preprint arXiv:1610.01178 (2016).

[6] Even-Dar, Eyal, Yishay Mansour, and Peter Bartlett. "Learning Rates for Q-learning." Journal of machine learning Research 5.1 (2003).

[7] Singh, Satinder, et al. "Convergence results for single-step on-policy reinforcement-learning algorithms." Machine learning 38.3 (2000): 287-308.

[8] Anahta May, Peter J. "Policy learning and failure." Journal of public policy (1992): 331-354.rcı, Berkay, Can Deha Karıksız, and Naci Saldi. "Value iteration algorithm for mean-field games." Systems & Control Letters 143 (2020): 104744.

[9] Coggan, Melanie. "Exploration and exploitation in reinforcement learning." Research supervised by Prof. Doina Precup, CRA-W DMP Project at McGill University (2004).