

Óbudai Egyetem
Neumann János Informatikai Kar
Szoftvertervezés és -fejlesztés Intézet



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

On-policy mély megerősítéses tanulás tumorkezelés optimalizálására

Szerzők: Almásy Márton György
mérnökinformatikus MSc. szak, II. évf.
Hörömpő András
mérnökinformatikus MSc. szak, II. évf.

Konzulensek: Dr. Kertész Gábor
egyetemi docens
Kiss Dániel
egyetemi tanársegéd

Budapest, 2022

Tartalomjegyzék

Absztrakt	3
1. Motiváció és célkitűzés	4
2. Irodalomkutatás	6
2.1. A megerősítéses tanulás alapjai	6
2.2. Tumornövekedés matematikai modelljei	8
2.2.1. Naiv megközelítések	8
2.2.2. Angiogenezis hatását figyelembe vevő modellek	8
2.2.3. Tumornövekedés és gyógyszerhatás modellezése	10
2.3. Vizsgált eljárások	12
2.3.1. Q-tanulás	12
2.3.2. Deep Q-Network (DQN)	14
2.3.3. Deep Recurrent Q-Network (DRQN)	14
2.3.4. Csúszó ablakos különbözeteken alapuló DQN	15
2.3.5. Proximal Policy Optimization Deep Q-Network (PPO-DQN)	16
2.3.6. Quantile Regression Deep Q-Network (QR-DQN)	17
2.4. Vizsgált jutalomfüggvények	17
2.5. Konklúzió	18
3. Módszertan	19
3.1. Környezet bemutatása	19
3.1.1. Használt modellek bemutatása	20
3.1.2. Önellő kutatói munka határai	21
3.2. Implementáció nagy vonalakban	21
3.2.1. Környezet implementációja	21
3.2.2. Dosemax és terminálás implementációja	23
3.2.3. Eljárások implementációja jellegük szerint	23
3.3. Végrehajtandó kísérletek	27
4. Eredmények	28
4.1. Konkrét eredmények	29
4.1.1. DQN teszt	29
4.1.2. ARDQN teszt	31
4.1.3. PPO teszt	33
4.1.4. PPO2 teszt	34
4.1.5. QRDQN teszt	35
4.2. Eredmények értékelése	37
4.2.1. Tesztek struktúrája	37

4.2.2.	Módszertanok összehasonlító elemzése	37
4.2.3.	Konklúziók a módszertanokról	40
4.2.4.	Eredmények összegzése	41
5.	Összefoglalás	43
5.1.	Továbbfejlesztési és további kutatási tervek	43
	Ábrajegyzék	45
	Táblázatok jegyzéke	46
	Forráskódok listája	47
	Irodalomjegyzék	48

Absztrakt

A daganatos betegségek továbbra is a vezető halálokok között szerepelnek annak ellenére, hogy az utóbbi években és évtizedekben a kezelési módszerek hatékonysága sok esetben nagyságrendileg javult. A jelenleg széleskörűen elérhető terápiák között általánosan alkalmazzák a gyógyszeres kezelést vagy kemoterápiát. Az utóbbi során a páciens sok esetben hosszútávon is szenved a kezelés mellékhatásaitól, mivel a terápia során az ép és egészséges sejtek egy részére is károsodik. Az általánosan alkalmazott kemoterápiás protokollok alapja jelenleg az, hogy a kezelést a beteg számára még tolerálható maximális dózissal végzik, az adagolt hatóanyag mennyiségével együtt azonban a mellékhatások gyakorisága és súlyossága is növekszik. Épp ezért az elmúlt években több olyan kutatás is indult, amelyek új terápiás protokollok kidolgozását és vizsgálatát tűzték ki célul. Általánosan elmondható, hogy az optimális hatást a lehető legkisebb mennyiségben, de viszonylag gyakran adagolt hatóanyaggal igyekeznek elérni, ezen protokollokkal kapcsolatban számos kérdés azonban jelenleg is nyitott.

Kutatásunkban a kezelés folyamatára intelligens szabályozási eljárásként tekintünk: az alkalmazott hatóanyag beadásának idejéről és a beadott dózis nagyságáról egy virtuális orvosként felfogható öntanuló eljárás, egy úgynevezett ágens hoz döntést. Az ágens által hozott döntések hatását egy matematikai modell segítségével szimulált páciensen vizsgáljuk, az ágens különféle akcióinak következtében a szimulált páciens állapota megváltozik, miközben a tumor reagál a kezelésre (vagy épp annak a hiányára).

Egy megerősítéses tanuláson alapuló eljárás alkalmas lehet az ágens (a virtuális orvos) és a környezet (a szimulált tumor mérhető jellemzői) közötti akción és jutalmazáson alapuló kölcsönhatás összefüggéseinek felderítésére. Jelen dolgozat célja olyan megerősítéses tanuláson alapuló megoldások vizsgálata, amelyeket jelenleg viszonylag ritkán alkalmaznak orvosi területeken, például gyógyszeres kezelések optimalizálására. Ilyen megoldások közé tartoznak az idősoros adatokkal dolgozó rekurrens megoldások, úgymint a *Deep Recurrent Q-Network (DRQN)*, de a hibrid *policy* optimalizáló változatok, például a *Proximal Policy Optimization (PPO, PPO2)* is ígéretes irányok. Hasonlóképp biztató a *Quantile Regression Deep Q-Network (QRDQN)* alkalmazása, amely regresszió alapú megközelítést biztosít, így az ágens döntését a diszkrét akció-modell helyett folytonos változó reprezentálja. Az alkalmazott kezelési modellt a korábbihoz képest a gyógyszeradagok felső határának realiztikusabb megválasztásával javítottuk.

1. fejezet

Motiváció és célkitűzés

A daganatos betegségek kezelésében a gyógyszeres kezelés – más néven kemoterápia – viszonylag hosszú múltra tekint vissza, mely alatt a hatékonysága jelentősen javult. Általánosan kijelenthető, hogy ma már olyan betegek életét is lényegesen meglehetősen hosszabbítani, vagy akár teljesen meggyógyítani, akinek néhány évtizeddel korábban sajnos még nem tudott volna érdemi javulást hozni semmilyen kezelés. A gyógyszeres terápia ennek ellenére is kihívásokkal küzd. Egyrészt a felhasznált hatóanyag vagy hatóanyagok az esetek egy részében a kezdeti hatékonyságukat elvesztik, mivel a szervezet sejtjei ellenállóvá (rezisztenssé) válnak a hatóanyagokkal szemben. Másrészt a napjainkban általánosan alkalmazott terápiák előre definiált kezelési protokoll szerint zajlanak, amelyek a betegek egyedi jellemzőit alig veszik figyelembe, és így az optimális hatást kiváltó gyógyszernél sokszor jóval többet juttatnak a beteg szervezetébe, amely jelentős mellékhatásokat okozhat a gyakran indokolatlan terhelés miatt.

A matematikai és számítógépes módszerekkel támogatott gyógyászat az utóbbi évek egyik gyorsan fejlődő területe. Ezen belül a modell-alapú terápiagenerálás egy ígéretes iránynak tűnik, mivel az általános protokolltól eltérően képes lehet a kezelt beteg egyedi jellemzőit is figyelembe véve döntést hozni a kezelés menetéről. A mesterséges intelligencia és a gépi tanulás alkalmazása ezen a területen is új eredményekhez vezethet azáltal, hogy a páciens és a kezelési folyamat mérhető jellemzői, valamint a kezelés sikeressége között összefüggésekre mutathat rá.

A megerősítéses tanulás (*reinforcement learning*), illetve annak "mély" változata (*deep reinforcement learning*) jelenleg nem egy általánosan használt módszer a gyógyszeres terápia kialakítása során, azonban a megközelítés alapgondolatából kiindulva működőképes lehet. A valódi kezelési protokollok kidolgozása során ugyanis a protokollt megtervező szakemberek (orvosok) lényegében egy olyan optimalizációs folyamatot hajtanak végre, melynek eredményeképp megalkotnak egy olyan "megoldást", amely a betegek többségének a lehető legnagyobb túlélést biztosítja, bizonyos korlátok megtartása mellett. A protokoll kidolgozása során természetesen mind előzetes információkat (pl. a gyógyszer várható hatását és mellékhatásait adott dózis mellett), mind a terápia alkalmazása közben begyűjtött új adatokat (pl. a gyógyszer valódi hatékonysága bizonyos típusú betegek esetén) figyelembe véve módosítják az ajánlott kezelést.

Ha ezt a protokoll alkotási eljárást megvizsgáljuk, sok hasonlóságot találunk közte és a megerősítéses tanulás alapgondolata között, épp ezért dolgozatunkban azt vizsgáljuk meg, hogyan lehetne virtuális orvosoknak virtuális pácienseken egy opti-

mális terápiát kidolgozni. Az optimális terápia definíciója nem nyilvánvaló: számos szempontból lehet egy terápia optimális, attól függően, milyen jellemzőire helyezzük a hangsúlyt. Nem tartható megfelelőnek például egy olyan megoldás, amely annak ellenére, hogy a tumorméretet a lehető legnagyobb mértékben le tudja csökkenteni, hatalmas mennyiségű dózist tart a kezelendő páciens szervezetében. Az optimális megoldás emiatt várhatóan azt fogja jelenteni, hogy az egyedi paraméterek függvényében egy olyan gyógyszerelési eljárás alapelveit találja meg a virtuális orvos, amelynél a tumorméret stagnál, vagy minél kisebb ütemben növekszik csak egy relatív alacsonyan tartott hatóanyagszint mellett, ezáltal a virtuális páciens élettartama a lehető leghosszabbra nyújtható. Amennyiben az eljárás működőképességét a szimulációs eredmények alátámasztják, úgy a módszertan akár állatkísérletes rendszerbe is átültethető.

2. fejezet

Irodalomkutatás

2.1. A megerősítéssel tanulás alapjai

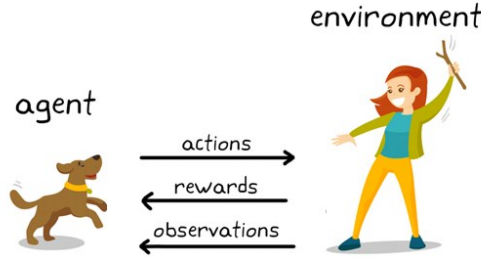
Ebben a fejezetben be fogjuk mutatni nagy vonalakban azt, hogy mi a megerősítéssel tanulás (*reinforcement learning*, röviden: RL) [1]. Először is az alapfogalmakat, koncepciókat szeretnénk vázolni, olyan kifejezéseket, szavakat, amiket a dolgozat hátralevő részében is használunk, olyanokat, amik az RL-ben használatosak.

Fontosnak tartjuk azt, hogy legelőször kettéválasszuk a megerősítéssel tanulást és a felügyelt tanulást. Felügyelt tanulásra egy konkrét példa, amikor képeket szeretnénk osztályozni. Nincsen konkrétan definiálva a cél, jövőbeli terv vagy bármi, amit el akarnánk érni, csak az, hogy minden kép egy előre meghatározott csoporthoz tartozzon. Ez nagyon absztrakt szinten olyan, mint egy sima függvény, amit ha meghívunk, akkor szétválogatja a képeket. Ezzel szemben a megerősítéssel tanulás sokkal inkább olyan, mint egy ciklus. Iterációkban lépkedünk előrébb és előrébb egy konkrétan definiált cél érdekében. Az algoritmusnak a célt mutatjuk, azt, hogy hova kell eljutnia. És nem mondjuk meg neki, hogy ezt hogyan csinálja, hogy minden egyes lépésben mit tegyen.

Az alábbiakban néhány alapfogalmat szeretnénk bevezetni, melyek segítik a probléma definiálását:

- **agent (ágens / ügynök):** az entitás, ami a célt el szeretné érni, ő hozza a döntéseket;
- **environment (környezet):** itt dolgozik az ágens a feladat megoldásán, ezzel lép interakcióba, ezt figyeli meg;
- **episode (epizód):** tanulási folyamat, a végén egy eredménnyel (sikerült vagy nem sikerült elérni a célt);
- **state (állapot):** a környezetet definiáló jellemzők aktuális értéke;
- **terminal state (végállapot):** az az állapot, amikor elértük a célt vagy egyáltalán nem sikerült és nem is fogjuk tudni elérni;
- **action (művelet):** az ágens által hozott döntés, amellyel beavatkozik a környezet állapotába;
- **reward (jutalom):** számszerű érték, amelyet az ügynök a döntésének eredményeképp kap;

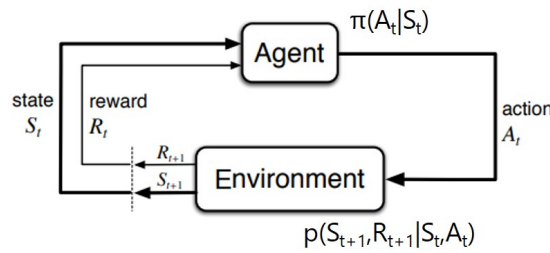
- **state spaces (állapottér):** összes lehetséges állapotot tartalmazó halmaz;
- **action spaces (művelettér):** összes lehetséges műveletet tartalmazó halmaz;
- **policy (irányelv):** az a szabályrendszer, amelyet az ügynök követ a feladat megoldása közben, ez alapján dönti el, hogy melyik műveletet végezze el.



2.1. ábra. Ahogy a megerősítéses tanulást el lehet képzelni [2].

