

Tanítóminta-priorizálási eljárások vizsgálata folytonos kimenetű ágensek esetére

Szerző:

Knáb István Gellért



Közlekedésmérnöki és Járműmérnöki Kar

Konzulens:

Kővári Bálint

OTDK dolgozat

Tartalomjegyzék

1. Bevezető	1
1.1. Probléma feltárása, történelmi áttekintés	1
1.2. Kapcsolódó kutatások	4
1.3. Megoldási kísérlet	6
2. Algoritmusok	8
2.1. Gépi Tanulás	8
2.2. Megerősítéses Tanulás	9
2.3. Actor-Critic algoritmusok	12
2.4. Soft Actor Critic	14
2.5. Alkalmazott neurális hálózatok	16
2.6. UCB alapú priorizálás	16
3. Környezetek és a módszerek összehasonlítása	18
3.1. Pendulum-v1	19
3.2. InvertedPendulum-v5	20
3.3. DoubleInvertedPendulum-v5	21
3.4. HalfCheetah-v5	22
4. Eredmények	24
5. Konklúzió	28

Kivonat

A Mesterséges Intelligencia már mindenki életének a szerves részét képezi. Ahogyan napjainkra a hardverek teljesítménye utólérte a Gépi Tanulós algoritmusok valós idejű futásához elengedhetetlen követelményeket, számos területen terjed a Deep Learning alapú modellek és neurális hálózatok használata. A valós idejű döntéshozás mellett viszont egyes rendszerek kialakítása esetén megfontolandó tényező a tanítás, hiszen jelentős anyagi vonzatokkal járhat ez az offline folyamat. Számos módszer létezik a gyorsabb tanítás megvalósítására, melyek közül kedvelt eljárások Megerősítéses Tanulás terén a hiperparaméter optimalizálás, a reward függvények tanítás előtti kiértékelése valamint a memóriából tanuló algoritmusok esetén a priorizált tanítóminta visszajátzás. A 2016-ban megjelent Prioritized Experience Replay a Google DeepMind által fejlesztett módszer, ami a mintákat a Temporal Difference Error szerint címkézi fel, ezzel nagyobb valószínűséggel mintavételezve azokat a tapasztalatokat, amik elmaradnak az ágens jelenlegi ismerete szerinti optimálistól, így többet tud belőle tanulni. Diszkrét kimenetű algoritmusok terén fellelhetőek olyan kutatások, melyekben más metrikák gyorsabb konvergenciát eredményeznek az elsőként említett alkalmazásánál. A mérnöki problémák viszont sokszor folytonos kimenetet igényelnek. Ennek fényében a dolgozat a Soft Actor Critic nevű folytonos döntési terű algoritmuson mutat be egy sajátos priorizálási technikát, ami szélesebb skálán való alkalmazhatóságot biztosít a műszaki tudományok terén, konvergencia görbéit több környezetben végzett tanítást követően kiértékelve szemlélítve a téma kutatásának jelentőségét.

Köszönetnyilvánítás

A 2024-2.1.1-EKÖP-2024-00003 számú projekt a Kulturális és Innovációs Minisztérium Nemzeti Kutatási Fejlesztési és Innovációs Alapból nyújtott támogatásával, a EKÖP-24-2 pályázati program finanszírozásában valósult meg.



A Reinforcement Learning Continuous Agent Performance Comparison projekt nevében köszönetet mondok a HUN-REN Cloud (lásd: Héder et al. 2022; <https://science-cloud.hu/>) használatáért, ami hozzájárult a publikált eredmények eléréséhez.



1. fejezet

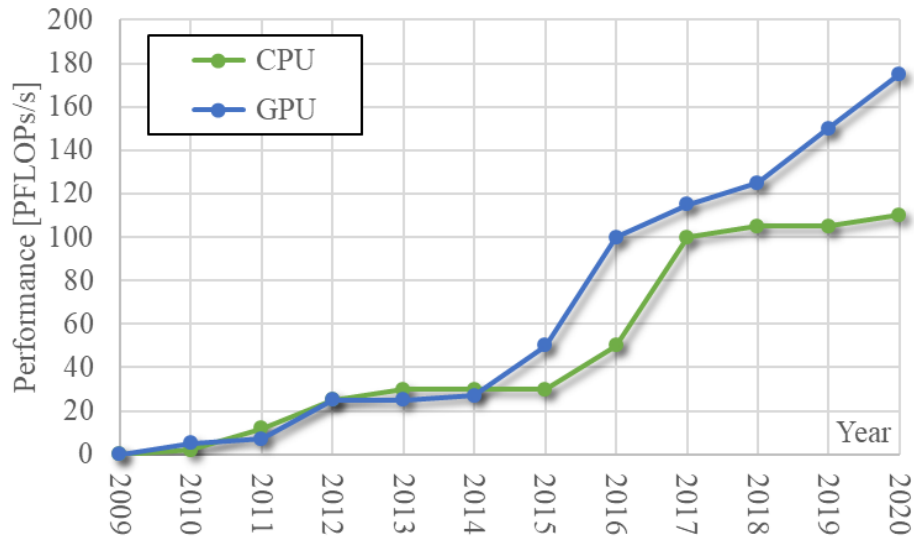
Bevezető

1.1. Probléma feltárása, történelmi áttekintés

A mesterséges intelligencia (Artificial Intelligence - AI) lassacskán mindenki hétköznapijainak elengedhetetlen elemévé válik. Találkozhatunk vele a keresések alapján hangolt böngészőkben, számítógépes játékokban mint ellenfél, vagy a nagy nyelvi modellek formájában, amik teljesen átalakították a korszerű böngészést és gyors információhoz való hozzájutást. Emellett orvosi alkalmazásai[1] és műszaki téren[2] való bevetetősége is egyre szélesebb skálán mutatkozik meg. Bár az ipari fejlődés ezen ága a 2010 környékén indult AI boom alatt tudatosult a legtöbb emberben, története és a kutatása nagyjából 80 éves múltra tekint vissza, ugyanis az első munka a témában 1943-ban került bemutatásra Warren McCulloch és Walter Pitts által[3]. Itt már megjelenik az első gépi neuron, ami később a Mély tanulás(Deep Learning - DL) elemi alkotójaként kerül képbe.

Az, hogy az AI és DL csak napjainkra került a köztudatba, több oldalról is indokolható. A főbb kutatási hullámokat AI telek határolják el egymástól, ami mind a túlzott kezdeti elvárásoknak, mind pedig a szükséges számítási kapacitások hiányának betudható. Az első AI tél az 1970-es évekre tehető, ahol az említett elvárásoktól való elmaradás valamint a rendelkezésre álló hardverek korlátai egyaránt visszaszorították a kutatók motivációját a témában való tevékenykedésre. A 80-as évek közepére a neurális hálózatok(Neural Network - NN) új irányt vettek, a backpropagation algoritmus[4] széles körben való alkalmazásával lehetővé vált a hibák rétegek közti visszaterjesztése, ezzel addig nem látott hatékonyságú tanítást eredményezve. Az algoritmus a korszerű DL rendszereknek is fontos eleme.

A 90-es évekre ismét szembesülni kellett azzal, hogy bár algoritmusok terén a korszerű AI sok alkotója rendelkezésre állt, a hardverek még mindig nem képesek a neurális hálózatok effektív tanításához szükséges teljesítményre, és bár értek el sikereket, mint



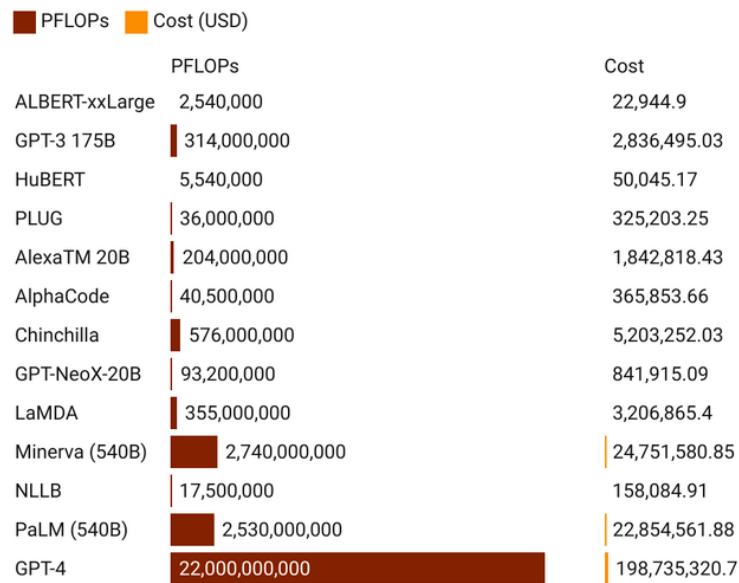
1.1. ábra. A korszerű CPU és GPU által elvégezhető maximális lebegőpontos műveletek száma[5].

többek között a DeepBlue nevű AI, ami legyőzte az aktuális sakk világbajnokot 1997-ben, újabb 10-15 évet kellett várni a valódi áttörésre.

2010 környékére azonban eljött az idő, amikor az AI addig nem látott mértékben indult fejlődésnek. Ez köszönhető annak, hogy a hardverek folyamatos robbanásszerű teljesítménynövekedés során elérték a nagyobb számításigényű DL algoritmusok tanításához szükséges szintet, és ez a folyamat napjainkban sem lassul, ahogy a 1.1 ábra is mutatja. Főleg a grafikus processzorok(Graphical Processing Unit - GPU) tekintetében számottevő a fejlődés, hiszen bár kezdetben grafikus megjelenítésre tervezték őket, nagy sebességgel tudnak mátrix és tenzor műveleteket végrehajtani, amik a DL magját képző számítások.

Sorra jelentek meg a tudomány legkülönbözőbb területein alkalmazások, 2012-ben számítógépes látórendszerek terén az AlexNet[6], ami jelentősen túlteljesítette klasszikációban az addig legjobb teljesítményűnek számító gépi látás módszereket. Ezt követte az AlphaGo és AlphaGo Zero[7] győzelme az aktuális Go világbajnokkal szemben. Ezek mellett az eredmények mellett kétségtelenül a legnagyobb hatása a nagy nyelvi modellek(Large Language Modell - LLM) 2020-as berobbanásának van. A legújabbak már nem csak a keresőbe beadott üzenetekre adnak választ, de képesek dokumentumokat és képeket is generálni. Komplexitásuk növekedése a 1.2 ábrán jól követhető, mind a tanításhoz szükséges lebegőpontos műveletek tekintetében, mind pedig a kapcsolódó költségeket illetően. A folyamatot tekintve, láthatóvá válik a DL tanítások egy újabb megoldandó feladata. A kapcsolódó költségek csökkentése valamint a tanítási idők visszaszorítása egyaránt lényeges metrikává vált az ipari résztvevők számára.

Training Cost in US Dollars using PFlops



1.2. ábra. A szükséges lebegőpontos műveletek és kapcsolódó költségek nagy nyelvi modellek tanítása esetén[8].

A hatékonyabb tanítás a Gépi Tanulás (Machine Learning - ML) alapú rendszerekben több módon is kivitelezhető. DL tekintetében általános metodológia a hiperparaméterek optimalizálása, ami többek között az Optuna Python könyvtárral elvégezhető feladat. Emellett a dolgozat témáját is képző Megerősítéses Tanulás (Reinforcement Learning - RL) körében fellelhetőek olyan kutatások, ahol a különböző jutalmazási struktúrák teljesítménye tanítás előtt összehasonlításra kerül, és ezek közül csupán a legalkalmasabb vesz részt a tanítási folyamatban, ezzel is csökkentve a futások számát, csupán a hiperparaméter keresésre szűkítve az elvégzendő tanításokat. Ennek az előzetes kiértékelésnek az egyik alkalmazott módja a Monte Carlo Tree Search[9][10], amiben az egyes értékek a reward függvényében kerülnek meghatározásra. A harmadik módszer, ami a dolgozat témáját is képi, a tanítómintákat érintve hoz változást. Az NN alapú rendszerek tekintetében megfigyelhető, hogy nem minden mintának van azonos információtartalma az ágens számára, egyes adatokból többet tud tanulni a hálózat. Ebből adódik, ha a mintavételezés nem sztochasztikusan valósul meg, hanem olyan valószínűségek mentén, hogy a nagyobb információtartalmú adatok nagyobb valószínűséggel kerülnek mintavételezésre, gyorsítható a konvergencia.