Egy konkrét, nagyon egyszerű példán keresztül mutatnánk be az előbb definiált fogalmak egy részét. Tegyük fel, hogy van egy kiskutyánk (agent). A feladata az, hogy visszahozza a botot, amit eldobunk neki a réten (ez egy ideális episode). Az, hogy a bot hol található, az maga az environment. A kutyánk kap jutalom falatot (reward), ha visszahozza a botot, és nem kap, ha nem sikerül neki. A kiskutya választhat azok közül, hogy felveszi-e a botot vagy nem, vagy hogy melyik irányba indul el a botért (ez az action). Ha a bot ott van a közelében, akkor fogja meg és hozza vissza (ilyen policy-t, irányelvet követ).

Szakszerűen fogalmazva, a megerősítéses tanulás folyamata leírható egy Markov döntési folyamatként (*Markov Decision Process*, MDP) [3], ahol az alább látható 2 feltételes valószínűség szerepel az MDP folyamatban [4]. Ezt a folyamatot, hajtja végre és oldja meg az RL rendszerünk. Ezt a viselkedést a következő ábra segíti megérteni:



2.2. ábra. A megerősítéses tanulás valószínűségekkkel [5].

Az ügynökhöz és a környezethez is tartozik egy-egy állapot-átmeneti valószínűség. Itt az *Agent* választ egy t időpillanatban, egy π *policy* alapján egy adott *state*-ben egy *action*-t. Ez megadható mint $\pi(A_t|S_t)$, ahol A_t : *action* adott t időben és ahol az S_t : *state* adott t időpillanatban. Ez az ügynökhöz tartozó állapot átmeneti feltételes valószínűség. Ezzel szemben a környezet állapot átmeneti valószínűsége a következő: $p(S_{t+1}, R_{t+1}|S_t, A_t)$. Ez azt jelenti konkrétan, hogy a következő állapotba S_{t+1} lépve adott *reward*-ot kaphatunk (R_{t+1}), ha az adott *state*-ben (S_t) adott *action*-t (A_t) választjuk.

2.2. Tumornövekedés matematikai modelljei

A következőekben néhány általánosan használt tumornövekedési modellt mutatunk be. Kitérünk benne a modellek jellemzőire, a modellek által használt leírási módra, valamint említést teszünk még azok alkalmazhatóságáról.

2.2.1. Naiv megközelítések

A legelső, tumornövekedést leíró modellek több évtizedes múltra tekintenek vissza [6]. Az első megközelítések lényegében a populációdinamikában használatos megközelítéseket vették alapul, ezek közül a fontosabbakat R. K. Sachs cikke alapján foglaljuk össze [7]. A modelleket differenciálegyenletek segítségével formalizálták, így megfelelő paraméterek választásával predikció tehető a tumor viselkedésére (növekedésére).

A modell feltételezése szerint a kezdeti tumorméret exponenciális ütemben növekszik, így a növekedési dinamika a

$$\frac{dN}{dt} = \lambda N \quad (2.1)$$

differenciálegyenlettel írható le, ahol a λ paraméter szabályozza a növekedés ütemét, $N(t)$ pedig a szaporodóképes tumorsejtek mennyiségét adja meg.

Egy ennél pontosabb modell figyelembe veszi azt is, hogy az exponenciális növekedési ütem csak a kezdeti időszakban valós, sok tumor növekedése a mérettel arányosan lassul. Ez a megfigyelés az egyenlet alábbi általánosításával fogalmazható meg:

$$\frac{dN}{dt} = \left(\frac{\mu N}{\nu}\right) \left(1 - \left(\frac{N}{K}\right)^\nu\right), \quad (2.2)$$

ahol $K > 0$ egy méretkorlát, melyhez közeledve a növekedés lelassul, μ és ν pedig a növekedés dinamikáját befolyásolják. A valamilyen hatóanyag segítségével kiváltott sejtpusztulás leírása egy újabb tag bevezetésével megoldható.

$$\frac{dN}{dt} = -\alpha c(t)N + f(N), \quad (2.3)$$

ahol α a kemoterápiás hatóanyag erősségét mutatja, $c(t)$ pedig értelmezhető a hatóanyag időbeli koncentrációjaként. Az $f(N)$ függvény a hatóanyag nélküli növekedési dinamikát leíró tag, ez helyettesíthető például 2.2 jobb oldalával.

2.2.2. Angiogenezis hatását figyelembe vevő modellek

Az angiogenezis (érképződés) a tumorok növekedése szempontjából lényeges elem, hiszen a tumort alkotó sejtek tápanyag- és oxigénellátottságát a szövetekben lévő erek biztosítják, így ezek kialakulásának gátlása lassítja a tumor növekedését is.

Yang egy olyan modellt mutatott be, amelyben a tumor növekedésére az érképződés is hatással van [8]. A modell esetén megemlítsük a *vegő-angiogenezis* rész is, ahol a tumorsejtek a tömegvonzás törvényének (*mass action law*) megfelelően formálódnak meg. Ezt az úgynevezett *VEGF* termelődés és az *endothelialis*

kihajtás, szétterjedés követi. Feltételezve van továbbá, hogy a normál-, a tumor- és a hámsejtek is logisztikus növekedési ütemet mutatnak.

A modell az alábbi differenciálegyenlet-rendszerrel írható fel:

$$\frac{dC}{dt} = \alpha_1 C \left(1 - \frac{C}{k_1(E)} - \frac{\beta_1' T}{k_1(E)} \right) - \mu_1 C - C_m \delta_D(t) \quad (2.4)$$

$$\frac{dE}{dt} = \alpha_2 E \left(1 - \frac{E}{k_2(C)} \right) - \gamma ET - \mu_2 E \quad (2.5)$$

$$\frac{dT}{dt} = C_m \delta_D(t) + \alpha_3 AT \left(1 - \frac{T}{k_3} \right) + \alpha_3' T \left(1 - \frac{T}{k_1(E)} - \frac{\beta_2' C}{k_1(E)} \right) - \mu_3 T \quad (2.6)$$

$$\frac{dP}{dt} = \gamma ET - \delta P - \mu_4 P \quad (2.7)$$

$$\frac{dA}{dt} = \delta P + \epsilon T A \left(1 - \frac{A}{k_4} \right) - \mu_5 A \quad (2.8)$$

Az egyenletrendszerben a paraméterek az alábbiak:

- C, E, T, P, A : koncentrációjellemzők a normál- (C), a hám- (E), a tumor- (T), az elő-angiogenezis- (P) és angiogenezissejtek (A) vonatkozásában
- $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \epsilon$: a belső növekedési ráták a normál-, hám-, tumor- és angiogenezissejtek vonatkozásában
- $\mu_1, \mu_2, \mu_3, \mu_4, \mu_5$: a pusztulási ráták a normál-, a hám-, a tumor-, az elő-angiogenezis- és angiogenezissejtek vonatkozásában
- k_1, k_2, k_3, k_4 : a szállítási kapacitások a normál-, hám-, tumor- és angiogenezissejtek vonatkozásában
- δ : az elő-angiogenezis- és angiogenezis- sejtek közti átmeneti ráta
- γ : a hámsejtek kihajtási rátája
- β_1 : megmutatja, milyen mértékben akadályozzák a normál sejteket a tumor-sejtek
- β_2 : megmutatja, milyen mértékben akadályozzák a tumorsejteket a normál sejtek

Látható, hogy a modell számos paramétert tartalmaz, amely a modell flexibilitása szempontjából előnyös, ugyanakkor identifikálhatóságát rontja. Mindenféle sejttípus koncentrációjára, ezáltal következtethető térfogatára tesz utalást, amely az optimalizálás szempontjából is hatékonyan bizonyulhat. A mi esetünkben a nehézséget a megállapítható és használható paraméterek okozzák. A valóságban a tumor térfogata egyetlen változóként, nem pedig az azt alkotó különféle típusú sejtek szerint szétválasztva határozható csak meg, így szempontunkból ez a modell nem előnyös választás a probléma leírására.

2.2.3. Tumornövekedés és gyógyszerhatás modellezése

Drexler és munkatársai cikkükben egy minimál tumornövekedési modellt mutatnak be, amelyben a növekedési dinamikát viszonylag alacsony számú, relatív könnyen identifikálható paraméter, valamint az adagolt hatóanyag együttesen alakítják [9]. Magát a modellt rendszer- és irányításelmélet alapokra fektették, a szabályozás során ügyelve számos, kezeléssel kapcsolatosan felmerülő kérdésre. A modellt formális reakciólépéssel alapján származtatott differenciálegyenletek segítségével írják fel az alábbi formában:

$$\dot{x}_1 = (a - n) \cdot x_1 - b \cdot \frac{x_1 \cdot x_3}{ED_{50} + x_3} \quad (2.9)$$

$$\dot{x}_2 = n \cdot x_1 + b \cdot \frac{x_1 \cdot x_3}{ED_{50} + x_3} \quad (2.10)$$

$$\dot{x}_3 = -c \cdot \frac{x_3}{K_B + x_3} - b_k \cdot \frac{x_1 \cdot x_3}{ED_{50} + x_3} + u \quad (2.11)$$

x_1 : az osztódásra képes tumorsejtek össztérfogata köbmilliméterben

x_2 : a halott tumorsejtek össztérfogata köbmilliméterben

x_3 : a vérben lévő hatóanyag koncentrációja milligramm/kilogrammban

u : a befecskendezésre kerülő hatóanyag mennyiségét milligramm/kilogramm · nap mértékegységben leíró függvény

Az egyenletekben lévő paraméterek pedig az alábbiakat adják meg:

a : a tumor növekedési rátája

n : a tumor nekrotikus rátája (pusztulás mértéke)

b : a befecskendezett dózis hatása

c : a hatóanyag ürülési rátja (klírensz)

w : a nekrotikus szövet kimosódási rátája

ED_{50} : a hatóanyag effektív medián dózis értéke (elméleti érték, ilyen dózis mellett érne el a gyógyszer az 50%-os sejtpusztulást)

K_B : a hatóanyag Michaelis–Menten-konstansa

b_k : a hatóanyag felszívódási gyorsaságát jellemző érték

A rendszer kimenete (y) egy olyan, köbmilliméterben kifejezett térfogat, amely az élő és az elpusztult tumorszövet együttes térfogatát adja meg. A rendszer dinamikáját az

$$\dot{y} = a \cdot x_1 \quad (2.12)$$

egyenlettel lehet leírni, azaz a tumorméretbeli változást közvetlenül a tumornövekedési rátával (a) és a jelenleg "aktív", szaporodásra képes tumorsejtek térfogatával lehet jellemezni. Megállapítható, hogy a rendszer akkor van egyensúlyi állapotban, ha a szaporodásra képes tumorsejtek térfogata (x_1) 0 köbmilliméter, továbbá a vérben lévő dózis mennyisége (x_3) 0 milligramm/kilogrammmal egyezik meg, ha az első egyenletben szereplő $a - n$ érték negatív szám, ami akkor lehet, ha a több tumorsejt pusztul el, mint amennyi újonnan, szaporodóképes állapotban van jelen. Akkor viszont nyeregpontra van a rendszernek, ha ugyanezen kifejezés értéke pozitív, azaz ha több a szaporodóképes, új tumorsejt, mint az elpusztult. A rendszer irányíthatóságára vonatkozóan akkor tekinthető ez a rendszer irányíthatónak, azaz válik képessé

arra a rendszer, hogy a befecskendezett dózis hatására a tumorsejtek térfogata csökkenjen, ha az $a - n - b$ kifejezés negatív.

A Drexler-féle modellről megállapítható, hogy az általunk vizsgált probléma szempontjából sok tekintetben ideális. Optimalizálási feladatunk során csupán arra van ráhatásunk, hogy a kezelt páciens adott időpillanatban kap-e a hatóanyagból valamekkora dózist, avagy nem. Ennek hatása mérhetővé is válik, ugyanis a valóságban mérhető tumortérfogat az élő és halott szövet együtteseként áll elő. Ezen felül a modell viszonylag alacsony számú paramétert használ csak, amelyek megállapítása így könnyebb feladat, ezek alkalmas megválasztásával mégis jól leírható az egyes páciensek egyedi jellemzői.

A modell egy későbbi változata pontosabbá tette a gyógyszer farmakokinetikájának leírását egy kétkompartment-rendszer bevezetésével [10]. A modell ezen változatában a befecskendezett hatóanyag előbb a centrális kompartmentbe (vérbe) kerül, innen jut a perifériális kompartmentbe (a szervekbe, szövetekbe). A farmakokinetikát leíró kétkompartment-modell használatával a valóságban megfigyelt adatokra jobban illeszthető modellt kaptak, amely képes volt prediktálni az egyes dózisok hatására bekövetkező sejtpusztulást és növekedési dinamikát.

2.3. Vizsgált eljárások

Ebben a fejezetben a megerősítéses tanuláshoz kapcsolódó eljárásokat fogjuk taglalni. Azt, hogy konkrétan egy adott MDP-nél hogyan alkalmazható általánosan az adott megoldás. Ezeket az eljárásokat azonos, vagy közel azonos szempontok alapján igyekeztünk felkutatni, megvizsgálni és összehasonlítani. Ezek a szempontok a következők:

- Eljárás jellemzése:
 - Megemlítjük, hogy mit jelent a neve konkrétan és hogy miért kapta azt a nevet, amit;
 - Mutatunk egy egyszerű példát a működésére;
- Jellemezzük a működés elvi hátterét;
- Említést teszünk az egyes módszerek összehasonlított előnyeiről és hátrányairól.

Ebben a fejezetben egészen a megerősítéses tanulás alapjaitól indulunk ki, annak alap problémájának egyszerű megoldásától. Majd egészen odáig jutunk el hogy milyen új megközelítéseket, hibrid megoldásokat van lehetőségünk alkalmazni hogy minél pontosabb és jobb megoldásokat érjünk el.

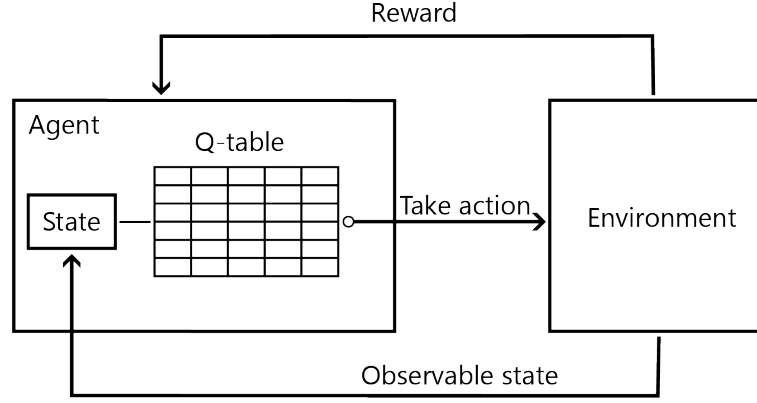
2.3.1. Q-tanulás

A megerősítéses tanítás egyik alap koncepciója a Q-tanulás (*Q-learning*) [11]. A *Q-learning* kifejezésben maga a Q a minőségre (*quality*) utal, lényege, hogy megtalálja adott feladat elvégzése során azt az optimális szabálysorozatot (*policy*), melyekkel az adott állapotokból (*S*, *state*) a megfelelő lépésekkel (*A*, *action*) az optimális megoldáshoz lehet jutni. Az optimális eredmény pedig minden esetben ugyanaz, miszerint találjuk meg a lehető legmagasabb összjutalmat adó megoldáslehetőséget. Mégpedig úgy, hogy minden lépésben, iteratívan haladunk tovább, és fejlesztjük az ágens (*agent*) viselkedését.

Az egyes *state* és *action* lehetőségeket egy úgynevezett Q-táblában (*Q-table*) tároljuk, minden állapotot és minden műveletet. Ebből is következik, hogy maga a Q-tábla $S \times A$ méretű. Ez persze problémaforrás is lehet, ha igen nagyok ezen elemszámok.

Alapesetben az *agent* egy kezdeti állapotból kiindulva, véges számú lépést téve eljut a különböző köztes állapotokon keresztül egy keresett végállapothoz (*terminal state*), amihez az előre definiált action-öket használja fel, a környezet (*environment*) jelenlétével és beavatkozásával. Minden action során az agent kiszámítja az adott állapot-átmenettel járó jutalmat (*reward*), majd a következő állapotra tér. Ezt ismételve jut el a már említett végső állapothoz, ahonnan tovább már nem vezet action, nem lehet tovább lépni.

Kérdésként merülhet fel, hogy az ágens egy adott állapotban mi alapján dönt a következő lépésről. Eleinte nagyon fontos tényező, hogy a modell maga "fedezze fel" (*exploration*), hogy milyen megoldásokhoz juthat a tanulás során. Ha már kellő mennyiségű lehetőség érintve volt, akkor céltudatosan, a már megismert tudás-halmaz minél mélyebb megismerése mellett tegyen kísérletet az optimális megoldás



2.3. ábra. A Q-Learning egyszerűsített architektúrája

megtalálásra (*exploitation*). A megoldások, illetve az optimális megoldás megtalálásához kétfajta megközelítés is használható. A *mohó* elvű megközelítésben minden esetben a lehetséges lépések közül azt választja, amely a legnagyobb várható jutalmat adja az egyes epizódok végére. A másik, *Epsilon-greedy* [12] eljárás során ϵ valószínűséggel (ahol $0 < \epsilon < 1$) véletlen lépést választ, és úgy lép tovább. A konvergencia esetenként javítható, ha a választott lépés nem teljesen véletlenszerű, hanem a Boltzmann-eloszlás szerinti [13].

Míg a mohó megközelítésnek előnye, hogy az aktuális állapotból mindig a legkedvezőbbnek tűnő állapotba léphet tovább, addig az *Epsilon-greedy* mellett szól, hogy így több, "felfedezetlen" irányok felé is haladhat a kimenetel, amely a globális eredményt tekintve akár jobbnak is ígérkezhet a lokálisan legjobbnak vélt action kiválasztásához képest. Az egyértelműen igazolt, hogy az optimalizáció során a mohó megközelítés helyett valamilyen felfedezést alkalmazni szükséges a tanítás fázisában a legjobb eredmény elérése érdekében [14, 15, 13].

A Q-tanulás fontos eleme, hogy az egyes lépések után frissíteni szükséges a Q-táblán belüli értékeket. A Q értékek frissítése [11] két egyenlettel írható fel:

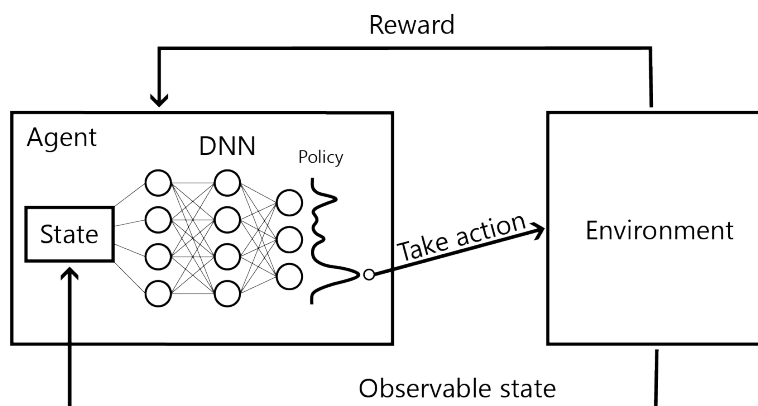
$$y = r + \gamma \cdot \max_a Q(S_{t+1}, a) \quad (2.13)$$

$$Q(s, a) = Q(s, a) + \alpha \cdot (y - Q(s, a)) \quad (2.14)$$

A két egyenletből az y -nal a relatív változást akarjuk megbecsülni, mi lehet annak várható értéke. Lényege, hogy kiválasztva az aktuálisan legjobbnak számító action-t minden lehetséges átmenetből, súlyozzuk, mellyel megadjuk, hogy a következőkben jelentkező jutalmak milyen súllyal (γ) számítsanak bele a végeredménybe. Ehhez hozzáadása kerül a már kiszámításra került jutalom. Az érték frissítésekor figyelembe veendő szempont, hogy frissítés előtt milyen Q érték állt itt korábban, hozzáadva egy olyan súlyozott értéket, ahol a lépésszám a paraméter (α), miszerint milyen gyakran frissítjük azt. Maguk a változtatható paraméterek lehetővé teszik a szabadabb választás jogát, hiszen ezek segítségével tudjuk szabályozni többek között azt, hogy mely állapotbeli jutalmak számítsanak bele leginkább az összjutalomba, a korai vagy a késői döntésekkel járó állapot-átmenetekéi. [1]

2.3.2. Deep Q-Network (DQN)

A napjainkban használt és elterjedt megerősítéssel tanuló eljárások mind DQN-re épülnek. A DQN [16, 17] rövidítése Deep Q-Network-nek. Ha egy mondatban kellene ismertetni, akkor erre az lenne a legegyszerűbb, ha a már megismert Q-Learning-re gondolnánk, ahol a Q táblát egy mély neurális hálóval helyettesítjük. Ez maga a DQN.



2.4. ábra. Az DQN egyszerűsített architektúrája

A DQN ugyanúgy, ahogy az összes Q-tanulás alapú módszer, ez is jutalom maximalizálásra törekszik, az epizód végére. Ezt ez a megoldás nem egy egyszerű lookup-table megoldásával teszi, hanem itt hívja segítségül a neurális hálót, ami adott state-re választ egy action-t.

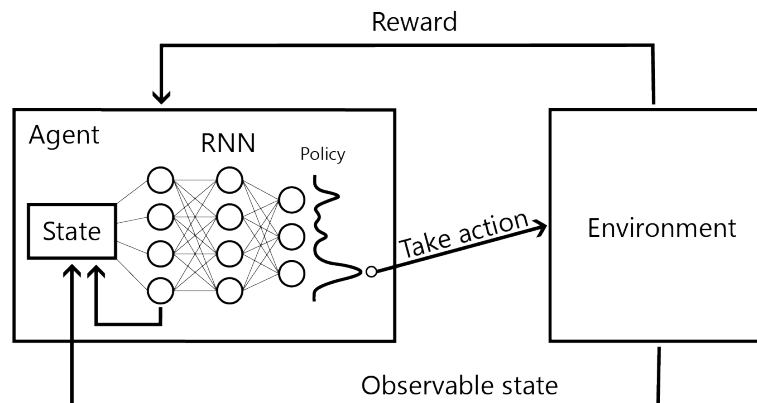
Ennek a megoldásnak az előnyei szemmel láthatóak a sima Q táblás megoldáshoz képest. Mivel itt egy DNN-el (Deep Neural Network-kel) dolgozunk, a háló képes arra hogy mintákat, összefüggéseket találjon state-ek és döntések (action-ok) között olyan komplexitással, amikre algoritmikailag szinte lehetetlen vagy csak nagyon nehezen lehetne. Ezen felül azért is egy robosztus megoldás a DQN mivel, ha végtelen vagy a végtelenhez közeli hatalmas állapottérrel (state space-el) rendelkezünk, akkor ezt a rengeteg értéket nem tudjuk eltárolni bármekkora adatszerkezetben. Előbb-utóbb ki kellene dobunk egy részét, tömöríteni kellene, ami mind-mind a pontosság rovására menne. Ezzel szemben a DQN ezeket az összefüggéseket a neurális háló súlyaiban tanulja és őrzi meg.

2.3.3. Deep Recurrent Q-Network (DRQN)

Mielőtt belekezdenénk abba, hogy mi is az a DRQN (Deep Recurrent Q-Network) [18, 19], előtte fontos megemlíteni azt, hogy a Q-Learning és a DQN megoldások mindegyike az MDP-hez (Markov Decision Process) készült megoldások. Ezzel szemben a DRQN POMDP-khez (Partially Observable Markov Decision Process) [20] készült megoldás főként.

Egy gyakorlati példán keresztül szeretnénk érzékeltetni a fennálló problémát és az arra felvázolt megoldást. A feladat az, hogy 2 gyerek labdázik egy videón, és a videó képeiből egy DQN ágens szeretné eldönteni azt, hogy ki ütötte kinek a labdát. A probléma az, hogy az ágensünk pusztán egy kép megfigyelésével (ez maga az environment) nem tud döntést hozni, mivel a labda mozgását (egyszerűsítve) nem lehet detektálni egyetlen képen. Ez a probléma egy POMDP feladat. Erre a megoldás

a DRQN, ahol a sima neurális hálóval szemben visszacsatolt, rekurrens, memóriával rendelkező neurális hálót használunk, ami képes elraktározni a környezetünkről készített megfigyeléseket.



2.5. ábra. A DRQN egyszerűsített architektúrája

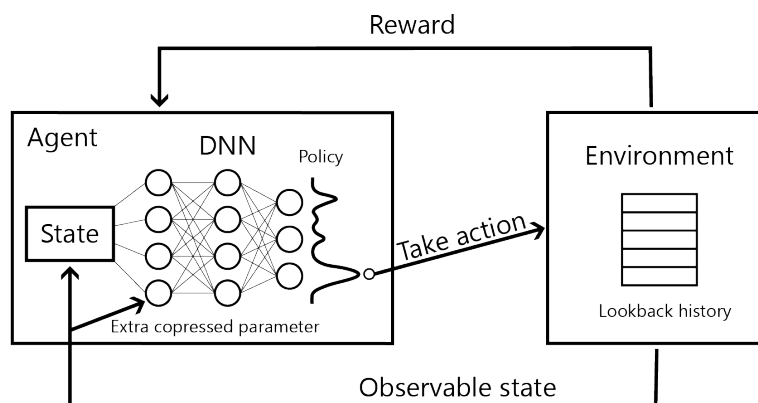
A DRQN segítségével egy adott memória ablakon belül az RNN-nel (Recurrent Neural Network) [21, 22] felvértezett ágensünk képes még komplexebb és pontosabb döntéseket hozni. Ilyen rekurrens hálózatoknak a használata nem elterjedt és szokványos eljárás a megerősítéses tanulás alkalmazásánál, így rengeteg kérdés és kiaknázatlan terület van, ahol ezt be lehetne vetni. Maga a háló rekurrens mivoltjának az implementációja, kialakítása nem nehéz, azonban a környezet, ágens felkészítése, hogy képes legyen jól kezelni ezt az újfajta megközelítést, kíván némi előkészületet és kihívásokkal jár.

2.3.4. Csúszó ablakos különbözeteken alapuló DQN

A csúszó ablakos különbözeteken alapuló DQN gyakorlatilag fél úton van a már megismert DRQN és a DQN között (de talán a DQN-hez egy kicsit közelebb áll ahogy az a nevéből is kitűnik). A legtöbb való világbeli probléma nem MDP, hanem POMDP, azonban nem mindig van lehetőségünk arra, hogy DRQN-t tudjunk használni, erőforrás vagy architektúráis indokok miatt. Erre egy kiváló hibrid megoldás a csúszó ablakos különbözeteken alapuló DQN.

Hogy ez mit jelent, egyet hátrébb lépve vessünk egy pillantást a DQN mivoltjára. A DQN-ben a neurális háló az environment-et megfigyelve (a különböző mérhető, látható paramétereit) hoz döntéseket, egy adott pillanatban, egy adott "pillanat-képet látva". A DQN-nek nincs információja a múltból gyakorlatilag. A csúszó ablakos különbözeteken alapuló DQN-nél a bemeneti, megfigyelhető paraméter lista van kiegészítve egy speciális history-val. Alapértelmezetten ez a history egy 2D struktúra, ami az utolsó X db megfigyelt tulajdonságot tartalmazza a környezetünkről. Ez azért speciális, mivel nem maga a lista kerül felhasználásra a DQN-ben visszacsatolva, hanem ennek egy redukált változata 1D-re. Ez azt jelenti, hogy a környezetről megfigyelt tulajdonságok először egy listába kerülnek majd ezt redukcióval, egy statisztikai eljárással 1D-ra zsugorítjuk. Ez az amit vissza adunk egy kumulált paraméterként a DQN ügynökünknek. Itt olyan aggregációs megoldásokra kell gondolni, mint például egy szimpla összegzés, átlag stb. Ez a keletkezett 1D új paraméter rendelkezni fog a visszatekinthető utolsó X db megfigyelt paraméter ku-

mulált tulajdonságaival. A gyakorlatban ennek a listának a napra készen tartását, redukálását a környezet végzi el.