1.2. Kapcsolódó kutatások

Az ML egy szignifikáns hatással bíró mérföldköve volt a Deep Q-network(DQN)[11] algoritmus 2015-ös megjelenése, amelyben számos játékban az emberi teljesítményt felülmúló ágensek kerültek bemutatásra. Az algoritmus egyik hatalmas előnye, hogy memóriából tanul, ami mind az adatok későbbi felhasználását lehetővé teszi mintavételezés során, mind pedig az adott trajektóriában lévő adatok korrelációját is megtöri, ezzel hatékonyabb tanulást eredményezve.

Kezdetben a memóriából sztochasztikus mintavételezés során került kiválasztásra egy batch, amin a két NN értékei alapján kiszámításra került a veszteség és elvégezhetővé vált a backpropagation. Bár megfelelő hiperparaméterek esetén ez a módszer sikeresnek bizonyul, mégis tekintve a 1.2 ábrát, megfogalmazódik az igény a tanítási idő rövidítésére.

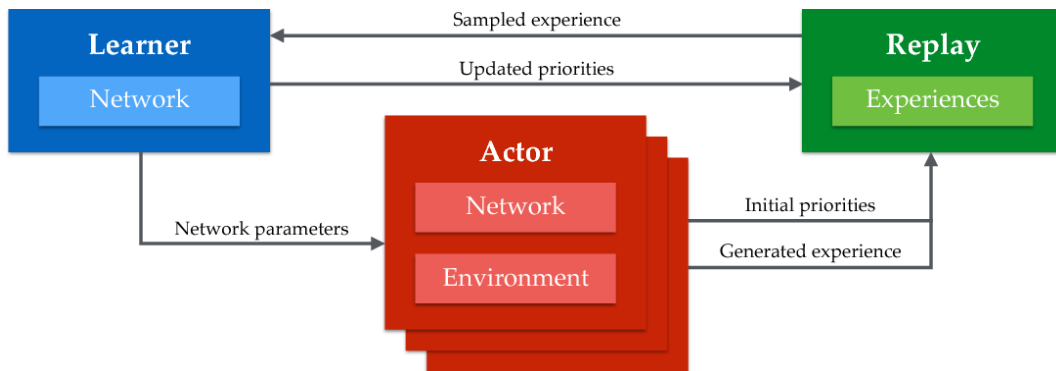
A DeepMind 2015 végén mutatta be publikációját, melyben ismertette az 1.3 ábrán látható Prioritized Experience Replay(PER)[12] koncepcióját. A 1.1 részben leírtak szerint, az adatok információtartalma egymástól eltérő, így amennyiben egy adott stratégia mentén valósul meg a mintavételezés, gyorsabb konvergencia állítható elő. Az adott tanítóminták információtartalma a Temporal Difference(TD) Errorban rejlik, mivel kapcsolatot teremt az ágens adott döntésének minősége és az aktuális ismeretek szerinti legjobb érték között. A TD Error annak a mérőszáma, mennyire meglepő egy adott állapotátmenet, mennyire van messze a következő becsült elérhető értéktől. Számítása az alábbi módon tehető meg[13]:

$$\delta_t \doteq R_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t), \quad (1.1)$$

ahol a V_t az egyes állapotok értéke, az R_{t+1} pedig a kapott jutalom. Bár sok esetben jól működik, a cikkben két másik módszer is ismertetésre került, mivel az első mintavételezés során kis TD Errorra eredményező esetek csak hosszú idő után kerülnek újra mintavételezésre, és ez lecsökkentve a kezdetben nagy hibával rendelkező mintákra a tanítást, túltanulásba vezethet. Másik probléma, hogy a becsült adatok és mérési zajok szintén torzíthatják az adatok valós információtartalmát, így a greedy mintavételezés önmagában nem biztosít optimális megoldást. Ennek feloldására került bevezetésre az alábbi módszer:

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \quad (1.2)$$

ahol a TD errorok által számított prioritások egy α értékével vannak tolvá a sztochasztikus mintavételezés irányába. Az α értéke adja meg, milyen mértékben véletlenszerű a mintavételezési valószínűség, és mennyire támaszkodik a TD error által megadott



1.3. ábra. A PER sematikus ábrája[16].

prioritás értékekre. Kezdetben a $p_i = |\delta_i| + \epsilon$ mentén kerültek számításra a prioritások, ahol az ϵ egy kis konstans érték. A második módszer a δ_t értékek sorba rendezése alapján meghatározott $rank(i)$ szerint történik, ahol nem a konkrét $|\delta|$ adja meg a prioritást, hanem a rendezés utáni pozíciója. Ebből az eredményezett tényleges prioritás az alábbi módon számítható:

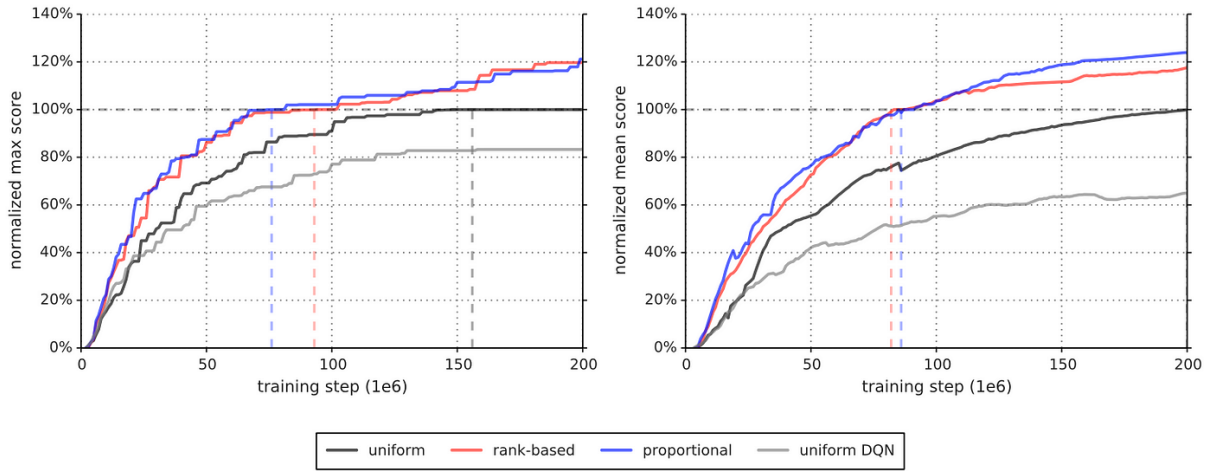
$$p_i = \frac{1}{rank(i)} \quad (1.3)$$

A publikáció az importance-sampling segítségével arra is megoldást nyújt, hogy a priorizált visszajátszás miatt torzított valószínűségek korrigálhatóak legyenek, jobban közelítve a valódi eloszláshoz. A PER először a DQN esetre került alkalmazásra, de nem sokkal később fellelhetővé vált folytonos döntési terű algoritmusok esetre is, mint a Soft Actor Critic(SAC)[14]. Ez kiemelkedően fontos, mivel a modern mérnöki kihívások megoldása sok esetben folytonos döntési térben eszközölt beavatkozásokat igényel.

A PER sikerességét tekintve a kutatókban megfogalmazódott az igény további módszerek alkalmazására, a gyorsabb konvergencia vagy a más jellegű bevezethetőség jegyében. A DeepMind az előző eredményeket folytatva 2017-ben publikálta a Hindsight Experience Replay-t (HER)[15], ami lehetővé tette a sikertelen epizódokból tanulást olyan módon, hogy az eredményezett állapotot egy másik célként értékelve úgy dolgozza fel a tanítómintákat, mintha egy másik feladat megoldásához került volna közelebb az ágens. Gyakran használt módszer robotkarok irányításának megtanulására.

A PER alapvető koncepciójában tárgyalt hiányosságok kiküszöbölésére fellelhetőek olyan cikkek, melyben veszteség alapú priorizálást valósítanak meg[17], hogy a zajos minták miatti magas TD errorokat kiküszöböljék, és ne a TD error mértéke alapján tanuljon az ágens, hanem ennek csökkenthetőségét alapul véve.

További módszer az entrópia alapú priorizálás, ahol a bizonytalanság mértékét kifejező $\mathcal{H}(X)$ entrópia adja meg az egyes tanítóminták mintavételezési valószínűségét[19]. Ennek lényege, hogy az ágens ne ragadjon le korábbi szuboptimális megoldásoknál, minél diverzebb tanítómintákat gyűjtsön. A Shannon-entrópia[20] az alábbi módon



1.4. ábra. A PER-ben alkalmazott prioritizálási eljárások összehasonlítása[18].

számítható:

$$\mathcal{H}(X) = - \sum_i P(x_i) \log P(x_i), \quad (1.4)$$

ahol a $P(x_i)$ adja meg az adott kimenetel valószínűségét, míg a $\log(P(x_i))$ határozza meg az információtartamot, ugyanis a kevésbé valószínű állapotátmenetek meglepőbbek az ágens számára. Mivel valószínűségek logaritmusát tekintjük, mindig negatív lesz a $\log(p(x_i))$ értéke, így egy további negatív előjel biztosítja az egyenletben hogy a $\mathcal{H}(X)$ pozitív értékeket vegyen fel. Ennek az eljárásnak a lényege hasonló az eddigiekhez, ugyanúgy az ágenst olyan tapasztalatokkal akarja szembesíteni, melyek számára meglepőek és az eloszlás szerint nem feltétlenül lépné meg.

Az említett prioritizálási módszerek igen széles körű felhasználást fednek le, ugyanis teljesítményük nagyban függ az adott problémától. Ennek ellenére mégis előszeretettel használt módszer a PER, ugyanis olyan metrikát vezet be, ami a legtöbb esetben jól használhatónak bizonyul, valamint ennek az alap koncepciójából indul ki a legtöbb korszerű kutatás ezen a téren.

1.3. Megoldási kísérlet

A dolgozat célja egy olyan tanítóminta prioritizálási eljárás bemutatása folytonos döntési terű ágensok esetére, ami hatékonyabb működést eredményez, mint a tiszta $|\delta|$ alapú PER. Az elvárt viselkedés az RL központi kérdését jelentő exploration-exploitation dilemmához kapcsolódik, mivel a sztochasztikus PER-hez hasonlóan az overfit és a kezdeti nagy TD errorok miatti beragadások elkerülését célozza meg az alkalmazott új metrika. A dilemma feloldása igen fontos, hiszen amennyiben az ágens nem ismeri eléggé környezetét, nem fogja megtanulni az adott állapotokban a helyes cselekvést,

viszont ha minden esetben felfedező lépést tesz, a kívánt konvergencia gyorsulás nem érhető el.

A dolgozatban bemutatott hozzáadott érték egy olyan Upper Confidence Bound(UCB) alapú megoldás a SAC algoritmus használatának esetére, ami a sztochasztikus PER-rel ellentétben nem csupán a véletlen felé tolja vissza a TD errorok mentén számított prioritásokat, hanem a felfedezésért egy olyan tag felel, ahol az adott minta kiválasztásának száma befolyásolja a mintavételezési súlyt.

A bemutatott módszertől elvárt viselkedés, hogy egy univerzálisan használható metodológiaként kínáljon egy stabilabb alternatívát, mint a tiszta $|\delta|$ alapján történő mintavételezés. A felfedező tag segítségével különösen nagy előrelépés elvárt olyan környezetek esetére, ahol nagy mértékű exploráció igénye merül fel a feladat sikerességének érdekében. Ennek szemléltetésére különböző méretű és nehézségű környezeteken kerül összehasonlításra a dolgozat által kínált módszer és a PER, ahol az elvégzett tanítások és a konvergenciagörbék megfigyelésével kirajzolódik a konvergencia sikeressége és a tanulási idő jól láthatóan megjelenik, azonos alkalmazott hiperparaméterek esetén. Az összehasonlítás négy környezeten kerül elvégzésre, amelyek az RL metodológiák összehasonlításában általánosan használt Gymnasium könyvtárban találhatók, ideértve annak MuJoCo környezeteit is. Amennyiben bebizonyosodik az eljárás alkalmazhatósága ezeken a környezeteken, az elért javulások nagy mennyiségű tanítást igénylő valós ipari alkalmazások esetében hatalmas megtakarításokat jelenthetnek.

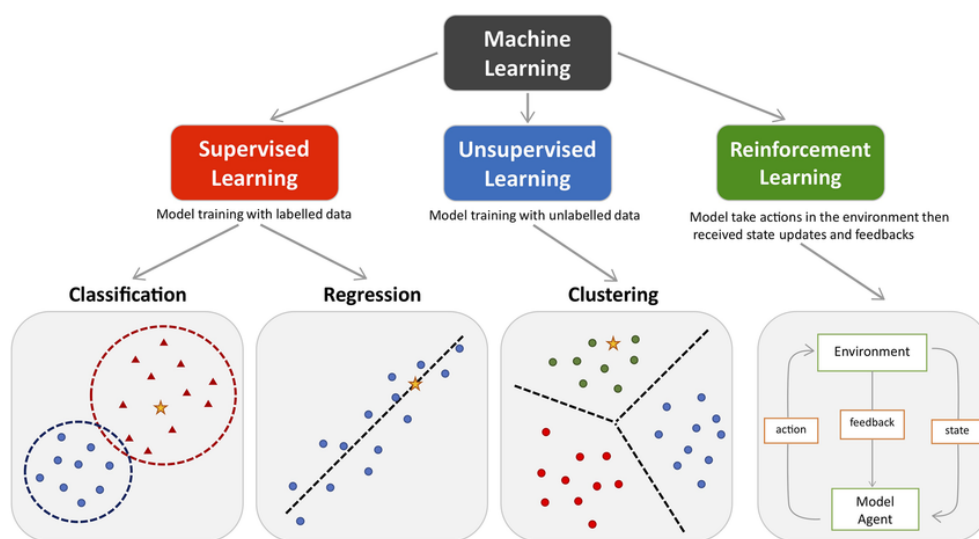
2. fejezet

Algoritmusok

2.1. Gépi Tanulás

A mesterséges intelligencia egyik legjelentősebb alcsoportja az ML, ami 3 fő további csoportra osztható. Ezek a csoportok a Supervised Learning(SL), az Unsupervised Learning(USL) és a Reinforcement Learning(RL), ahogy a 2.1 ábra is szemlélteti.

Az SL a legnagyobb ipari részesedéssel bíró ágazat, melynek fő jellemzője hogy az egyes bemeneti kombinációkhoz címkék vannak rendelve, és az algoritmus feladata a megfelelő címke hozzárendelése az új kapott bemenetekhez. A címkék lehetnek diszkrét vagy folytonos értékűek, ezek alapján határozható meg, hogy az adott feladat regresszió vagy klasszifikáció. Számos felhasználása közül a járműiparban az egyik legjelentősebb fejlődés a számítógépes látórendszerek terén figyelhető meg, ahol a klasszifikáció, a detekció és a szegmentáció feladatok is elláthatóak az SL címkézett adatainak segítségével.



2.1. ábra. A gépi tanulás csoportjai[21]

gével. Az SL algoritmusok alkalmazásában megfontolandó tényező az adatmennyiség meghatározása, hiszen előállítása és címkézése hosszú és erőforrásigényes feladat, így törekedni kell a szükséges legkisebb mennyiségre, viszont amennyiben nincs elég adat, a modell nem tanul meg kellő képpen generalizálni. Bár sok esetben augmentációval a tanító adatkészlet mérete növelhető, bizonyos esetekben nem célravezető a szintetikus adatok alkalmazása. Az SL-hez tartozó fő algoritmusok a Lineáris és Logisztikus Regresszió, Döntési fák, Support Vector Machine, K-Nearest Neighbors, a Naiv Bayes és az NN alapú megoldások.

Az USL szintén előre elkészített adatokkal dolgozik, viszont itt a címkézés folyamata nem része a procedúrának. Ezen a téren a lényeg olyan mintázatok és stuktúrák megállapítása, melyek során az adatok klaszterekbe sorolhatóak, valamint csökkenthető dimenziójuk a rejtett struktúrák feltárásával. A klaszterező algoritmusok közé tartozik a K-means, a Hierarchikus Klaszterezés, míg a dimenziócsökkentő eljárások között szerepel a Principal Component Analysis és a t-SNE.

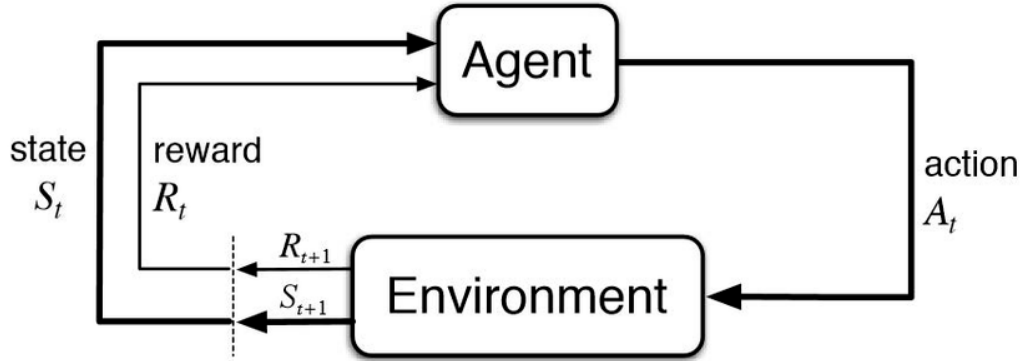
Az előző két csoporttól nagyban eltérő ágazat az RL, hiszen az adatok ebben az esetben tapasztalat formájában jönnek létre. Az eddigi ML felhasználások mellett egy egészen más alkalmazási területet nyit meg, hiszen olyan szekvencialis döntéshozásra is alkalmas, ami az emberi teljesítőképességet messze felülmúlja. Bár ipari részese-dése kisebb mint az első két ágazatnak, kedvelt kutatási terület, hiszen számos téren már napjaink legkorszerűbb algoritmusai közé tartoznak a DRL(Deep Reinforcement Learning) módszerek.

2.2. Megerősítéssel Tanulás

Az RL alapvető koncepcióját két objektum kommunikációja képzi, ahol az ágens egy π^* optimális viselkedési forma elsajátítására törekszik, amivel a számára legnagyobb elérhető jutalmat realizálja egy adott környezetben. A beavatkozások diszkrétizált időlépések mentén valósulnak meg, mely lépésekből epizódok állnak össze, ami az adott tanítási iterációt jelenti. Amint a 2.2 sematikus ábrán látható, az ágens a kezdeti S_t állapotból minden időlépésben meglép egy A_t actiont, amire egy S_{t+1} állapotot és egy R_{t+1} jutalmat kap, ami a döntés minőségének a számszerű reprezentációja. Az egyes állapotokat tekintve a π lehet determinisztikus vagy sztochasztikus:

$$\pi(S) = A \qquad \pi(A|S) = P(A|S), \qquad (2.1)$$

ahol az első esetben minden állapothoz hozzá van rendelve egy beavatkozás, amit az ágens meglép, míg a másik megközelítésben egy valószínűség mentén kerülnek meg-



2.2. ábra. Reinforcement Learning[22]

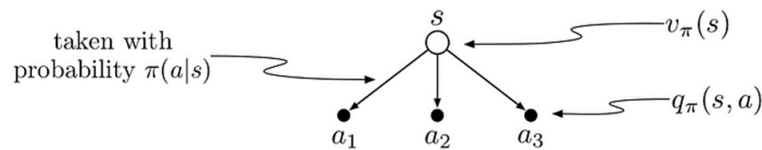
hozatalra az egyes döntések. Minden állapotátmenethez valószínűségek rendelhetőek és ezek segítségével foglалható a Markov Döntési Folyamat keretrendszerébe az RL. A döntési folyamat elemei a következők:

- S : Állapotok halmaza
- A : Megléphető beavatkozások halmaza
- R : S állapotban A beavatkozásra kapott közvetlen jutalom
- P : annak a valószínűsége, hogy S állapotban meghozott A döntésre a rendszer S' állapotba kerül,

melyből összeállítható egy láncolat, ahol az átmenetek egymást követve megadják, milyen állapotból milyen beavatkozást érdemes meglépni az ágensnek, a maximális jutalom elérésének érdekében. Az egyes állapotokhoz a 2.3 ábra alapján $V_\pi(s)$ értékek rendelhetőek, míg az állapotátmenetek minőségét a $Q_\pi(s, a)$ state-action értékkel lehet jellemezni. Az egyes absztrakciók egymást követve láncolatokban írhatóak fel egy epizód során, amik a τ trajektóriák:

$$\tau = (S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_T, A_T) \quad (2.2)$$

ahol a T az adott epizód hosszát jelöli. Az on-policy megközelítésben az ágens rögtön felhasználja a trajektóriákat, és az aktuális π -t kiértékelve határozza meg a tanítás



2.3. ábra. Az állapotátmenetek és egyes absztrakciók kapcsolatának szemléltetése[22]

irányát. Egy másik megközelítés az off-policy RL, ahol nem az adott stratégia kerül minden egyes epizódban kiértékelésre, hanem az ágens egy bufferben tárolja az adatokat, és onnan véletlenszerű mintavételezéssel valósítja meg a hálózatok hangolását. Az egyes állapotátmenetek tárolása az alábbi formában történik meg:

$$Experience = (S, A, R, S', D) \quad (2.3)$$

ahol a korábban bevezetett értékek mellett az epizód végét jelző Done flag D -vel szerepel. Ennek előnye, hogy megtöri az állapotátmenetek közti korrelációt, így az ágens csak arra koncentrál minden egyes S állapotban, mi a legmagasabb realizálható R a számára, és abba az S' állapotba törekszik jutni, aminek során ezt a jutalmat realizálni tudja.

Abból adódóan, hogy az ágens feladata nem csupán egyetlen jó döntés meghozása egy adott környezetben, hanem minden esetben a maximális elérhető R_t realizálása, így a rewardok epizódon belüli szekvenciájának kiértékelésére van szükség. Érdeemes még belátni, hogy nem minden R_t azonos mértékben befolyásolja a tanulást, az adott állapothoz közelebb lévők nagyobb mértékben határozzák meg az ágens viselkedését, így bevezetve a γ discount factort ezek magasabb értéke prioritizálható a csupán későbbi lépésekben szerzett rewardokkal szemben. A reward kumulált értéke a G return, ami az alábbi módon számítható:

$$G(\tau) = \sum_{t=0}^{\tau} \gamma^t r_{t+1} \quad (2.4)$$

ahol a γ mivel 1-nél kisebb értékre van beállítva, a τ korábbi rewardjai nagyobb mértékben felelősek a $G(\tau)$ értékéért.

Az algoritmusok két fő családjá a modell alapú és a modell nélküli RL. Egyszerűsége miatt az utóbbi szélesebb körben használatos megközelítés olyan irányítási feladatokra, ahol nem szükséges a modell működésének pontos és részletes ismerete, elfogadható a black box működés. Ezen belül is létrehozható 3 csoport. A policy gradient algoritmusok esetén az ágens közvetlenül a π stratégiát tanulja. Ilyen algoritmusok a REINFORCE, a Cross-Entropy Method és a Evolution Strategies. Az érték alapú algoritmusok használata során az állapotátmenetekhez tartozó $Q(s, a)$ értékek szerint valósul meg a döntéshozás, ami a Bellman-egyenlet segítségével frissíthető:

$$Q(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q(s', a'), \quad (2.5)$$

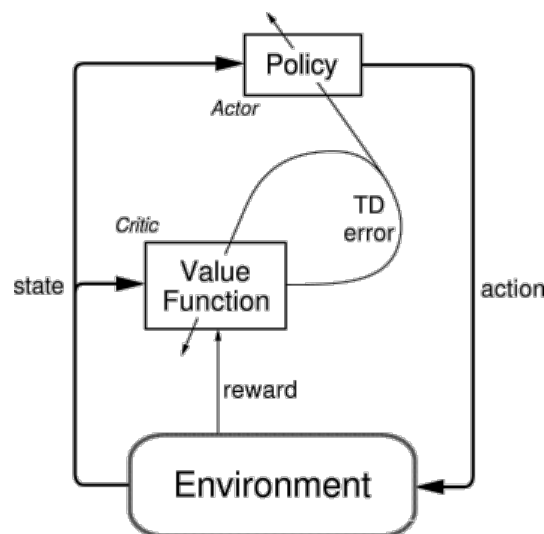
bal oldalon $Q(s, a)$ -val jelölve az aktuálisan számított state-action értéket, míg jobb oldalon az állapotátmenethez tartozó elérhető jutalom látható, valamint az állapotát-

meneti valószínűségek szorzatösszege az elérhető maximális $Q(s', a')$ értékekkel, ezzel az adott állapotátmenetek hasznosságát előállítva mint egy várható érték. Az állapotátmenetekhez tartozó $Q(s, a)$ értékek valamint a target értékek alapján számítható a veszteség, ami alapján a hálózat approximálja az optimális $Q^*(s, a)$ értékfüggvényt. Az legfontosabb Q-Learning algoritmusok a Deep Q-Network[11], a SARSA és ezek módosulatai mint a Double Q-Learning[23].