2.6. ábra. A Csúszó ablakos különbözeteken alapuló DQN egyszerűsített architektúrája

Ez azért rendkívül hatásos mivel az ügynökünk és a hálónk az utolsó X darab döntésünkből fakadó paraméterváltozást is fogja érzékelni emiatt az új bevetetett kumulált paraméternek köszönhetően, egy új döntés meghozatalánál. Így nem csak egy megfigyelt pillanattal rendelkezik, hanem a múltbéliek egy részével is, emiatt később pontosabb döntéseket tud majd hozni a rendszer. Ami miatt mégis különlegesebb, mint a DRQN, hogy a DRQN-nek olykor nagy számítási igénye van és ezzel szemben a csúszó ablakos különbözeteken alapuló DQN egy változóban látja a teljes múltat, nem pedig a háló visszacsatolásában. Ennek a megoldásnak nem sokkal nagyobb a számítás igénye mint a sima DQN-nek mivel gyakorlatilag 1 plusz paraméter kerül bevezetésre, a history számításán és redukcióján kívül.

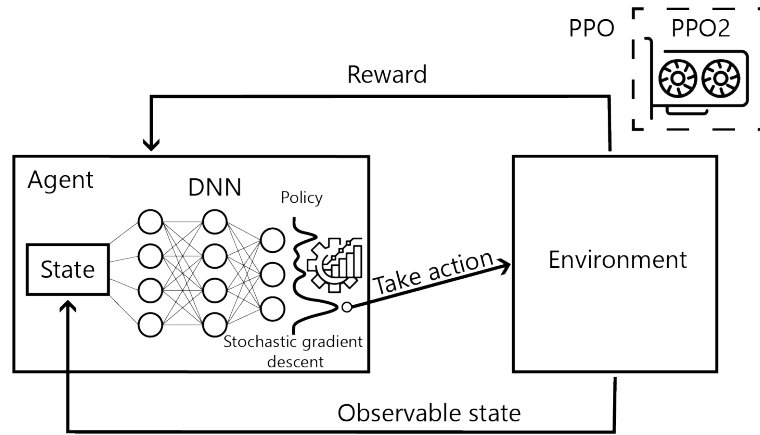
2.3.5. Proximal Policy Optimization Deep Q-Network (PPO-DQN)

A *Proximal Policy Optimalization* megoldás a policy optimalizálás család egyik tagja. Gyakorlatilag több epoch-nyi gradient descent-et használ minden egyes irányelv (*policy*) megválasztásához. Erre azért lehet szükség mert *DQN*-nél könnyen lehet hogy akkorákat ugrunk a policy megválasztás tekintetében hogy átugorjunk a megoldást és az idáig lévő pontosságunkat ezzel teljesen elrontjuk. Gyakorlatilag a gradient descent előnyét használja ki a *policy* megválasztásánál minden lépésben. Ahol a célfüggvény lapos ott hagyja a jelenlegi irányelvet (*policy*-t) eltávolodni és engedi az agent-et hogy drasztikusan más irányelvet válasszon, ahol meg ezzel szemben meredek a függvény ott megpróbálja a jelenlegi *policy*-hez a legközelebb tartani az új választandó policy-t. Ez a megoldás rendkívül robosztus és rövidebb idő alatt tud megoldást biztosítani a megerősítéses tanulással megoldandó feladatok számára.

A *PPO* megalkotói hamar rájöttek arra, hogy ezt a megoldást könnyedén lehetne még gyorsabbá tenni. Így született meg ennek a verzióknak az implementációja *GPU*-ra az OpenAI által. Ez a verzió vektorizált formában várja a környezeti változókat a sima *PPO* verzióval szemben az elosztottság és multiprocessing végett. Mindez még nagyobb gyorsaság elérésére képes, tesztek szerint háromszoros [23] sebességnövekedést képes produkálni.

Fontos megemlíteni, hogy ezeknél a megoldásoknál nincs a múltba való visszatekintés, így tendenciákat, megoldás irányokat messziről nem képes meglátni és felis-

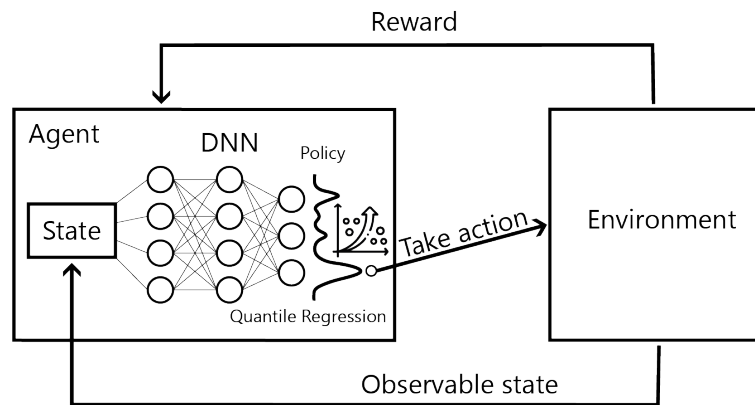
merni.



2.7. ábra. A PPO-DQN és PPO2-DQN egyszerűsített architektúrája egy képen

2.3.6. Quantile Regression Deep Q-Network (QR-DQN)

Az eddigi megoldások mindegyike közös volt abban egy pillanat vagy egy idősor alapján hoz egy döntést [24]. Az idősoros megoldásokban benne van az a közvetett lehetőség, hogy adott időablakon belül „észre tudjon venni” az *agent* egy tendenciát, mintát és az alapján hozzon egy döntést. Azonban mindez csak lehetőség, amit vagy sikerül elsajátítania, vagy nem. Nem beszélve arról is, hogy az időablak az csúszik vagy csúszhat és így csak egy bizonyos szintig képes visszatekinteni az *agent*-ünk. Erre a problémára ad megoldást a *QR-DQN*. Ez a megoldás egy eloszlás alapján dolgozik, úgy, hogy megnézi azt, hogy milyen megfigyelt értékekre milyen megoldásokat adtunk és ez mivel járt. Erre az eloszlásra illeszt egy olyan görbét, ahol az úgynevezett *Wasserstein-távolságokat* igyekszik minimalizálni. Ezáltal képes ez a megoldás a tendenciákra fókuszálva meghozni egy-egy döntést használva a regresszió erejét.



2.8. ábra. A QR-DQN egyszerűsített architektúrája

2.4. Vizsgált jutalomfüggvények

Jutalomfüggvények tekintetében ahogy feljebb ismertettük ez az a része a megoldásunknak, ami osztályozza a modellünk döntéseit jóság alapján. Mindezt részletesen

kifejtve a 2021-es „*Tumorkezelési eljárás optimalizálása mély megerősítéses tanulás alapon*”[25] című TDK pályamunkánkban ismertettük, és használtuk fel az ebből származó ismereteket jelen kutatás során.

2.5. Konklúzió

Ahogy az előző kutatásunkból is kiderült[25], modellek tekintetében a Drexler modell az, ami a leginkább megfelel ezen kutatás megvalósítására. Több szempontot is figyelembe vettünk ezen előző kutatásban, amit itt nem tervezünk részletezni. Emiatt ettől a modelltől nem tervezünk eltérni jelen projekt tekintetében sem.

Megerősítéses tanulás eljárásai tekintetében előzőleg[25] felhasználtuk a DQN-t, és a Csúszóablakos különbözetű DQN-t. Ezen kutatásban ezzel szemben a következőket tervezzük kipróbálni:

- Deep Recurrent Q-Network (DRQN)
- Proximal Policy Optimization 2 Deep Q-Network (PPO2-DQN)
- Quantile Regression Deep Q-Network (QR-DQN)

Előző kutatásunkban éppen, hogy érintettük a DRQN-t így tesztelni, implementálni most tervezzük. A standard PPO-t azért nem tervezzük használni mivel nem GPU optimalizált így rendkívül lassú. A QRDQN lehet a megoldásunk kulcsa mivel a tendenciákat figyelembevéve pontosabb, összetettebb kezelés érhető el.

A jutalomfüggvények felől az előző kutatásban[25] 4-et implementáltunk a 9 kuttatottból. A konkrét szobán forgó függvények:

- Yazdjerdi-féle jutalomfüggvény [26]
- Hassani-féle jutalomfüggvény [27]
- Zhao-féle jutalomfüggvény [28]
- Yauney-féle jutalomfüggvény [29]

Ezeket tervezzük itt is felhasználni alapvetőleg. Ha a megvalósítás során találunk, fejlesztünk empirikus úton új függvényt akkor az is megjelenhet az eredményben.

3. fejezet

Módszertan

A következőekben összefoglaljuk, hogy kutatásunk pontosan milyen részekből tevődik össze, kitérve természetesen magára az implementációra is. Említésre kerülnek az alkalmazott, irodalomkutatásból következtetett eljárások, továbbá konkrét forráskóddal demonstráljuk a rendszer jelenlegi állapotát.

3.1. Környezet bemutatása

Ha megerősítéses tanulásról van szó és annak környezetének kialakításáról, akkor az iparban a legelterjedtebb megoldások közé tartozik az OpenAI által fejlesztett úgynevezett *Gym* csomag. Ez azért örvend akkora népszerűségnek, mivel ez egy egységbe zárt megoldást ad arra, hogy saját probléma leíró modellel, egy saját környezetet legyünk képesek implementálni, egyedi megoldásokkal. Mindezt úgy teszi, hogy kívülről amikor az úgynevezett *agent* felhasználja ezt a környezetet akkor azt előre az *OpenAI* által meghatározott *interface*-n, metódusokon keresztül használja fel. Ez azért rendkívül hasznos és robosztus, mivel tetszőleges agent csomagokkal lehet ezt összepárosítani, tetszőleges megerősítéses tanulás eljárást képesek vagyunk erre az egységesített környezetre alkalmazni, szinte függőségek nélkül. (Egyedül annak a kritériumnak kell megfelelni, ahogy említettük, hogy valósítsák meg a *Gym* által definiált metódusokat, paramétereket)

A mélytanulási megoldások tekintetében az elsők között kerül szóba a *python* programozási nyelv is. Ez a nyílt forráskódjának, dokumentációjának, egyszerűségének köszönhető. A megerősítéses tanulásban sincs ez másképp. Az előbb említett *Gym* implementációja is alpból *pythonban* készült, így adta magát az hogy olyan csomagokat keressünk, olyan mély megerősítéses tanulás eljárásokat, amik ebbe az architektúrába illeszkednek. Ezen felül a mély tanulóhoz *python* tekintetében kettő típusú, igen elterjedt csomag található. Ezek az úgynevezett *pytorch* és *keras* alapú megközelítések. Mindkettőnek megvan az előnye és hátránya is. Ez a két irány a megerősítéses tanulásnál is megjelenik a következő két csomagnál amiket használunk:

- *keras* alapú: *keras-rl*
- *pytorch* alapú: *stable-baselines*

Az alábbi állításokat tudjuk tenni, amikor a nevezett két csomagról van szó. Megemlítendő, hogy a *keras-rl* csomag és annak verziói talán egy fokkal magasabb szintű

megoldásokat nyújtanak a megerősítéses tanulás tekintetében. Ismereteink szerint a magasabb szint limitációval, limitációkkal jár, és itt sincs másképp. Többek között a komplexebb, rekurrens változatú hálózati felépítéseket megerősítéses tanulás tekintetében a mi problémánkra nem kezeli megfelelően jól, emiatt ilyen típusú hálózatokat nem lehet alkalmazni ezen csomagon keresztül.

Emiatt is indultunk el a *pytorch* alapú *stable-baselines* irányába, ahol ugyan komplexebb implementációkra, kiegészítésekre volt szükség, viszont rengeteg olyan *agent* megközelítést tartalmazott, amit az előbbi *keras* alapú nem.

Így jutottunk arra, hogy mindkettő megoldásnak megvan a maga szerepe és helye a maguk megfelelő érveivel. Amiatt, hogy a kutatást minél szélesebb körben el tudjuk végezni, mindkettő kipróbálásra, alkalmazásra és a tanulási folyamat végén tesztelésre is kerül.