Mindkét ágazat algoritmusai rendelkeznek előnyökkel és hátrányokkal, ennek köszönhetően alakult ki az Actor-Critic(AC) algoritmusok irányzata, ahol az egyes típusok előnyeit kihasználva számos korszerű algoritmus lelhető fel, mint a Proximal Policy Optimization(PPO)[24], az Advantage Actor-Critic(A2C), a Deep Deterministic Policy Gradient(DDPG)[25] és a Soft Actor-Critic(SAC)[14].

2.3. Actor-Critic algoritmusok

Az AC megközelítés kedveltségének fő oka a feljebb említett hiányosságokban rejlik, ami mind az érték, mind pedig a policy alapú algoritmusokat érinti, csupán más jellemzők terén. Az érték alapú algoritmusok egyik fő hátránya, hogy a folytonos kimenetet igénylő problémákra közvetlenül nem alkalmazhatóak. Emellett viselkedésük stabil és jellemzően gyorsabban konvergálnak, köszönhetően a mintahatékonyságnak, ami az alkalmazott memóriának és a visszajátszható tanítómintáknak a pozitív hozománya. Számos jó tulajdonsága mellett azonban továbbra is alkalmatlan a folytonos akciótér kezelésére. A policy gradient algoritmusok bár jól kezelik a folytonos akciótérrel, mivel nem játszhatóak vissza a tanítóminták amennyiben nem kerül sor buffer alkalmazására, a konvergencia gyorsasága elmarad az érték alapú implementációkhoz képest. Ezek

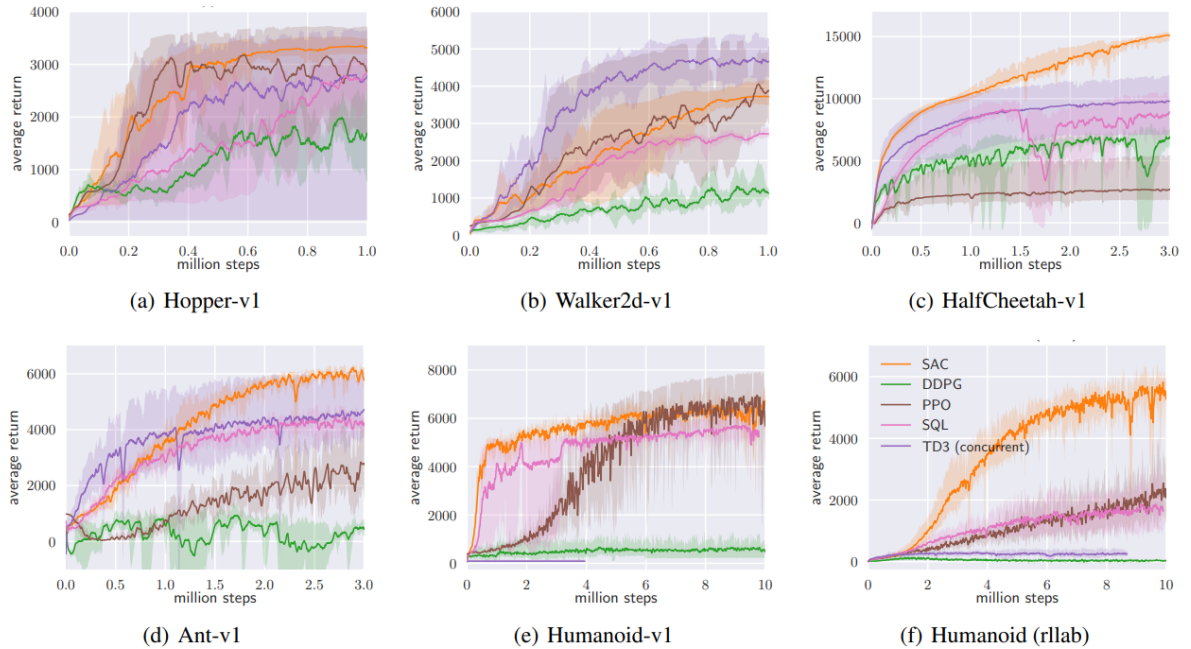


2.4. ábra. Az actor-critic algoritmusok működésének sematikus ábrája[26].

mentén fogalmazódik meg az igény, hogy az egyes ágazatok előnyei együttesen legyenek fellelhetőek egy új, hibrid megközelítésben, ami az AC[27].

Az AC algoritmusok felépítése a 2.4 ábrán látható, ahol a környezet mellett 2 egység képezi a rendszer részét. A policy gradient megközelítés az úgy nevezett Actor hálózatban jelenik meg, ami a tényleges döntéseket meghozza. Ez a feladat jellegétől függően lehet folytonos és diszkrét kimenetekkel rendelkező hálózat. A másik elem a Critic, ami az algoritmustól függően egy vagy több érték hálózat. A Critic feladata a TD error előállítása, aminek segítségével hangolja a saját súlyait valamint az Actor hálózatét. A Critic egy kimenettel rendelkezik ami maga a $Q(s, a)$ érték.

Amint a 2.5 ábrán látható, több AC algoritmus is tartozik a SOTA(State Of The Art) megoldások közé. A tanítóminták prioritizálásához olyan algoritmusra van szükség, ami memóriából tanul, így a kiválasztás során a PPO nem szerepelt a megfontolandó alkalmazások között, ugyanis az egy on-policy algoritmus. Másrésről egy olyan algoritmus tekintetében célszerű további javulást elérni, ami alapvetően is jó mintahatékonysággal rendelkezik. Tekintve a korszerű algoritmusokat és felhasználási területeket, a 2.5 ábra segítségével került meghatározásra a dolgozatban használt eljárás, a SAC ugyanis minden környezetben a legjobbak között teljesített, mind csúcserőtelket tekintve, mind pedig a konvergencia sebeségét vizsgálva.

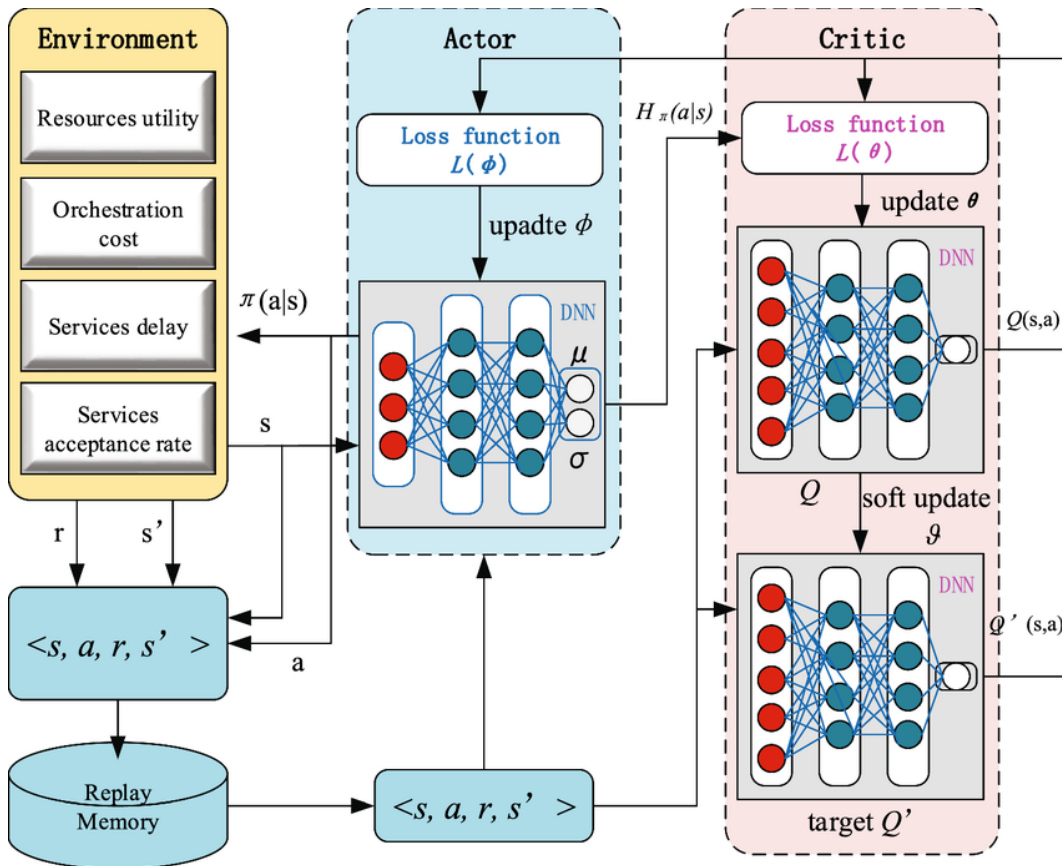


2.5. ábra. A korszerű AC algoritmusok[28].

2.4. Soft Actor Critic

A többi korszerű AC algoritmus közül a SAC számos tulajdonsága miatt kiemelkedik. Hatékonysága abban rejlik, hogy ötvözve a memóriát tartalmazó Actor-Criticet és a maximum entropy RL-t olyan módszer állt elő, ami képes a folytonos kimenet esetén is hatékony mintavételezés és robusztus tanulás megvalósítására, kis érzékenységet mutatva a modell becslési hibáinak tekintetében[29].

A maximum entropy RL a bevezetőben bemutatott módon működik. Az optimális döntések és felfedezések közötti egyensúlyt az 1.4 egyenletben leírt entrópia tényező biztosítja. Ennek segítségével a tanítás során nem csak olyan minták kerülnek be a rendszerbe, amik az adott tudás szerinti optimális viselkedést valósítják meg, hanem a felfedezés megtörténik olyan irányokba is, amiket az ágens csupán a valószínűséget tekintve nem fedne le. A szélesebb körű felfedezés jó módja a lokális szélsőértékek elkerülésének, szuboptimális pontokba való beragadásoknak. A SAC estében a $\mathcal{H}$ az α temperature factorral van súlyozva, mely bár lehet egy statikus hiperparaméter is, de jellemzően érdemes egy csökkenő tendenciát választani neki, hogy a kezdeti felfedezési szakaszt követően a végére a hálózat finomhangolása maradjon. A sztochasztikus jel-



2.6. ábra. A Soft Actor Critic algoritmus[30].

leget bevezetve adódik a $J(\pi)$ célfüggvény aminek értékét maximalizálni igyekszik az ágens. A Bellman egyenlet és entrópia együttesen az alábbi módon adják meg az adott π célfüggvényét:

$$J(\pi) = \sum_{t=0}^{\infty} \mathbb{E}_{\pi} \left[\gamma^t (R(s_t, a_t) + \alpha H(\pi(\cdot|s_t))) \right], \quad (2.6)$$

ahol az aktuális policy alapján várható érték kialakításában az α értékének csökkenésével az entrópia kisebb súlyban vesz részt, ezzel a felfedezési szakaszt követően fokozatosan tolódva a teljesen determinisztikus döntéshozás irányába.

Az algoritmus felépítése a 2.6 ábrán látható, ahol a sztochasztikus $\pi(s|a)$ policy mentén meghozott döntések következtében az állapotátmenetek eltárolásra kerülnek egy memóriában, valamint a Critic hálózatok is hozzáférnek, hogy ezek alapján számíthassák a veszteséget és a TD errorrt a mintavételezett állapotátmeneteken.

A SAC esetében 1 Actor hálózat és 4 Critic hálózat található, melyek közül kettő target hálózatként funkcionál a Loss számításának érdekében. Az ágens n beavatkozáshoz az Actor hálózaton $2n$ kimenet tartozik, mivel a valószínűségi eloszlások a μ átlag és σ varianciaként jelennek meg, amiből aztán a sztochasztikus policy választ egy beavatkozást. A target hálózatok súlyainak frissítése ellentétben a DQN-ben alkalmazottal soft megközelítés mentén valósulnak meg. A soft update esetén a súlyok frissítését a τ értékével lehet szabályozni:

$$\theta_{\text{target}} \leftarrow \tau \theta + (1 - \tau) \theta_{\text{target}} \quad (2.7)$$

aminek segítségével az adható meg, az egyes hálózatok súlyai milyen mértékben határozzák meg a target hálózat következő súlyait. A két-két modell és target hálózat alkalmazásának előnye, hogy csökkenthető a túlbecslés, ugyanis a két hálózat közül ahogyan a 2.8 egyenletben látható, mindig a konzervatívabb vesz részt a valós target érték kiszámításában, melynek kisebb a becsült értéke.

$$Q_{\text{target}} = r + \gamma \min(Q1(s', a'), Q2(s', a')) \quad (2.8)$$

A PER alkalmazásában a TD Error több módszerrel is számítható, mivel 2 érték-hálózat párral rendelkezik a rendszer. A túlbecslés elkerülésére itt is egy lehetőség a kisebb TD Errorral rendelkező hálózatot választani, de sok esetben ezek átlagos értékét alkalmazzák, így a dolgozatban való összehasonlításban ez szerepel majd.

2.5. Alkalmazott neurális hálózatok

A tanítás során 5 NN került használatra, melyek bár architektúrájukban nem teljesen homogének, egyaránt igényelnek néhány feltételt a stabil konvergencia eléréséhez. A vanishing és exploding gradient jelenség elkerülésére egy gyakran alkalmazott megoldás a súlyok inicializálása. Ez a He és Xavier[31] inicializálás segítségével szokott történni, melyek közül a dolgozatban szereplő algoritmus az utóbbit alkalmazza. Ez az alábbi módon írható le:

$$\sigma_{\text{Xav}} = \frac{2}{n_i + n_o}, \quad (2.9)$$

ahol az n_i a bemenő, az n_o pedig a kimenő réteg neuronszámát jelöli. Ez közel normál eloszlást ad, ami segít elkerülni az említett problémát, kiküszöböli a "dead neuron" jelenséget.

A robbanó gradiens elkerülésére egy további módszer a gradiens számítását követően a clip függvény meghívása, ami egy bizonyos intervallumra szorítja vissza a gradienseket. Ezzel nem csak a kezdeti fázisban van védelem implementálva a gradiens felrobbanása ellen, hanem a tanítás közben is biztosítva van a stabilitás. Természetesen ez lassabb tanításhoz vezet, de különösen érzékeny környezetek tekintetében nagy mértékű javulást eredményez.

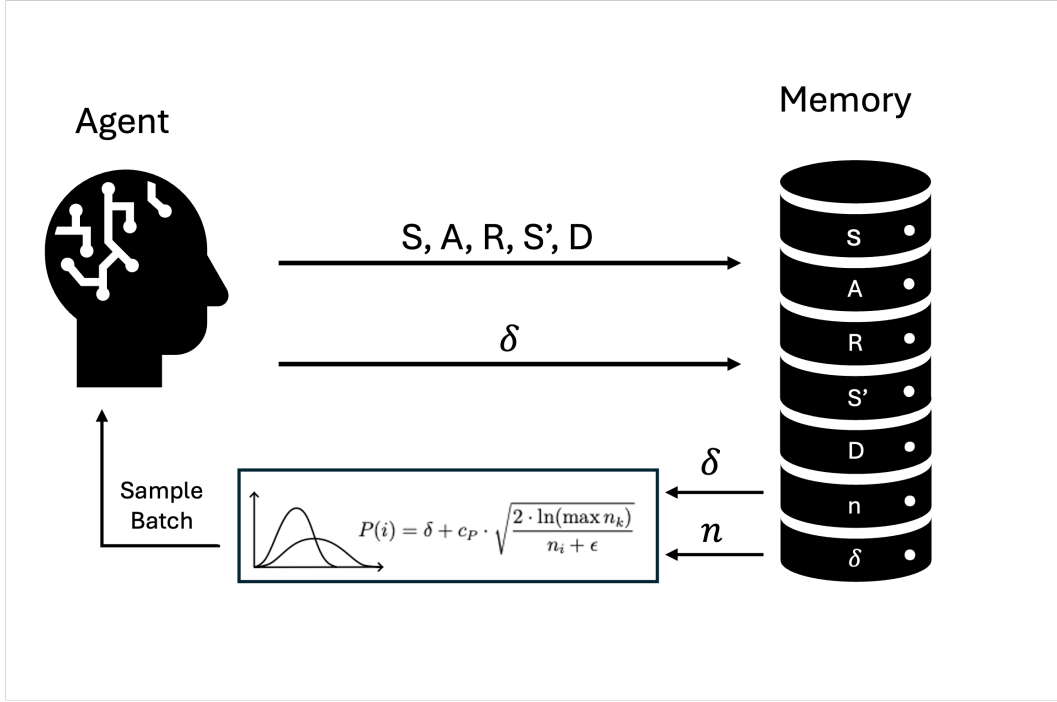
2.6. UCB alapú priorizálás

A tiszta TD error alapú mintavételezés problémáit figyelembe véve fogalmazódik meg az igény egy olyan eljárásra, ami a gyors tanulás mellett figyelmet fordít az explorációra is. A korábbi publikációval ellentétben a kínált módszer nem a véletlenszerűség irányába tolja vissza a mintavételezési valószínűséget, hanem egy külön taggal biztosítja az algoritmus számára a felfedezési állapotátmenetek kiválasztási számának a magasán tartását, ezzel elkerülve a kezdeti beragadásokat.

A PER-ből kiindulva a greedy tag, ami a legnagyobb információtartamot jelenti maradt a TD error, hiszen ez alapvetően egy működőképes metrika, csupán néhány hátrányának kiküszöbölésére kerül bevezetésre a másik része az egyenletnek. A felfedezésért felelős formula az Upper Confidence Bound(UCB) algoritmusból származik, ami többek között a Monte-Carlo Tree Search(MCTS) esetében is alkalmazott módszer az explore-exploit dilemma feloldására. Számítása az alábbi módon tehető meg:

$$explore = c_P \cdot \sqrt{\frac{2 \cdot \ln(N)}{n_i + \epsilon}}, \quad (2.10)$$

ahol az N az MCTS alkalmazásában a legtöbbet látogatott csomópont, jelen esetben



2.7. ábra. A tanítóminták kiválasztása

pedig a legtöbbet mintavételezett állapotátmenet. Ennek logaritmikus növekedésének köszönhetően lassú lesz az érték változása, így szignifikánsabb tud maradni a nevezőben szereplő tag. Az n_i az aktuálisan vizsgált mintának az addigi mintavételezési száma. Az egyenletben továbbá megfigyelhető egy c_p konstans, ami a felfedezési tag súlyát adja meg a valós prioritások számításának tekintetében.

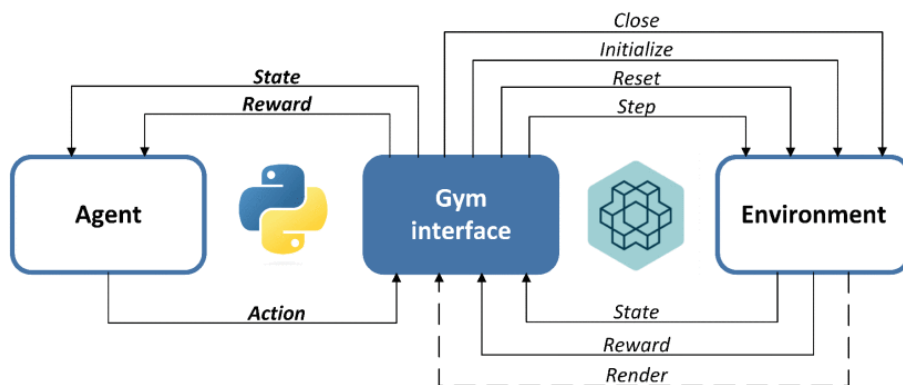
Így hogy már mindkét szükséges tag előállt a tanításhoz, látható hogy csupán 1 hiperparaméter bevezetése szükséges hasonlóan a sztochasztikus PER-hez. A c_p hangolása hasonló hiperparaméter keresési probléma, mint Learning Rate és γ optimális értékeinek meghatározása. A 2.11 egyenlet mentén számított prioritások ennek köszönhetően különböző c_p értékek mentén futattásra kerülnek, és megtörténik az összehasonlítás azzal az esettel, ahol csak a δ értéke kerül számításra, ezzel vizsgálva a felfedezés jelentőségét a mintavételezés során.

$$P(i) = \delta + c_P \cdot \sqrt{\frac{2 \cdot \ln(\max n_k)}{n_i + \epsilon}} \quad (2.11)$$

3. fejezet

Környezetek és a módszerek összehasonlítása

Az algoritmusokon túl az RL feladatok sikeres elvégzéséhez igény van az ágens környezetének megfelelő leírására, a szükséges szenzoradatok rendelkezésre bocsátására, amik alapján meghozhatóak a beavatkozások. Az algoritmusok könnyű és hatékony tesztelhetőségének érdekében merült fel az RL területén az igény olyan környezetekre, amik szabványos bemenetekkel és kimenetekkel rendelkeznek, ezzel amennyiben erre kompatibilisre van megírva egy algoritmus, lehetővé teszi a teljesítményének kiértékelését, összehasonlítva más megoldásokkal. Ezt az igényt a Gymnasium struktúra [32][33] elégíti ki, ami a 3.1 képen látható módon valósít meg kommunikációt a környezettel. Az ágens 4 függvényen keresztül kommunikál a környezet felé információt, melyekkel létrehozza vagy lezárja a kapcsolatot, visszaállítja alaphelyzetbe valamint megvalósít diszkrét időben egy lépést, ezt elküldve a környezet számára. Erre válaszul 3 visszatérő jel figyelhető meg a környezetből az ágens irányába, melyek a visszacsatolt S_{t+1} állapot, az R_{t+1} azonnali jutalom valamint az opcionális vizuális megjelenítésért felelős



3.1. ábra. A Gym interfész[34]

render függvény.

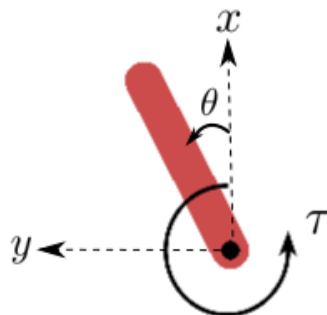
A könyvtár számos környezettel rendelkezik, lefedi a diszkrét és folytonos bemeneti és kimeneti halmazú problémákat. Tartalmaz implementációkat olyan egyszerű irányítási esetekre, mint az inverz inga, de egészen komoly környezetek is elérhetőek benne, mint a Mujoco alapú robotikai feladatok, ahol az ágensek dolga egy robot csuklóira adott nyomatékok együttes összehangolt irányítása. Ennek mentén került kiválasztásra pár különböző nehézségű környezet, amivel a cél a nehézségtől függetlenül bemutatni a felvázolt metódus alkalmazhatóságát.

3.1. Pendulum-v1

A Pendulum-v1 a Gymnasium könyvtár Classic Control környezetei közé tartozik, ahol az ágens feladata az adott állapotokban olyan τ nyomaték eszközölése, aminek hatására a rudat függőleges állapotba hozza, és ott is tartja. Az állapotreprezentáció egy olyan 3 elemű vektor, melynek elemei a következők:

$$state_i = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ \dot{\theta} \end{bmatrix}, \quad (3.1)$$

ahol az első két elem az x és y irányú pozíciók jelölésére szolgálnak, míg a harmadik állapotváltozó a szögsebességet jelöli. A szögfüggvények korlátait tekintve az első két érték a $[-1.0, 1.0]$ intervallumon vehet fel folytonos értéket, míg a $\dot{\theta}$ értéke $[-8.0, 8.0]$ intervallumon belül mozog. Az action space elemei $[-2.0, 2.0]$ intervallumon vehetnek fel értéket, melynek kezelése a wrapperben valósul meg, ugyanis a logaritmus számításánál a negatív értékek nem adnak megoldást a valós számok halmazán.