3.1.1. Használt modellek bemutatása

Környezetünk egy igen fontos része egy annak megfelelő modell is, amelyen a megerősítéses tanulást el tudjuk végezni. Ahogy az irodalomkutatásunk végén megfogalmaztuk, elsődlegesen a legfrissebbnek tartott Drexler modellt vennénk alapul. Ehhez kétféle modellimplementációt is figyelembe vettünk. Az egyik a tényleges *DrexlerModel* nevet kapta, míg a másik a *DrexlerTwoCompartmentModel*. Mivel hasznos tényező lehet az, ha több adattal tudunk tanítási folyamatot végrehajtani, több minden között lehet összefüggéseket megfigyelni és következtetni, így leginkább az utóbbi modell került felhasználásra. A *DrexlerTwoCompartmentModel* esetén nagyon fontos pluszinformáció ugyanis, hogy rendelkezik egy *history*-val, mégpedig az előzőekben mért és azonosított értékekre vonatkozóan. Egy ilyen *history* szerkezetben eltárolásra kerül ugyanis az előző tumortérfogatokból, dózisszintekből és döntésekből (valamint az azokkal járó beadott dózis mennyiségéből) adott elemszámú, hogy azok figyelembe vételével lehessen a további döntéseket is meghozni. Ezek egy-egy listában kerülnek eltárolásra, ugyanakkor megemlítendő tényező a méret kapcsán, hogy ezeket az adatokat folyamatosan frissíteni szükséges, egy megfelelően választott döntés mentén (amelyből a legalapvetőbb a legrégebbi adat elvetése). *Twocompartment* lévén a modell esetén feltételezett, hogy a tumorsejt 2 egymástól elválasztható "rekesszel" rendelkezik. Differenciálegyenleteinek leírása során figyelembe is vett tényező, ugyanis állapotváltozói közé tartozik

- az élő tumorsejtek össztérfogata köbmilliméterben;
- az elhalt tumorsejtek össztérfogata köbmilliméterben;
- a belső, centrális kompartmentben mért gyógyszerkoncentráció milligramm/-kilogrammban;
- a külső, periférikus kompartmentben mért gyógyszerkoncentráció milligramm/-kilogrammban,

ahol magát a tumorméretet, a tumor térfogatát az első két állapotváltozó értékének összegeként lehet meghatározni. Elérendő cél tehát az, hogy ezen modell, az ehhez társított környezet felhasználásával, valamint egy megfelelő jutalomfüggvény megtalálásával egy olyan megoldás kerüljön meghatározásra, amely a lehető legtöbb szempontból optimálisnak tekinthető.

3.1.2. Önálló kutatói munka határai

Fontos kiemelni azt, hogy ezt a kutatást, tanulmányt egy kutató csapat részeként alkottuk meg. Jelen dolgozatban saját munkánk a megerősítéses tanítás további eljárásainak áttekintése, vizsgálata, implementációja, az eredmények értékelése. Ide sorolható még az adagolt dózis maximumának bevezetése, implementációja, továbbá a környezet kibővítése és fejlesztése úgy, hogy képes legyen a megoldás több *action*-t kezelni.

3.2. Implementáció nagy vonalakban

Az alábbi fejezetben az implementáció lesz tömören bemutatva. A *Drexler Environment* rövid bemutatója után egy pár szóban fogjuk taglalni az implementált eljárásokat, majd egy összefoglaló képet szeretnénk adni egy lépést hátrább lépve arról hogy mindezt hogy és hol használjuk fel.

3.2.1. Környezet implementációja

Ha megnézzük a Diszkrét Drexler Environment kódját, akkor jól látszik, hogy maga a környezet implementálja az *OpenAI Gym Env* ösosztályából származó metódusokat, amivel létrejön az a környezet, amit a megerősítéses tanulás keretrendszerünk elvár. A környezetet különböző paraméterekkel láthatjuk el a konstruktorában, amik a következőket jelentik:

- *action_num*: hány fajta cselekedte van az ügynöknek, esetünkben 5, vagy nem adunk gyógyszert a páciensnek, vagy kettesével növekedve (8-ig) adunk (azaz a lehetséges *action*-ök: {0, 2, 4, 6, 8})
- *dose_unit*: amennyi hatóanyag egy egységnyi dózisban van, ha az **action** az, hogy adjunk gyógyszert a páciensnek
- *max_episode_length*: meddig tartson egy kezelés, epizód
- *model*: milyen típusú Drexler modellt használjunk
- *reward*: milyen típusú jutalomfüggvényt használjunk

```

class DiscreteDrexlerEnv(Env):
    def __init__(self, action_num=5, dose_unit=8, max_episode_length=100,
                  model="twocompartment", reward="default"):
        super(DiscreteDrexlerEnv, self).__init__()
        if model not in ["twocompartment", "original"] and
            not isinstance(model, Model):
            raise ValueError("model")
        if reward not in ["default"] and not isinstance(reward, Reward):
            raise ValueError("reward")
        self.action_num = action_num
        self.dose_unit = dose_unit / (action_num-1)
        if model == "twocompartment":
            self.dm = DrexlerTwoCompartmentModel(max_episode_length)
        elif model == "original":
            self.dm = DrexlerModel(max_episode_length)
        else:
            self.dm = model
        if reward == "default":
            self.reward = LinearReward()
        else:
            self.reward = reward
        self.observation_space =
            Box(np.array([0, 0, 0]),
                np.array(
                    [self.dm.volume_lim,
                     self.dm.dose_lim,
                     self.dm.normalised_givendosage]),
                dtype=np.float32)
        self.action_space = Discrete(self.action_num)
        self.reset()

    def step(self, action):
        self.dm.next(dose=action * self.dose_unit)
        return np.array(self.observations),
            self.reward_function(action),
            self.dm.done, {}

```

1. Listing. A *Discrete Drexler Environment* osztálydefiníciója és konstruktora

Ahogy az implementációból is látszik, ezek a bemeneti paraméterek változhatnak, és ezzel a megvalósítással tetszőleges variációkat hozhatunk létre a Drexler környezetből. Ez rendkívül rugalmassá teszi a rendszerünket a sok fajta kísérlet kivitelezésére. A példakódból jól látszik, hogy egy ősosztályon keresztül bármilyen tetszőleges jutalomfüggvény osztályt is meg lehet adni. Amit kiemelnénk még itt az *action_num* és a *dose_unit* kapcsolata. A *dose_unit* és az *action_num* hányadosa lesz az új *dose_unit*, amit később az *Env* ősosztály által definiálandó **step** függvényben használunk fel. Itt az alapján szorozzuk be az alap dózis egységet, hogy melyik

action-t választottuk (0-4-ig). (Természetesen más, itt fel nem tüntetett metódusok is implementálva vannak az *Env* őssztály miatt)

3.2.2. Dosemax és terminálás implementációja

A **DiscreteDrexlerEnv** **step** függvényének első sorában kerül meghívásra a Drexler modell **next** függvénye, ami a következőket tartalmazza:

```
def next(self, dose):
    solution = odeint(self.ode_model, self.x,
                      np.linspace(0, self.t_step, 100), args=(dose,))
    self.x = solution[-1]
    self.life += 1
    if (self.tumor_volume > self.vol_lim) or (self.drug_level > self.dose_lim) or
        (self.dose_sum + dose >= self.dose_max) or
        (self.life >= self.max_episode_length):
        self.done = True
    self.update_history(self.tumor_volume, self.drug_level, dose)
```

2. Listing. A *Drexler* modell *next* függvénye

A fentiekben jól látszóik, hogy a differenciálegyenlet megoldása és az élettídnövelése után minden lépésben, ha kell, egy összetett feltétel vizsgálata alapján termináljuk a kezelt páciens. Itt játszik szerepet a *dose_max* is. Ha elérünk egy beadott dózis mennyiséget, akkor szintén termináljuk a páciens, ugyanúgy, ha a tumor már egy limiten túl növekedne vagy pedig az ellenanyag koncentrációja a vérben is meghaladna egy szintet.

3.2.3. Eljárások implementációja jellegük szerint

Az implementációk jellegeinek tekintetében jól látszanak a különbségek és hasonlóságok is a *torch* alapú eljárások és a *keras* alapúak között. Ezek az implementációk a következőkből épülnek fel:

- környezet példányosítás
- *agent* létrehozása, felparaméterezése
- tanítás
- tesztelés, adatgyűjtés és adat vizualizáció

```

# Creating the environment
env = DiscreteDrexlerEnv(model="twocompartment",
                        reward=LinearLongLivingPreferredReward())

# Creating the NN for the agent
model = Sequential()
model.add(Flatten(input_shape=(1,) + env.observation_space.shape))
model.add(Dense(4, activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(nb_actions, activation='linear'))

# Creating the agent based on the NN
memory = SequentialMemory(limit=50, window_length=1)
policy = BoltzmannQPolicy()
dqn = DQNAgent(model=model, nb_actions=nb_actions, memory=memory,
               nb_steps_warmup=10, target_model_update=1e-2, policy=policy)

# Construction of the system
dqn.compile(Adam(lr=1e-3), metrics=['mae'])

# Teaching
dqn.fit(env, nb_steps=5000, visualize=False, verbose=2)

```

3. Listing. A Keras alapú eljárások struktúrája, egyszerű DQN

A felfebb levő *keras* alapú implementációban jól látszik hogy a környezet példányosítása után gyakorlatilag kedvünkre finomíthatjuk, saját magunkra szabhatjuk a hálózatot, paramétereit és az *agent*-et is. Ez az implementáció finomhangolt megoldást tud nyújtani, ami azonban sokszor felesleges is. Ezzel szemben a következő példában láthatjuk a három sorból megvalósított **torch** megoldást:

```

# Creating the environment
env = DiscreteDrexlerEnv(model="twocompartment",
                        reward=LinearLongLivingPreferredReward())

# Creating the agent with built-in NN
model = PPO2('MlpLstmPolicy', env, verbose=1, nminibatches=1)

# Teaching
model.learn(total_timesteps=10000)

```

4. Listing. A Torch alapú eljárások struktúrája, PPO2

Itt az *agent*-et magába foglalja a hálózatot már, amivel nekünk nem kell foglalkoznunk. Az érdekesség az az, hogy a model példányosításánál rekurrens, illetve összetettebb *policy*-kre van lehetőség, míg a *keras* alapú megoldás ilyen szempontból szűkös. Nem beszélve arról hogy sok fajta *stable-baselines* modellt támogat a *torch*, rengeteg olyat is, amit a *keras* nem. Ilyen a példában szereplő *PPO2* is.

Megfigyelhető az is, hogy az idáig taglalt *Gym* keretrendszer miatt a környezet példányosítása megoldástípustól függetlenül ugyanaz. Ez az, ami az egyike annak, ami a robosztusságot adja ennek a projektnek.

```
# Testing & data collection with Keras
def test_for_given_patient(idx):
    env = DiscreteDrexlerEnv(
        model=DrexlerTwoCompartmentModel(
            accessible_params=[idx],
            random_params=False,
            randomize_init_tumor_volume=False,
            max_episode_length=100,
            dose_max=50),
        reward=LinearLongLivingPreferredReward())

    cb = LogCallback()
    dqn.test(env, nb_episodes=1, visualize=False, callbacks=[cb])

    vol = [a[0] for a in cb.observations[0]]
    drg = [a[1] for a in cb.observations[0]]
    inj = [a for a in cb.actions[0]]

    plotter = Plotter(inj, vol, drg, env.dose_unit)
    plotter.plot_multiple_levels(pick_from_color_presets=True,
                                plot_title="DQN {}.patient".format(idx),
                                save_plot_result=True)
```

5. Listing. A *Keras* alapú tesztelés és adatgyűjtés

Tesztelés és adatvizualizáció tekintetében kijelenthető az, hogy a *keras* alapú megoldás szofisztikáltabb, mivel egy *callback*-en keresztül működik, így tesztelés közben nincs szükség külön adatmentő sorokra, melyekből később a vizualizáció készül.

```

# Testing & data collection with Torch
def test_one_episode(save_to_history=False):
    vol = list()
    drg = list()
    inj = list()

    episode_reward_list = list()
    steps = 0

    obs = env.reset()
    for i in range(one_episode_length):
        action, _states = model.predict([obs])
        obs, rewards, dones, info = env.step(action)

        vol.append(obs[0])
        drg.append(obs[1])
        inj.append(action)

        episode_reward_list.append(rewards)
        if dones:
            steps = i
            break

    if save_to_history:
        hist['episode_reward'].append(sum(episode_reward_list))
        hist['nb_steps'].append(steps)

    return vol, drg, inj

def run_for_given_patient(idx):
    env = DiscreteDrexlerEnv(
        model=DrexlerTwoCompartmentModel(
            accessible_params=[idx],
            random_params=False,
            randomize_init_tumor_volume=False,
            max_episode_length=100,
            dose_max=50),
        reward=LinearLongLivingPreferredReward())

    vol, drg, inj = test_one_episode(save_to_history=False)

    plotter = Plotter(inj, vol, drg, env.dose_unit)
    plotter.plot_multiple_levels(pick_from_color_presets=True,
                                plot_title="PPO {}.patient".format(idx),
                                save_plot_result=True)

```

6. Listing. A Keras alapú tesztelés és adatgyűjtés

A *torch* esetében sajnos azoban külön tesztelő és adatmentő metódust kell implementálnunk, aminek az eredményeit a szintén általunk alkotott *Plotter*-nak adunk át. Ez is azt mutatja, hogy a *torch*-nál és a *keras*-nál is más-más helyen csoportosul a logika. Az is szemre vehető, hogy egy jól standardizált *Plotter* segítségével mindegy is az, hogy milyen megoldásból származnak az adataink, ha kellően egységbe van zárva a logika (ahogy itt van) úgy könnyedén, ugyanolyan struktúrájú diagrammokat kapunk a végén.

3.3. Végrehajtandó kísérletek

Az implementációkból levonva a konklúziót meghatároztuk azt hogy milyen kísérletek legyenek végrehajtva. Model és környezet tekintetében egyértelmű a válasz mégpedig az hogy a *Drexler*-féle *Twocompartment* modelt felhasználva és az ebből alkotott úgynevezett *DiscreteDrexlerEnv*-et használjuk fel. Jutalomfüggvény tekintetében egy általunk és tapasztalataink alapján választottnál maradtunk, a *LinearLongLivingPreferredReward*-nál. Megerősítéses tanulás tekintetében a fókuszban a *DRQN*, *QRDQN*, *PPO* és a *PPO2* lesz. Mindezt fogjuk összehasonlítani a standard DQN megoldással. A standard *PPO*-t azért nem tervezzük használni mivel rendkívül lassúnak bizonyult az előzetes kísérleteink alapján. Mindegyik eljáráshoz 3 vagy több teszteredmény kerül rögzítésre, így összesen több mint 12 eredmény összehasonlítása történik meg a kiértékelés során.

4. fejezet

Eredmények

Az eredmények bemutatása során, a tesztek végrehajtását követően azt a jutalomfüggvényt választottuk ki szemléletes eredmények bemutatására, melyről megállapításra került, hogy eredményeit tekintve átlagban a legjobban teljesültek. Noha ez nem feltétlenül jár azzal, hogy a dedikált tesztalanyok esetén is megfelelő ábrát kapjunk. Ugyanakkor az mindenféleképpen elmondható, hogy a nagy átlagot tekintve az azon jutalomfüggvényhez nagyobb valószínűséggel, biztosabban kapcsolódik a kutatás szempontjából megfelelő kimenet. Eredmények leírásához az úgynevezett *LongLivingLinear* jutalomfüggvényt használtuk fel, mely a jutalmat a túlélési idő függvényében lineárisan adja, azaz leginkább a minél hosszabb túlélési időt helyezi előtérbe mint jutalmazási szempont. Az alábbiakban ismertetjük az eredmények leírásánál használt elnevezéseket:

- *LLL*: utalás a *Long Living Linear*-féle jutalomfüggvényre
- *TC*: utalás a Drexler-féle *Twocompartment* környezetre
- *PPO*: utalás a PPO-féle módszer nevére
- *PPO2*: utalás a PPO2-féle módszer nevére
- *QRDQN*: utalás a QRDQN-féle módszer nevére
- *DQN*: utalás a DQN-féle módszer nevére
- *ARDQN*: utalás a ARDQN-féle módszer nevére

Mivel a kísérletben egységenként több kísérlet is fog megtörténni, az alábbi jelölésmódot vezetjük be az eredmények későbbi pontos értékeléséhez:

$$< \textit{környezetNeve} > _ < \textit{módszerNeve} > _ < \textit{jutalomFüggvényNeve} > _ < \textit{kísérletSzáma} >$$

Így például, ha az az elnevezés adott, hogy *TC_QRDQN_LLL_8*, akkor annak jelentése, hogy

- A *Twocompartment* környezet került felhasználásra
- A *QRDQN* módszer került alkalmazásra
- A *Long Living Linear* jutalomfüggvény került felhasználásra
- Ez a kísérlet ezzel a felépítéssel a nyolcadik kísérlet

4.1. Konkrét eredmények

Az eredmények ismertetése során minden teszteredmény esetén

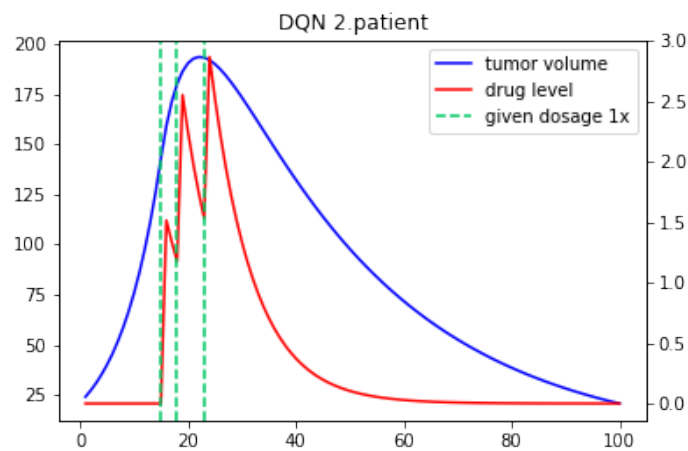
- a *Twocompartment* Drexler modellt vettük figyelembe, továbbá
- a *Long Living Linear* jutalomfüggvényt használtuk fel,

ezáltal lehetővé téve, hogy az egyes módszerek által adott eredmények, azon belül is a jutalomfüggvények értékei minél jobban értelmezhetőek, összehasonlíthatóak legyenek.

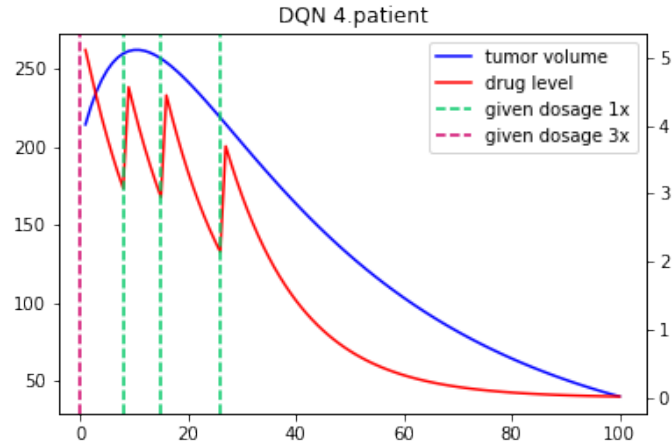
4.1.1. DQN teszt

	<i>reward</i>	<i>steps</i>
<i>minimum</i>	-23830	1
<i>maximum</i>	515000	100
<i>átlag</i>	361750	87.273

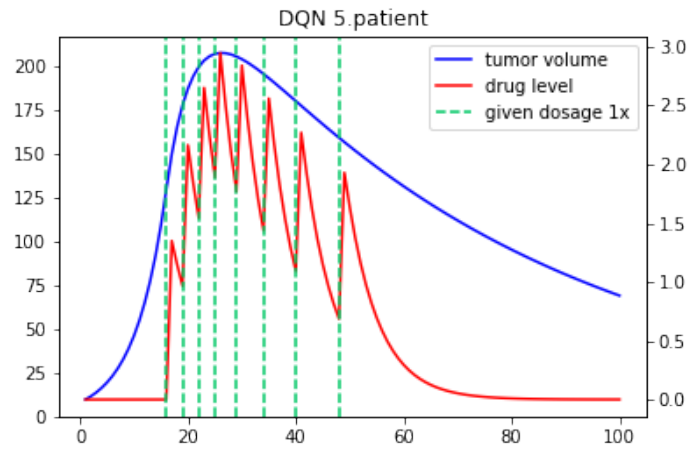
4.1. táblázat. DQN környezetben mért tesztek eredményei



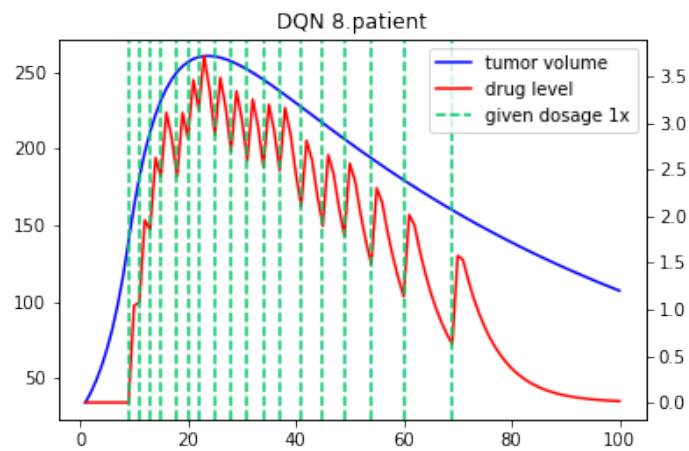
4.1. ábra. 2-es tesztalany DQN környezetben látható eredménye



4.2. ábra. 4-es tesztalany DQN környezetben látható eredménye



4.3. ábra. 5-ös tesztalany DQN környezetben látható eredménye

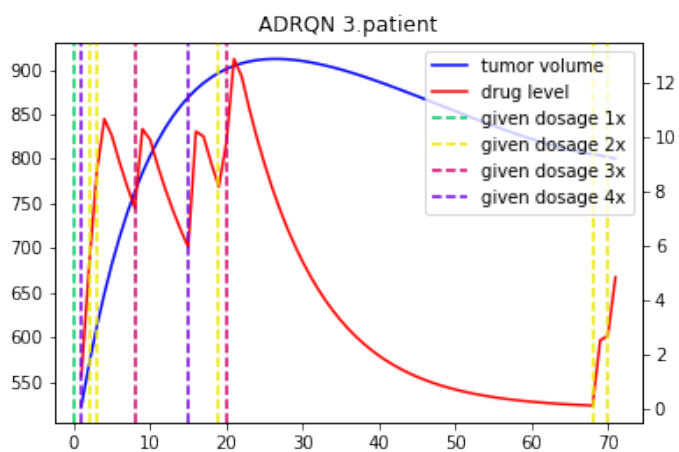


4.4. ábra. 8-as tesztalany DQN környezetben látható eredménye

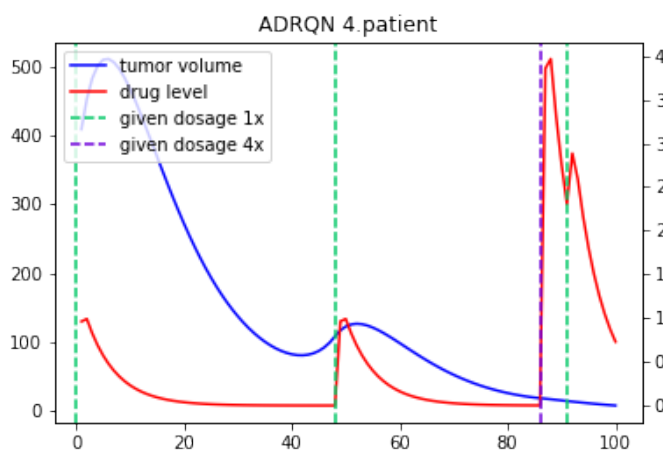
4.1.2. ARDQN teszt

	<i>reward</i>	<i>steps</i>
<i>minimum</i>	-20281.28	0
<i>maximum</i>	514986.63	100
<i>átlag</i>	377881.32	82.62

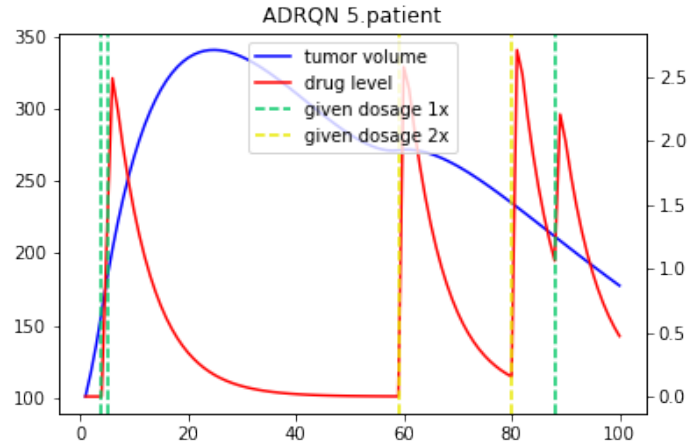
4.2. táblázat. ARDQN környezetben mért tesztek eredményei



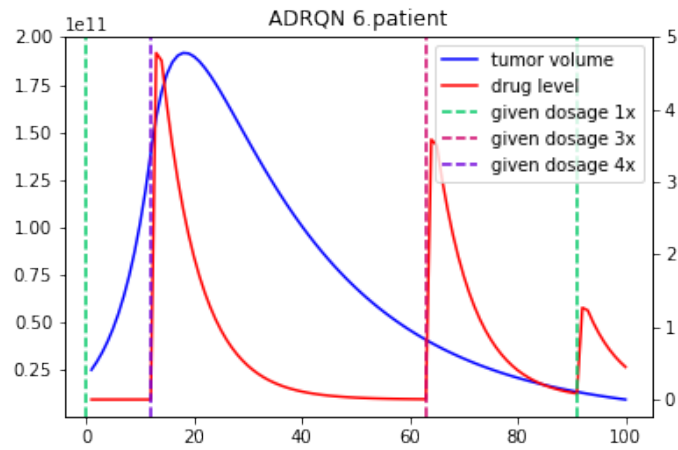
4.5. ábra. 3-as tesztalany ARDQN környezetben látható eredménye



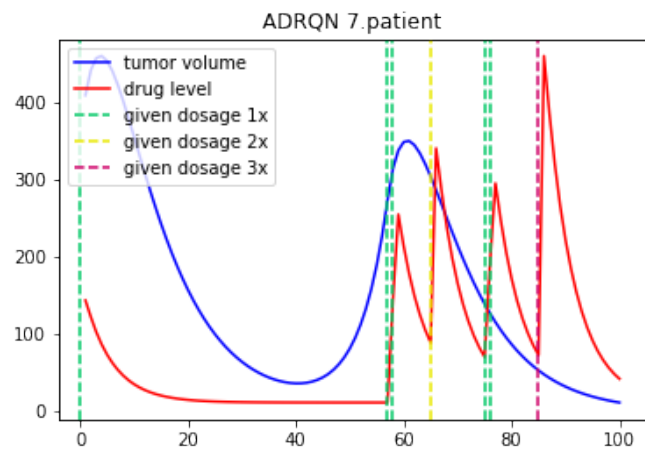
4.6. ábra. 4-es tesztalany ARDQN környezetben látható eredménye