3.2. ábra. Pendulum-v1

A jutalmazás tekintetében a környezet három törekvést valósít meg, ami az alábbi egyenletből adódik:

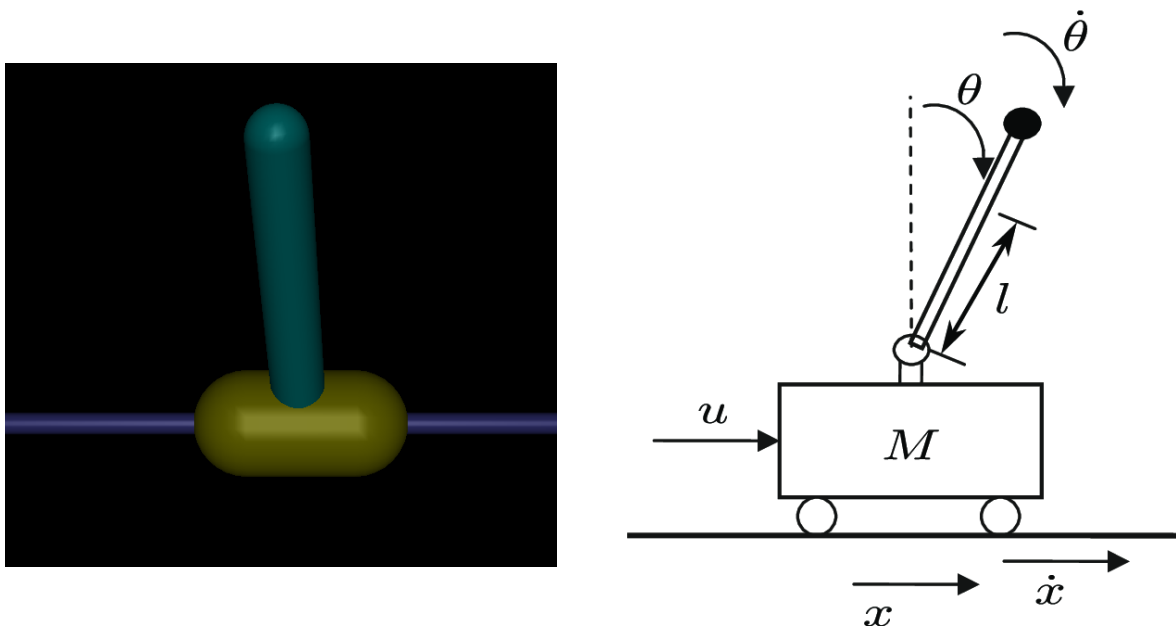
$$r = -(\theta^2 + 0.1 \cdot \dot{\theta}^2 + 0.001 \cdot u^2), \quad (3.2)$$

ahol az ágens egyszerre igyekszik a függőleges állással bezárt szöget minimalizálni, minél kevesebb szögsebesség mellett, valamint a minimális befektetett τ érték mellett mindezt megvalósítani.

3.2. InvertedPendulum-v5

Klasszikus környezetek esetén az egyik legkedveltebb mind irányításelméletben, mind pedig az RL világában az inverz inga[35] problémája. Az előző feladathoz nagyban hasonlít, csupán annyi eltérés van benne, hogy itt nem a lógó helyzetből való függőlegesbe hozatal a cél, hanem egy bizonyos $[-\theta, \theta]$ függőlegestől való eltérésen belül tartani a rudat, valamint nem egy fix pontra adott nyomatékkal valósul meg az irányítás, hanem egy kocsi jobbra vagy balra tolásával, melyet az ágens minden diszkrét időlépésben megtehet a $[-3.0, 3.0]$ N erőhatás intervallumon.

A gymnasium könyvtár alapvetően diszkrét kimenetű CartPole környezettel rendelkezik, de a Mujoco fizikai alapú szimulátor integrálásának köszönhetően számos más környezettel együtt egy folytonos beavatkozási terű CartPole is elérhetővé vált.



3.3. ábra. InvertedPendulum-v5

Az állapotváltozókat tekintve egy négy elemű folytonos értékeket tartalmazó vektor adja a méréseket, az alábbi módon:

$$state_i = \begin{bmatrix} x & \theta & v & \dot{\theta} \end{bmatrix}^T, \quad (3.3)$$

ahol az x a kocsí pozícióját jelzi, a θ és a $\dot{\theta}$ hasonlóan a Pendulum-v1 esetén alkalmazott függőleges eltérés szöge és szögsebessége, míg jelen esetben a kocsí sebessége is részét képi az állapotreprezentációnak.

Jutalmazását tekintve egy egyszerűbb modellről van szó, itt is minden időlépésben kap jutalmat az ágens, viszont az alábbi módon kerül számításra:

$$|\text{angle}| < 0.2, \quad (3.4)$$

amiből látható, hogy az ágens arra törekszik, hogy minél több időlépés alatt tudja a megadott intervallumon belül tartani a rudat. Az epizód 1000 időlépés után véget ér, ahol minden sikeres lépésért 1 pontot kap, így a környezet korlátját is 1000 pont képi.

3.3. DoubleInvertedPendulum-v5

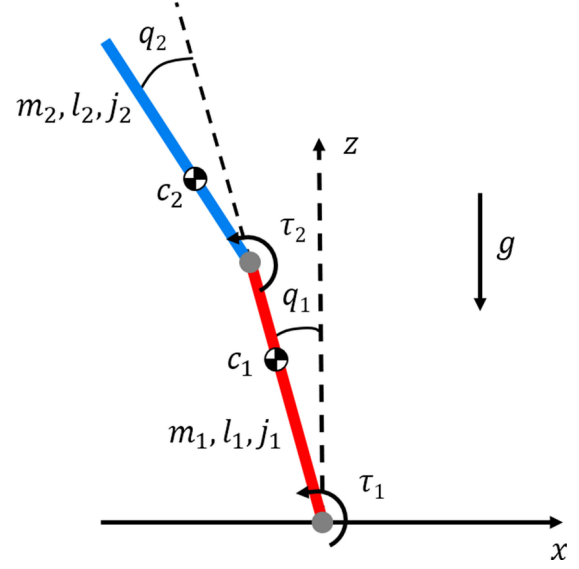
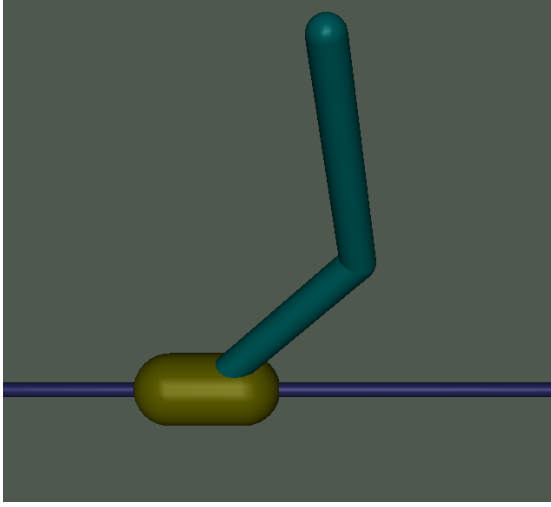
A dupla inverz inga környezete hasonló az előző részben tárgyalthoz, csupán ahogy a 3.4 képen is látható, itt a vízszintes erőhatások a két rúd függőlegesen tartását célozzák meg. Abból kifolyólag, hogy ez egy több komponensből álló környezet, nagyobb állapottér szükséges a leírásához is. A bemeneten 9 folytonos értékeket felvenni képes neuron helyezkedik el, melyek az alábbi fizikai jellemzőket takarják:

$$\begin{bmatrix} x & \sin(\theta_1) & \sin(\theta_2) & \cos(\theta_1) & \cos(\theta_2) & \dot{x} & \dot{\theta}_1 & \dot{\theta}_2 & F_x \end{bmatrix}^T \quad (3.5)$$

ahol látható a 3.4 ábrán jelölt két rúd állapota valamint a kocsit érő erőhatás mint bemenő érték. Mivel az ágens feladata változatlan az egyszerű inverz ingához képest beavatkozások terén, itt is 2 kimeneti neuronnal rendelkezik az NN. Ebben az esetben, eltérően az InvertedPendulum-v5 környezettől ez az érték a $[-1.0, 1.0]$ intervallumra van beszorítva, ami kedvező, ugyanis a wrapperben nem kell külön kezelni ezt a környezetet. A jutalmazás tekintetében is olyan összefüggés használata szükséges, mely mindkét rúd állapotát leírja. Az alapértelmezett reward 3 tagot foglal magába, melyek közül egy az életben maradásért felel, 2 pedig a minél kisebb mozgások megvalósításáért. Ez az alábbi módon néz ki:

$$\text{reward} = H_r - D_p - V_p \quad (3.6)$$

aminek az egyes tagjai a következő képpen számíthatók:



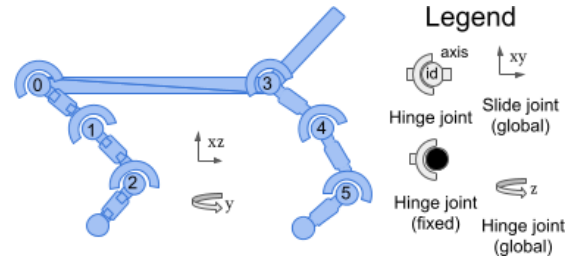
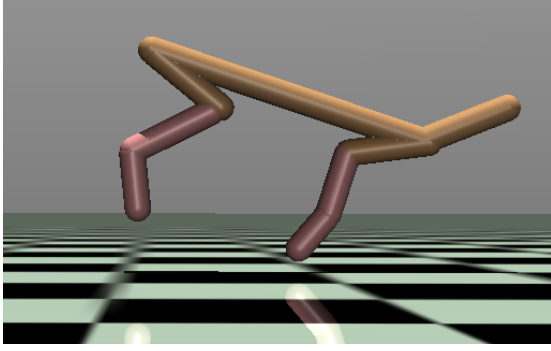
3.4. ábra. InvertedDoublePendulum-v5[36]

- H_r (Healthy reward): Az első tag minden olyan időlépésért jutalmazza az ágenst, ahol nem ér véget az epizód a szabályok sérülése miatt. Ez alap beállításként 10 pont minden sikeres lépésben.
- D_p (Distance Penalty): Ez az érték a felső rúd elmozdulását figyeli és bünteti az eltérés függvényében az alábbi egyenlettel számítva:

$$D_p = 0.01x_{\text{pole2-tip}}^2 + (y_{\text{pole2-tip}} - 2)^2$$
- V_p (Velocity Penalty): Megelőzve a hirtelen nagy váltásokat, a sebesség az alábbi taggal büntethető: $V_p = 10^{-3}\omega_1 + 5 \times 10^{-3}\omega_2$ amiből látszik, hogy az ágens feladata olyan viselkedés megvalósítása, ahol mind az első rúd, mind pedig a második a lehető legkisebb szögsebességgel mozog.

3.4. HalfCheetah-v5

Az eddig bemutatott egy beavatkozóval rendelkező környezetek mellett egy olyan esetben is összehasonlításra kerülnek a módszerek, ahol határozottan sokkal szélesebb körben szükséges exploráció, valamint egyszerre több kimeneti neuron felelős egy robot csuklóinak nyomatékaiért. Erre a MuJoCo és a Gymnasium által létrehozott gymnasium-robotics könyvtár HalfCheetah[37] környezete került alkalmazásra, ahol a 3.5 ábrán látható módon helyezkednek el a motorok. Az adható nyomatékok a $[-1.0, 1.0]$ intervallumon belül helyezkednek el minden csukló esetében. A cél ezeknek az összehangolt mozgásával megvalósítani, hogy a kutya a lehető legnagyobb sebességgel fusson előre, ameddig az epizód véget nem ér.