4.7. ábra. 5-ös tesztalány ARDQN környezetben látható eredménye



4.8. ábra. 6-os tesztalány ARDQN környezetben látható eredménye

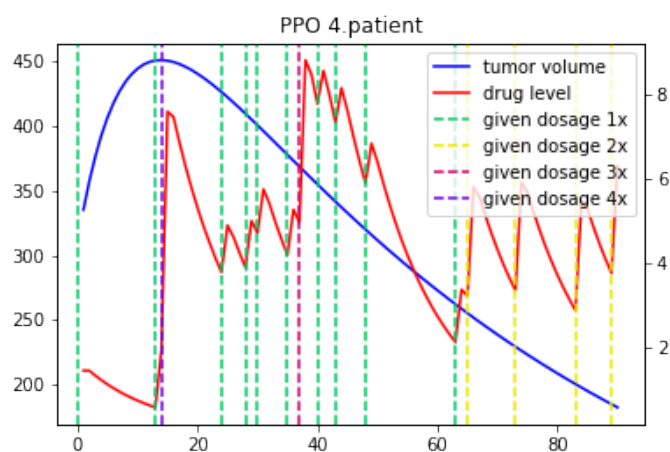


4.9. ábra. 7-es tesztalány ARDQN környezetben látható eredménye

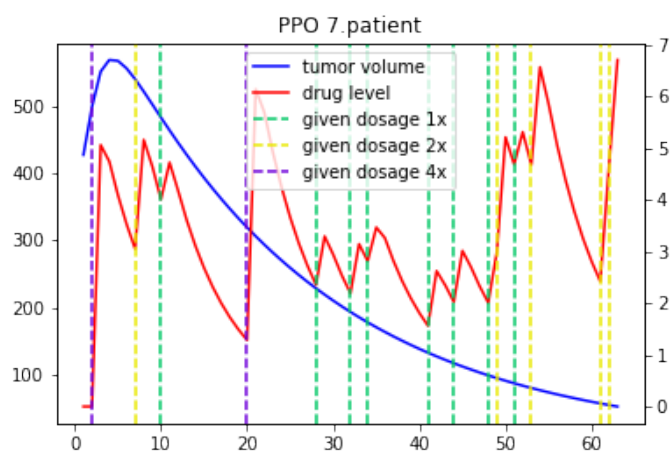
4.1.3. PPO teszt

	<i>reward</i>	<i>steps</i>
<i>minimum</i>	-19170.35	0
<i>maximum</i>	510191.29	100
<i>átlag</i>	135388.14	45.08

4.3. táblázat. PPO környezetben mért tesztek eredményei



4.10. ábra. 4-es tesztalany PPO környezetben látható eredménye

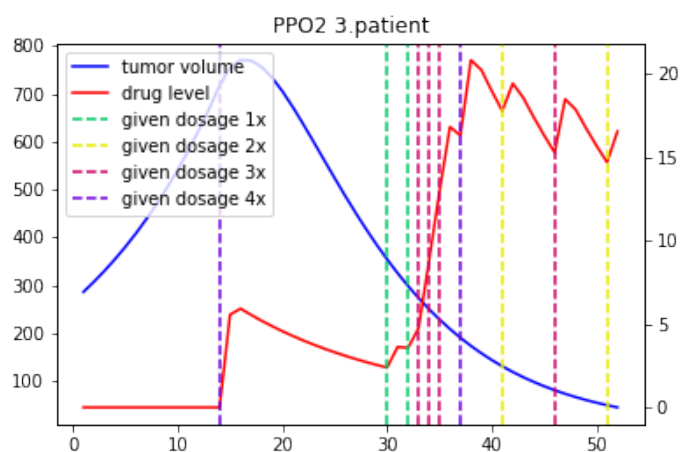


4.11. ábra. 7-es tesztalany PPO környezetben látható eredménye

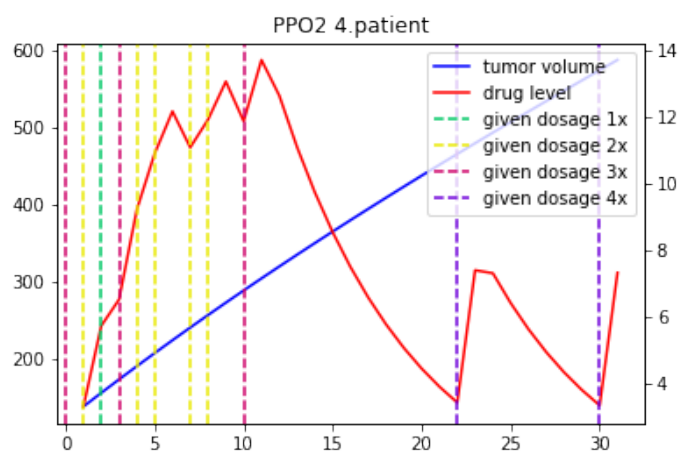
4.1.4. PPO2 teszt

	<i>reward</i>	<i>steps</i>
<i>minimum</i>	-21034.69	0
<i>maximum</i>	440764.87	100
<i>átlag</i>	16845.85	20.95

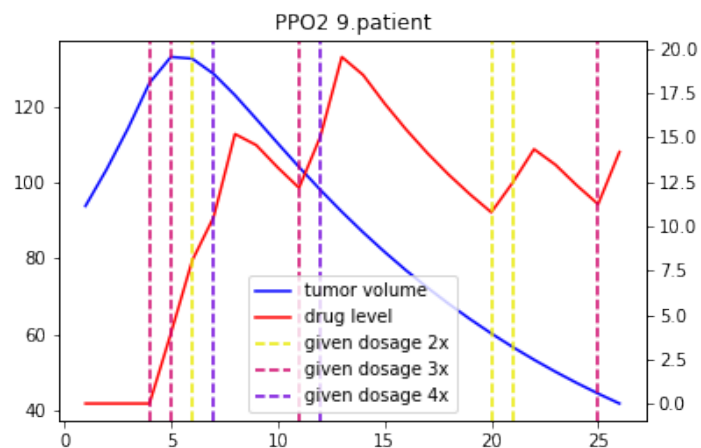
4.4. táblázat. PPO2 környezetben mért tesztek eredményei



4.12. ábra. 3-as tesztalany PPO2 környezetben látható eredménye



4.13. ábra. 4-es tesztalany PPO2 környezetben látható eredménye

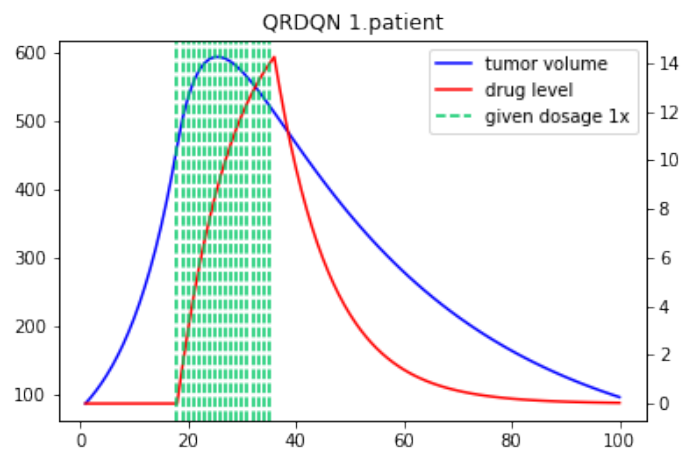


4.14. ábra. 9-es tesztalany PPO2 környezetben látható eredménye

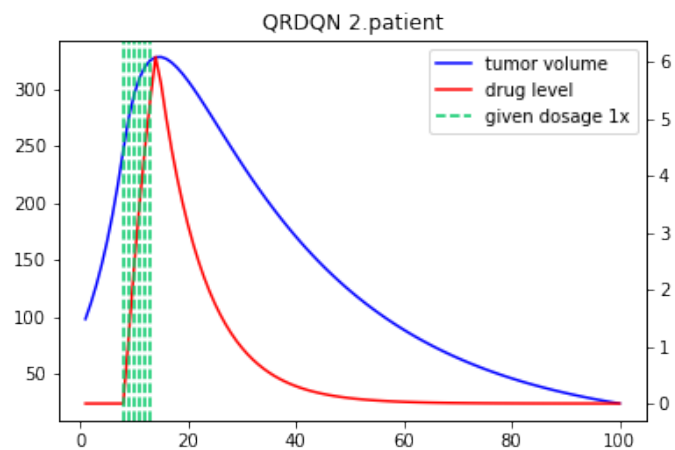
4.1.5. QRDQN teszt

	<i>reward</i>	<i>steps</i>
<i>minimum</i>	-23400	0
<i>maximum</i>	515000	100
<i>átlag</i>	387620	82.37

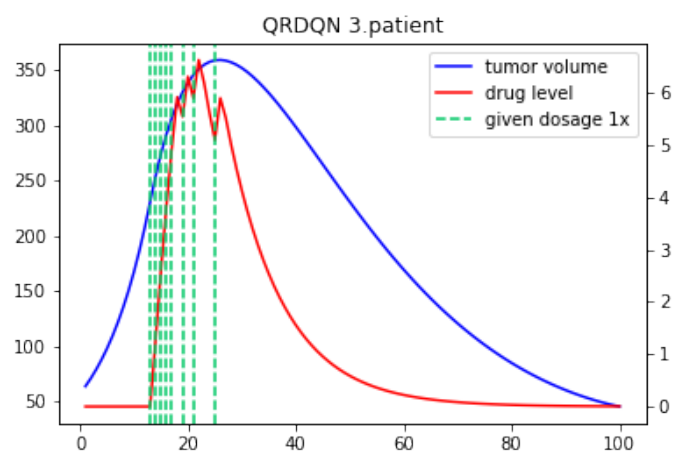
4.5. táblázat. QRDQN környezetben mért tesztek eredményei



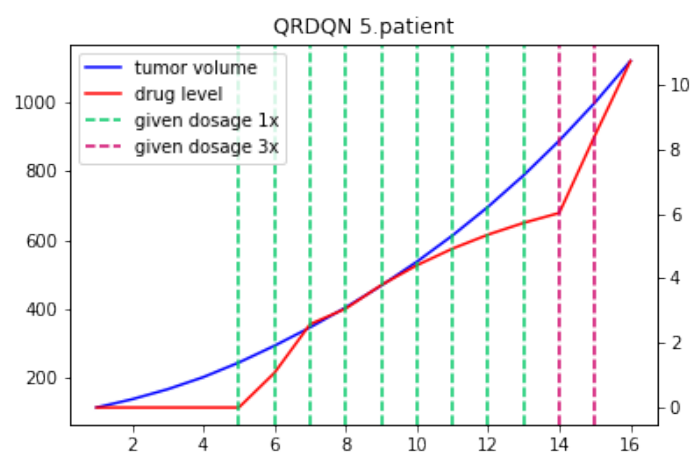
4.15. ábra. 1-es tesztalany QRDQN környezetben látható eredménye



4.16. ábra. 2-es tesztalany QRDQN környezetben látható eredménye



4.17. ábra. 3-as tesztalany QRDQN környezetben látható eredménye



4.18. ábra. 5-ös tesztalany QRDQN környezetben látható eredménye

4.2. Eredmények értékelése

Az eredmények a következőkben taglalt folyamat végén állnak elő. Legelőször generálunk virtuális alanyokat véletlenszerűen paraméterezett tulajdonságokkal, figyelve arra, hogy a valódihoz hasonló tulajdonságokkal rendelkezzenek, csak véletlenszerűen. Ezeken a generált alanyokon zajlik a tanítás maga. Egy ciklus egy kezelést jelent, ez az *episode*, ami *episode* 100 napig tart. A változók közül a *steps* nevű változó azt mutatja meg, hogy egy alany hány napig élt túl. Ez a változó az, ami igazán azt mutatja meg, hogy a különböző megoldások mennyire jók, hogy mennyi ideig sikerült életben tartani az egyedet. A másik változó, a *reward* pedig a tanulási, illetve tesztegységek modellre történő illesztése során adódó értéket fog meghatározni, amely, következésképpen a függvény lineáris voltából, minél magasabb értéket vesz fel, annál inkább valószínűsíthető és állapítható meg jobb, hatékonyabb eljárási módszer, kezelés.

4.2.1. Tesztek struktúrája

Ahogy az már említésre került, a tesztek mindegyikéről elmondható, hogy a *twocompartment* Drexler modell matematikai háttere mellett mint jutalomfüggvény rögzítése került a *Long Living Linear* függvény. Röviden, részletekbe nem belemenve, ez a függvény pozitívan jutalmazza, hogy ha a páciens minél tovább él, míg a mortalitást egy elég jelentős negatív jutalommal honorálja. Ezen párost felhasználva történtek meg az egyes módszerek tesztelése.

Egy adott módszer tesztje egy adott modellen és fix jutalomfüggvény mellett az alábbiak szerint épül fel:

- Véletlenszerűen megválasztott paraméterezésű tumormodell alapján az ágens tanítása
- 1000 darab kezelés végrehajtása random generált egyedeken
- Néhány fix paraméterezésű virtuális alany konkrét kezelése, részletesen

Az 1000 darab kezelésből származó információk, az egyes esetek elején, táblázat formájában vannak összesítve, megjelenítve a minimum, az átlag és a maximum értékeket mind a jutalom, mind a szimulált kezelés időperiódusa, a *steps* szempontjából. A táblázat után pedig következnek azon diagramok, melyek bizonyos, fix paraméterrel rendelkező tesztegységek rendelkeztek. A fix paraméterezés azt jelenti, hogy egy-egy sorszámozott alany tulajdonságai, tumorviselkedési jellemzői teszteltől függetlenül mindig ugyanazok. Emiatt egy adott egyed segítségével rendkívül jól össze lehet hasonlítani a módszerek hatását egy-egy kezelés során, ha minden más, a tanulást befolyásoló komponens, mint matematikai modell vagy mint jutalomfüggvény rögzítve lett.

Nem elhanyagolandó sem a jutalom, sem a *steps* értékek során az átlagérték, amelynél elvárás, hogy minél inkább közelítsen a maximumhoz, másképp megfogalmazva, minél kisebb különbség legyen a maximum és az átlag között.