3.5. ábra. HalfCheetah-v5[38]

Az állapotrepresentáció egy 17 elemű vektor, ahol az egyes elemek a bezárt szögek, a sebességek x és z irányban, valamint az egyes komponensek szögsebességei. Jutalmazás terén az elért cél és a befektetett munka vesznek részt, az alábbi módon:

$$r = w_{\text{forward}} \cdot \frac{dx}{dt} - w_{\text{control}} \cdot \|\text{action}\|_2^2, \quad (3.7)$$

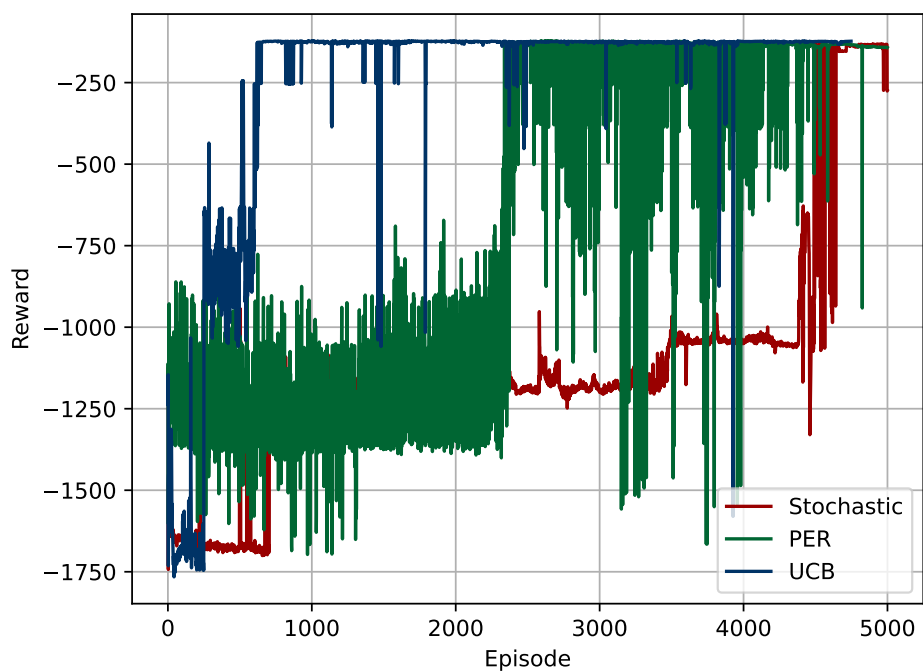
ahol a w_i súlyok a haladás és befektetett energia mértékét súlyozzák, mennyire legyenek szignifikánsak a jutalom kialakításában. Az haladás a kutya x irányú pozíciójának differenciájával számítható, míg a befektetett energiát reprezentáló büntető tag a beavatkozások vektorának Euklideszi normájának négyzetét minimalizálja. Ezzel az egyenlettel biztosítva van az ágens számára minden szükséges visszacsatolás, hogy a minimális beavatkozások mellett valósulhasson meg a legnagyobb haladás. Az epizód 1000 lépés után ér véget, akkor is végig fut, ha a kutya felborul.

4. fejezet

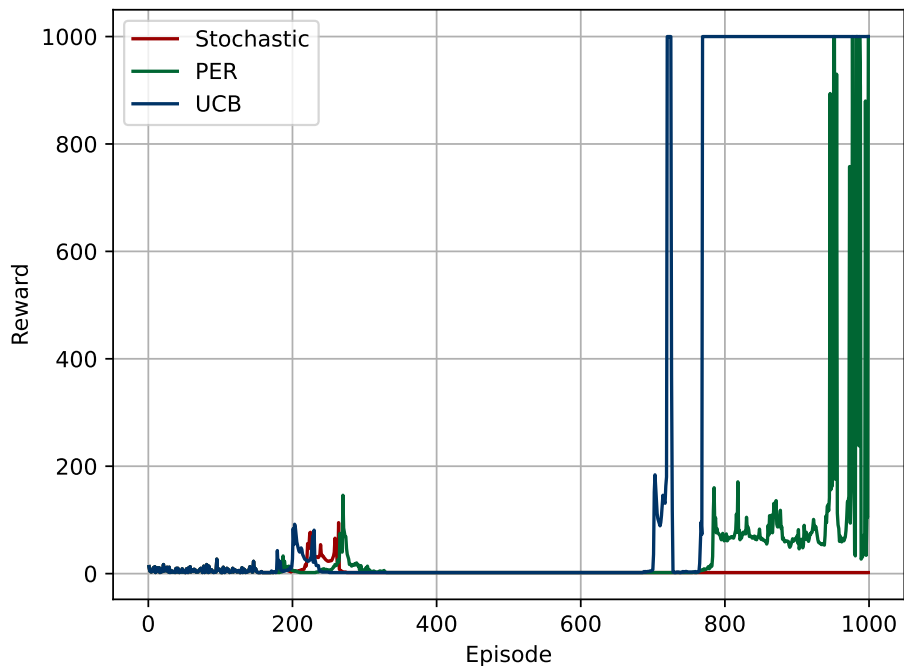
Eredmények

Annak érdekében, hogy az összehasonlítás teljeskörű legyen, számos hiperparaméter beállítás mellett kiértékelésre kerültek az algoritmusok. Ezek közül a konvergencia sebességén kívül a stabilitás is meghatározta, melyik tanítás kerül be az összehasonlító analízisbe. A kívánt eredmény megvalósulásához vezető paramétereket megtalálva több különböző seeden is egymás mellé lettek állítva a módszerek, hogy további bizonyosságot adjon az algoritmusok teljesítőképességének.

Az első vizsgált környezet a Pendulum-v1 volt, aminél láthatóan mind a sztochasz-



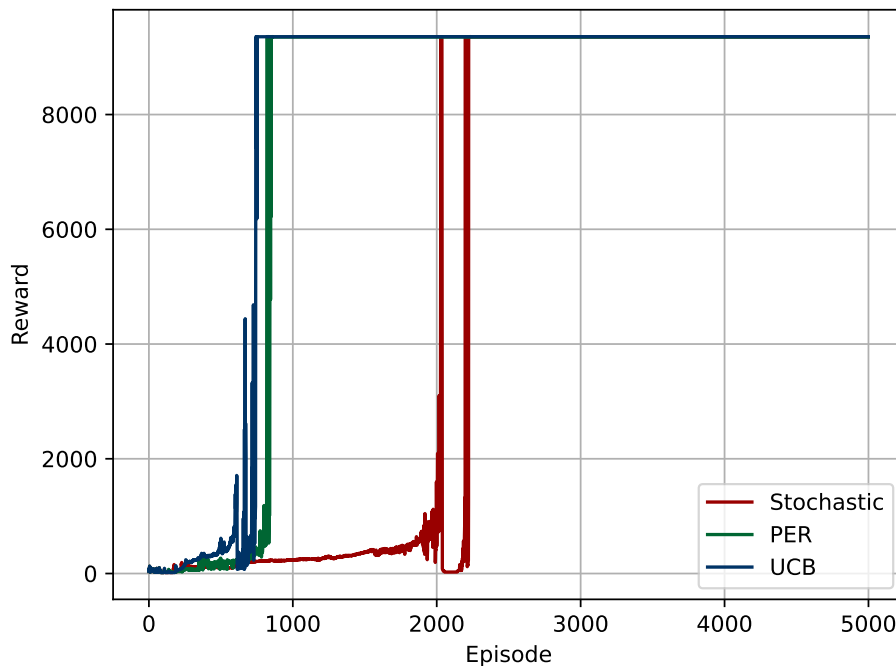
4.1. ábra. Pendulum-v1



4.2. ábra. InvertedPendulum-v5

tikus mintavételezés, mind pedig a PER és az UCB alapú algoritmus is sikerrel járt. A 4.1 konvergenciagörbéket tekintve azonban egyértelműen kirajzolódnak a dolgozat témáját képző priorizálási módszerekben rejlő lehetőségek. Míg a tiszta TD error alapú priorizálás hamarabb konvergált mint a priorizálás nélküli megközelítés, az új metodológiát jelentő UCB kevesebb zajjal, jóval korábban és a diagramon láthatóan kicsivel magasabb értékre állt be.

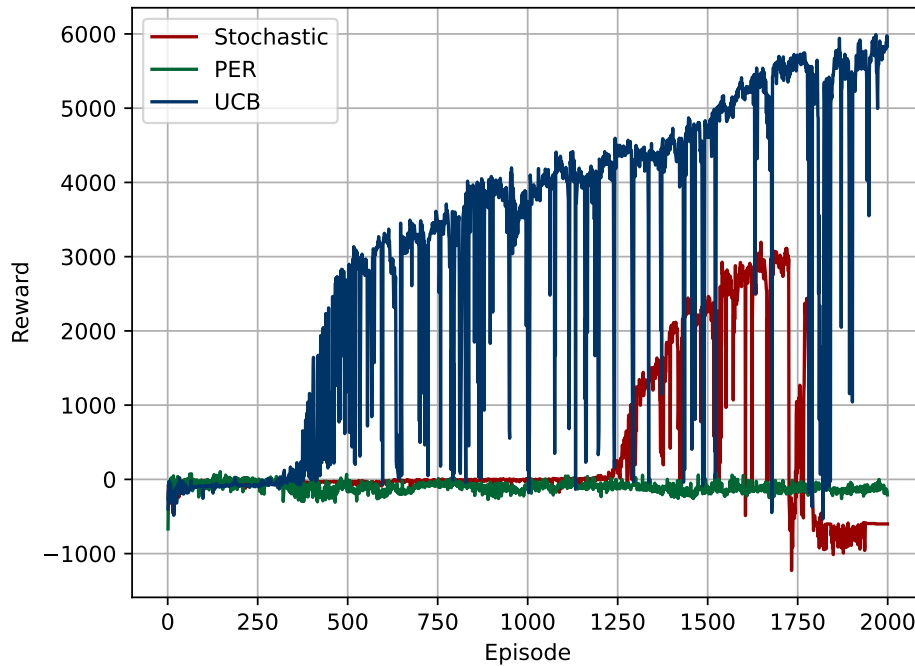
A második vizsgált környezet esetében némileg eltérő eredmény látható a grafikonon. Az InvertedPendulum-v5 környezet jól mutatja, hogy amennyiben kisebb értéket vesz fel az entrópiát szabályzó α , az egyéb hiperparaméterekre rendkívül érzékennyé válik a sztochasztikus mintavételezés esete. Mivel a feladat egy szűk szögen belül tartani a pálcát, a kezdeti szakaszban nagyon sokszor hirtelen sérül ez a kritérium, így kis beavatkozásokra is közel 0 jutalommal zárja az ágens az epizódot, ezért egy rendkívül érzékeny környezetről van szó. Abból adódóan, hogy a tanítás elején ritka a használható minta, nem érdemes csupán arra a kis valószínűségre bízni a mintavételezését, ami a sztochasztikus esetben jut neki. Természetesen ebben az esetben is sikerült olyan hiperparaméterhangolást találni, ahol a sztochasztikus mintavételezés is sikerrel járt, viszont az érzékenységet jól szemlélteti a 4.2 ábra. A diagramról jól leolvasható, hogy nem csak a konvergencia ideje rövidült ebben az esetben, hanem az oszcillációk is kisebb mértékben fordulnak elő a dolgozat témáját képző megközelítés esetén.



4.3. ábra. InvertedDoublePendulum-v5

Az InvertedDoublePendulum-v5 környezetet tekintve látható, hogy a konvergencia görbék jellege a feladat hasonlóságából adódóan nagyban egyezik az egyszerű inverz ingáéval. Itt egy tágabb időablakban lett elvégezve a tanítás, aminek során látható, hogy a jutalmak tekintetében minden megközelítés képes volt a környezet korlátait jelentő értéket megtalálni, ami az 1000 időlépés során elért nagyjából 9300 pont. Míg a sztochasztikus mintavételezésnek hozzávetőlegesen 2000 epizód alatt sikerült megvalósítani a konvergenciát, a TD error alapú mintaválasztás kevesebb mint fele annyi epizódot igényelt. Bár szemmel kevésbé látható, de itt is hamarabb bekövetkezett a konvergencia, így az egyre nagyobb állapottér ellenére mind a 3 környezeten sikerrel járt a kínált priorizálási eljárás.

A HalfCheetah környezet eredményeit szemléltető 4.4 diagramon nem várt eredmények jelennek meg. Az eddigi összes esetben jobban teljesített a PER a sztochasztikus mintavételezésnél, viszont ezen a környezeten megmutatkozik, miért is van szükség más jellegű metrikák bevezetésére a széles körű explorációt igénylő környezetekben. Mivel az ágens nagyon diverz állapotokat vehet fel és 6 beavatkozáson kell optimális nyomatókot kiadni, így az eddigi problémákhoz képest rendkívül jelentős erőforrást kell allokálni a felfedezésre. Ez a magasabb entópiával meg is történt, viszont a mintavételezés során már ez a diverzitás nem elmondható, hiszen csak a nagy kezdeti TD errorral rendelkező minták kerültek végig kiválasztásra, így a hálózat nem is tanult be. Látható, hogy



4.4. ábra. HalfCheetah-v5

a sztochasztikus esetben a konvergencia szépen megindult egy idő után, de bőven a környezet határa előtt bezuhant. Ellentétben a másik két módszerrel az UCB azonban sikerrel járt, mind a konvergenciagörbe emelkedésének megkezdését tekintve, mind pedig a csúcértéket vizsgálva.

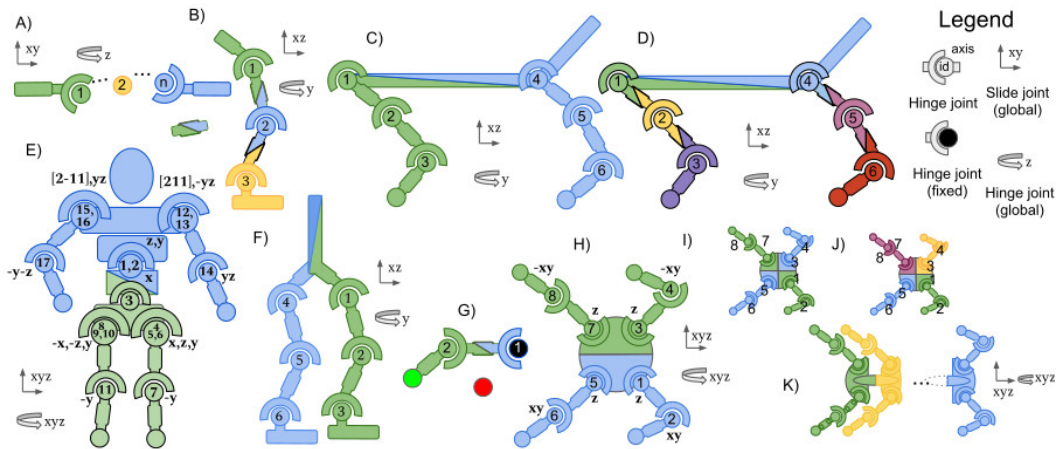
Az eredmények tükrében látható, hogy a TD error-t kiegészítve egy súlyozható explorációs taggal megnő a mintahatékonyság, ugyanis a kizsákmányoló viselkedés mellett egy irányított exploráció is bekerül a rendszerbe. Mivel 4 környezeten és több seeden is előállt a sorrend, kijelenthető, hogy az UCB alapú priorizálás alkalmasabb az összes vizsgált környezeten a konvergenciagörbét nézve, de egyes esetekben a csúcérték is magasabbra kúszik a saját metrika bevetése esetén.

5. fejezet

Konklúzió

A dolgozat célja egy olyan univerzálisan alkalmazható tanítóminta priorizálási eljárás bemutatása volt, ami hatékonyabb megoldást nyújt, mint a napjainkban is előszeretettel alkalmazott TD error alapú PER. A kiértékelést követően megfigyelhető, hogy igen lényeges tényező a mintavételezés során használt metrika, ugyanis egyes esetekben teljesen eltorzíthatja a minták valós információtartalmát és megghiúsítja a konvergenciát, ennek következtében rosszabb eredményt elérve mint az egyenletes eloszlású mintavételezés során. A tiszta TD error alapú mintavételezés okozta nehézségek miatt bevezetett új módszer több környezetben is megfelelőnek bizonyult, ezzel bemutatva, nem környezetfüggő megoldás, hanem széleskörűen bevethető módszer. Különösen a legbonyolultabb probléma esetén vonható le tanulságként, hogy a kezdeti nagy TD errorok okozta beragadás igen káros lehet, amennyiben nem része a mintavételezési valószínűségnek egy explorációért felelős tag.

A sikerek következtében számos kutatási terület nyílik meg, ahol a bemutatott eljárás további tanítási időket tud megspórolni. Kilépvé az összehasonlításra szolgáló kör-



5.1. ábra. Multi-Agent Mujoco[39]

nyezetek világából mind a járművek irányításában, mind pedig a forgalmi rendszerek tekintetében a jellemzően napokban mérhető folyamatokon százalékos nagyságrendű rövidítések már igen szignifikáns eredményekként lehetnének felmutathatóak. Emellett ha csak a felhasznált Mujoco könyvtárat tekintjük, itt is számos lehetőség van még bonyolultabb környezeteken való vizsgálódásra, aminek egyenlőre a számítási kapacitás és a rendelkezésre álló idő szabott határt. Ennek vizsgálata, valamint az egyes robotok több ágensre bontása, ahogy az a 5.1 ábrán látható képezik a kutatás következő lépését. Ezeken a környezeteken további cél annak kiértékelése, hogy a többi metrikával összehasonlítva milyen mértékű javulás érhető el azzal, hogy a felfedezés nem csupán a véletlen felé való visszatolással van reprezentálva, hanem az ágens tudatos módon válogat azok közül a tanítóminták közül, melyek még kevesebbszer kerültek kiválasztásra az optimalizációhoz.

Ábrák jegyzéke

1.1. A korszerű CPU és GPU által elvégezhető maximális lebegőpontos műveletek száma[5].	2
1.2. A szükséges lebegőpontos műveletek és kapcsolódó költségek nagy nyelvi modellek tanítása esetén[8].	3
1.3. A PER sematikus ábrája[16].	5
1.4. A PER-ben alkalmazott priorizálási eljárások összehasonlítása[18]. . . .	6
2.1. A gépi tanulás csoportjai[21]	8
2.2. Reinforcement Learning[22]	10
2.3. Az állapotátmenetek és egyes absztrakciók kapcsolatának szemléltetése[22]	10
2.4. Az actor-critic algoritmusok működésének sematikus ábrája[26].	12
2.5. A korszerű AC algoritmusok[28].	13
2.6. A Soft Actor Critic algoritmus[30].	14
2.7. A tanítóminták kiválasztása	17
3.1. A Gym interfész[34]	18
3.2. Pendulum-v1	19
3.3. InvertedPendulum-v5	20
3.4. InvertedDoublePendulum-v5[36]	22
3.5. HalfCheetah-v5[38]	23
4.1. Pendulum-v1	24
4.2. InvertedPendulum-v5	25
4.3. InvertedDoublePendulum-v5	26
4.4. HalfCheetah-v5	27
5.1. Multi-Agent Mujoco[39]	28

Rövidítések

AI	-	Mesterséges intelligencia
DL	-	Deep Learning
NN	-	Neural Network
GPU	-	Graphical Processing Unit
ML	-	Machine Learning
RL	-	Reinforcement Learning
DQN	-	Deep Q-Network
PER	-	Prioritized Experience Replay
TD	-	Temporal Difference
SAC	-	Soft Actor Critic
HER	-	Hindsight Experience Replay
UCB	-	Upper Confidence Bound
SL	-	Supervised Learning
USL	-	Unsupervised Learning
DRL	-	Deep Reinforcement Learning
AC	-	Actor-Critic

Irodalomjegyzék

- [1] Pranav Rajpurkar, Emma Chen, Oishi Banerjee, and Eric J Topol. Ai in health and medicine. *Nature medicine*, 28(1):31–38, 2022.
- [2] Isaac Kofi Nti, Adebayo Felix Adekoya, Benjamin Asubam Weyori, and Owusu Nyarko-Boateng. Applications of artificial intelligence in engineering and manufacturing: a systematic review. *Journal of Intelligent Manufacturing*, 33(6):1581–1601, 2022.
- [3] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.
- [4] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
- [5] Ivona Vasileska, Pavel Tomšič, and Leon Kos. Modernization of the pic codes for exascale plasma simulation. pages 209–213, 09 2020.
- [6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.
- [7] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.
- [8] Sunder Khowaja, Parus Khuwaja, Kapal Dev, and Weizheng Wang. Chatgpt needs spade (sustainability, privacy, digital divide, and ethics) evaluation: A review, 10 2023.
- [9] Bálint Kövári, Bálint Pelenczei, and Tamás Bécsi. Monte carlo tree search to compare reward functions for reinforcement learning. In *2022 IEEE 16th International*

Symposium on Applied Computational Intelligence and Informatics (SACI), pages 000123–000128. IEEE, 2022.

- [10] Bálint Kővári, Bálint Pelenczei, István Gellért Knáb, and Tamás Bécsi. Beyond trial and error: Lane keeping with monte carlo tree search-driven optimization of reinforcement learning. *Electronics*, 13(11):2058, 2024.
- [11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- [12] Tom Schaul. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.
- [13] Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.
- [14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.
- [15] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. 2018.
- [16] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, H. V. Hasselt, and David Silver. Distributed prioritized experience replay. *ArXiv*, abs/1803.00933, 2018.
- [17] Shivakanth Sujit, Somjit Nath, Pedro H. M. Braga, and Samira Ebrahimi Kahou. Prioritizing samples in reinforcement learning with reducible loss. 2023.
- [18] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2016.
- [19] Mirza Ramicic and Andrea Bonarini. Entropy-based prioritized sampling in deep q-learning. In *2017 2nd international conference on image, vision and computing (ICIVC)*, pages 1068–1072. IEEE, 2017.
- [20] PA Bromiley, NA Thacker, and E Bouhova-Thacker. Shannon entropy, renyi entropy, and information. *Statistics and Inf. Series (2004-004)*, 9(2004):2–8, 2004.

- [21] Junjie Peng, Elizabeth Jury, Pierre Dönnes, and Coziana Ciurtin. Machine learning techniques for personalised medicine approaches in immune-mediated chronic inflammatory diseases: Applications and challenges. *Frontiers in Pharmacology*, 12, 09 2021.
- [22] Richard S Sutton, Andrew G Barto, et al. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1):126–134, 1999.
- [23] Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.
- [24] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [25] TP Lillicrap. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
- [26] Simone Foa, Corrado Coppola, Giorgio Grani, and Laura Palagi. Solving the vehicle routing problem with deep reinforcement learning, 07 2022.
- [27] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.
- [28] Jesse Declercq. The feasibility of reinforcement learning in autonomous driving using synthetic imagery across remote locations, 03 2021.
- [29] Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.
- [30] Zhongwei Huang, Weijian Zhong, Dagang Li, and Lu Hua. Delay constrained sfc orchestration for edge intelligence-enabled iiot: A drl approach. *Journal of Network and Systems Management*, 31, 06 2023.
- [31] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterton, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.
- [32] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

- [33] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. 2016.
- [34] Bálint Kóvári, Bálint Pelenczei, and Tamás Bécsi. Enhanced experience prioritization: A novel upper confidence bound approach. *IEEE Access*, 2023.
- [35] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983.
- [36] Jongwoo Lee, Kuangen Zhang, and Neville Hogan. Identifying human postural dynamics and control from unperturbed balance. *Journal of NeuroEngineering and Rehabilitation*, 18, 03 2021.
- [37] Paweł Wawrzyński. A cat-like robot real-time learning to run. In *Adaptive and Natural Computing Algorithms: 9th International Conference, ICANNGA 2009, Kuopio, Finland, April 23-25, 2009, Revised Selected Papers 9*, pages 380–390. Springer, 2009.
- [38] Farama Foundation. Halfcheetah-v5, 2024. Accessed: 2024-11-02.
- [39] Bei Peng, Tabish Rashid, Christian A. Schroeder de Witt, Pierre-Alexandre Kamieny, Philip H. S. Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients, 2021.