4.2.2. Módszertanok összehasonlító elemzése

Az eljárások összehasonlító elemzése során alapul a DQN módszer által adott eredmények tekintendők. A DQN eljárásban rejlő potenciált korábbi TDK munkánk

során részleteit tekintve górcső alá vettük [25]. Ugyanakkor ezen munka is tartalmaz ahhoz képest előrelépést, már az alapnak számító koncepcióban is. A környezeti implementáció, módszertani részletezés során említésre került, hogy immáron bevezetésre került az egy kezelés alatt adható maximális gyógyszer mennyisége (*doseMax*), az egy kezelés során beadott összes gyógyszer mennyiség (*doseSum*), továbbá lehetősége van az *agent*-nek többféle döntés meghozatalára, ha ad gyógyszert, akkor mennyit, *multi-action* lehetőséget adva. A teszteredményekből általánosságban, ezek figyelembe vétele mellett, az olvasható ki, hogy a kezelések döntő részében hatékonynak bizonyulnak. A 4.1 táblázatból kiolvasható maximumhoz közelítő *reward* és *steps* értékek a kezelések hatásosságát mutatják, jóval tovább élnek túl a virtuális egyedek. Ezt megerősítik a hozzá kapcsolható ábrák is. A diagramok viszont egy fokkal árnyaltabb képet mutatnak. Ugyanis hiába került implementálásra a *multi-action* alapú megközelítés, néhány kivételtől eltekintve egyféle gyógyszer mennyiséget ad a páciensek számára, noha ez a mennyiség éppen a legkisebb adható mennyiség is egyben. Előfordul, hogy szinte kihagyás nélkül adagolja a következő gyógyszert. Ezen ellenpontok mellett, noha megállapítható, hogy az eljárás megfelelő, kereshető és megállapítható ennél ideálisabb megoldás is. Az általános elvárásoknak megfelel, és a szerint is tanul a rendszer, de a "haladóbb" elvárásokat nem teljesíti maradéktalanul.

DQN & DRQN összehasonlítása

Amennyiben össze akarjuk hasonlítani az alap eljárásnak számító DQN eredményeit a DRQN eredményekkel, akkor az alábbiakat lehet megállapítani:

- A végrehajtott *action*-ök nagyobb variációt mutatnak. A DRQN módszer felhasználja a *multi-action* megközelítést, válogat az adható gyógyszer mennyiség között, így sokkal finomabb, páciensre, tumor méretre szabottabb kezelési eljárást tud kialakítani.
- A DQN módszerrel szemben nem fordul elő jellemzően a rövid idő alatt többszöri gyógyszer adása a páciensek számára. A DRQN módszer során a modell törekszik arra, hogy ha már gyógyszert kell adni, akkor azt úgy adja, hogy a megfelelő mennyiségben. Ugyanakkor ellenpéldaként a 4.9. ábrán látható, hogy nem is egyszer ad kevésnek számító dózist a páciens számára. Ezt leszámítva általánosan jónak tekinthető.
- Ha a számszerűsített adatok vannak figyelembe véve, úgy hasonló eredményeket lehet kiolvasni a 4.2. táblázatból, mint amit a DQN-nél is lehetett látni. Az átlagértékek közelítenek a megfigyelhető maximumértékhez, igen kevés a különbség, sőt, *reward* érték szerint még jobbnak is bizonyul.

Megállapítható továbbá, hogy csak bizonyos tumortérfogat mellett adagol gyógyszert, továbbá ha azt érzékeli, hogy a tumortérfogat növekedésnek indul, úgy újból adagol. Emiatt általános szempontokból is megfelelőnek tekinthető, de még a haladóbb technikák szerint is: a maximális dózist nem lépi át, a tumor térfogat csökkenő tendenciát mutat a gyógyszer szint minél alacsonyabb szinten tartása mellett.

DQN & PPO-PPO2 összehasonlítása

Együttes figyelembe vétele mellett a PPO és a PPO2 igen hasonló eredményeket produkáltak, így célszerű ezen két eljárást együttesen a DQN-nel összehasonlítani. Az adatokból az alábbiak olvashatóak ki:

- A DQN eljárással ellentétben a PPO és a PPO2 is igyekszik alkalmazni a *multi-action* alapú megközelítésben rejlő potenciált. Noha előfordulnak sűrűbb adagolások, mégsem oly sűrűn, mint DQN esetében volt.
- Míg a DQN módszernél alacsony szinten van tartva a gyógyszer mennyisége, annyira, amennyire azt a kezelés igényli, úgy a PPO alapú megoldások esetén igen magas szintet mutatnak az azt mutató diagramok.
- Ha a számszerűsített adatokat vesszük figyelembe, úgy a DQN eljárás tesztjeiből kiolvasható értékek jóval jobbnak bizonyulnak, mint a PPO páros esetén. Olyannyira, hogy a DQN és a PPO2 eljárások átlag *túlélési ideje* szerint közel négyszeres különbség mutatkozik, négyszer gyengébb értéket ad a PPO2. De a PPO sem sokkal jobb, közel feleolyan magas értéket állapított meg, mint azt DQN esetén lehetett tenni.

Ahogy azt az egyes ábrákból és a megfelelő táblázatokból kiolvashatni lehet, a PPO-alapú eljárások nem teljesítik túl a DQN alapú eljárást, sőt, a tanítási eredményekből az olvasható ki, hogy a nem sikerül megfelelő kezelési tervet kialakítani. Jellemzőek az indokolatlan döntések mindamelllett, hogy a tumor térfogatának változása olykor pozitív mértékű, tehát csökkenésnek, illetve stabil csökkenésnek indulnak.

DQN & QRDQN összehasonlítása

Figyelembe véve a QRDQN módszer által adott eredményeket az alábbiakat lehet elmondani:

- A QRDQN megfigyelhető *action*-ök változatosságát tekintve, hasonlóan a DQN módszerhez *single-action* megközelítés jellemzi leginkább, annak ellenére is, hogy ezen módszer is *multi-action* alapú döntéshozásra be van állítva, a tanulási folyamat során többféle gyógyszeradagból választhat. Ennek ellenére ez a megoldás csupán egyféle dózist adagol, abból is a legkevesebbet jellemzően. Továbbá az is látszódik, hogy amennyiben adagol gyógyszert, akkor azt a gyógyszer mennyiséget, amit akar a páciensnek juttatni, rövid idő alatt be is adagolja, igaz, a legkisebb elérhető egységben. Van ugyanakkor egy-egy kivétel QRDQN tekintetében. A QRDQN 4.18. ábrán az látszik, hogy a folyamat végén próbálkozik a modell egy-egy nagyobb gyógyszer adagolásával, mindhiába, a páciens terminálásával a kezelés idővel korábban véget ér.
- Ha figyelembe a gyógyszer adagolási sűrűsége kerül, úgy hasonlóan a DQN-nél megfigyeltékhez, itt is gyakran lehet találkozni olyan viselkedéssel, hogy a rendszer rövid idő alatt adagol többször, keveset. Annak ellenére, hogy rendelkezésére állna egy nagyobb adag páciensnek adagolása, mégse azt használja fel. A periodikus adagolás sem általánosan megfigyelhető.

- Számszerűsített adatait tekintve is párhuzamot lehet vonni a DQN adataival, ahogy azt a 4.5. táblázatból kiolvasható. Az átlag *steps* érték gyengébbnek bizonyul, ugyanakkor az átlag *reward* pedig magasabbnak, és még itt is közelebb állnak az átlagértékek a maximumhoz, mint a minimum és a maximum számtani közepe.

Nagy átlagban, a diagramokból és a táblázat adataiból az olvasható ki, hogy alapvető koncepciókat tekintve a QRDQN is megfelelőnek bizonyul. A gyakori, sokszori gyógyszeradagolások ellenére a bevitt gyógyszer összemennyisége nem magas, mindemellett sikerül a tumor térfogatát csökkenteni a gyógyszer szint viszonylag alacsony szinten tartása mellett. Viszont, mint a DQN-nél is látható volt, elvész a *multi-action*-ben lévő lehetőség, nem használja fel a tanulás során.

DRQN & QRDQN összehasonlítása

Az eredményekből az jól látszik, jól kiolvasható, hogy a QRDQN és a DRQN módszerek kisebb-nagyobb mértékben jobbnak bizonyulnak a kiindulási eljáráshoz képest. Természetesen kiolvashatóak ezen módszerek között is a különbségek, mely különbségek hasonlíthatóak az egyes eljárások DQN-nel történt összehasonlítása során megállapított állításokkal.

- Az alkalmazott *action*-ök számát tekintve a DQRN a hatékonyabbnak bizonyuló *multi-action* eljárást tanulta meg és alkalmazta jól, míg a QRDQN jellemzően *single-action* megközelítést tart önmaga számára optimális stratégiának.
- Eloszlását tekintve a QRDQN viszonylag rövid időn belül adagolja azt a gyógyszer mennyiséget, amennyit akar adagolni, ellenben a DQRN a tumor térfogatot és annak esetleges újbóli növekedését figyelembe véve dönt újbóli adagolásról
- A 4.2. és 4.5. táblázatokat összehasonlítva megállapítható, hogy az átlag értékek szinte megegyeznek egymással, az átlag *steps* értékek között mindössze néhány századnyi különbség van. Azaz elmondható a két eljárásról, hogy a "számok szerint" hasonlóan képesek teljesíteni.

4.2.3. Konklúziók a módszertanokról

Összefoglalva az egyes módszertanokat, az alábbiakat lehet megállapítani:

- *DQN*: Az általunk kiindulási pontnak tekinthető DQN eljárás az egyes fejlesztések alkalmazása után is megfelelő eredményt volt képes produkálni. A maximálisan adható gyógyszer mennyiséget és a maximum páciensben lévő gyógyszer mennyiséget alkalmasan megtanulva nem lépi át. Ugyanakkor gyengeségei közé tartozik, hogy mindössze egyféle döntést képes meghozni, jóformán nem alkalmazza a többszöri döntés lehetőségét. A meghatározott gyógyszer adagonkénti mennyiségét sűrűn adagolja be jellemzően kis dózisokban. Viszont a kezelési módszer megfelelő, az ahhoz tartozó paraméterek megfelelően viselkednek. Számokkal kifejezhető hatékonysági mutatói az eljárás megfelelőségét bizonyítják.

- *DRQN*: Minden más eljárást figyelembe véve leginkább a *DRQN*-ről mondható el, hogy a megfelelő módon sikerült megtanulnia a tumor viselkedési modelljét, illetve ahhoz szorosan kapcsolódóan a tumor térfogat optimalizálását, kihasználva az idősoros adatok alkalmazását, hiszen nem csak adott pillanatra vonatkozóan dönt a rendszer adott pillanatban. Alkalmazza a *multi-action* alapú megközelítést, nem csak egyféle gyógyszer mennyiséget adagol, és amit adagol is, azt leginkább időben elosztva, a tumorra reagálva teszi. Nem jellemzőek az egyszerre nagy mennyiségben beadott gyógyszerek, a páciensben lévő gyógyszer szint alacsonyan van tartva. Csak abban az esetben adagol továbbá, ha éppen arra leginkább szükség van, azaz ha a tumor térfogata egy bizonyos érték fölé csúszik. *Reward* és *steps* értékei a számtani átlagtól jelentősen eltérőek, pozitív irányban.
- *PPO-PPO2*: Ha kontextusba helyezzük a többi, jelen kutatásunkban vizsgált további módszertanokkal, akkor azt lehet megállapítani, hogy a *PPO* alapú eljárások teljesítenek mind közül a leggyengébben. A *multi-action* alapú megközelítést noha használja, de nem megfelelően, így is jellemzi olykor gyakori gyógyszeradagolás a páciens számára. Számszerű adataiból sem olvasható ki pozitív információ: az átlag értékek inkább az átlag és a minimum között helyezkednek el. Elmondható, hogy az eljárások hasznossága és a jelenleg kiolvasható eredmények kontrasztjában szükséges a módszer további fejlesztése, egyes paramétereinek finomhangolása, jutalomfüggvényhez illesztése.
- *QRDQN*: Döntéseit legtöbb esetben egyféle *action* alapján határozza meg, amit a tanulási folyamat során elsajátított. Noha az általa adagolt gyógyszer egyszeri mennyisége nem túl magas, a gyógyszer szint maximumának elérése nélkül, de időben egymást szorosan kötve adagolja ezt a bizonyos mennyiséget. Abban az esetben próbálkozik más mennyiséggel, ha azt látja a modell, hogy az addigi mikroadagos kezelési mód nem vezet sikerre. Általánosan, többségében megfelelő eredményeket produkál, a páciensek jelentős része megéli terminálás nélkül a tesztelési időtartamot, melyet alátámasztanak a jutalomértékek és *steps* értékek is.

4.2.4. Eredmények összegzése

Az összes módszertant figyelembe véve két csoportot lehet elkülöníteni: az egyik csoport módszerei tudják és képesek alkalmazni a *multi-action* megközelítést, míg a másik csoport tagjai nem, és inkább maradnak a *single-action* eljárásnál. Ugyanakkor figyelembe vehető tényező, hogy azon eljárások, melyek csupán egyféle *action*-nel dolgoznak, megfelelő eredményt adnak. E mellett természetesen megemlítenendő, hogy azon módszerek, melyek képesek ki is használni maximálisan a több lehetséges *action*-ben rejlő lehetőséget, és működőképesek, úgy azok még inkább jobb eredménnyel szolgálnak. Ez lehetővé teszi az egyénre, a páciensre szabott kezelésnek a lehetőségét, sokkal rugalmasabb rendszert eredményezve. Ha egy viszonylagos sorrendet fel lehetne állítani a tesztelés során megismert tapasztalatokból, úgy első helyre az idősoros adatokat figyelembe vevő *DRQN* módszer kerülne. Mögötte lenne a *QRDQN* az illesztési módszerével, szorosan vele együtt pedig a *DQN*. A *PPO-PPO2* módszerek, mivel jelen kutatásra nem megfelelően kerültek illesztésre jelen dolgozat megírásának pillanatáig, úgy noha felhasználja a *multi-action* megkö-

zelítést, nem számítanak megfelelőnek a kiolvasható értékek, még az alapnak szolgáló DQN-nél sem.

5. fejezet

Összefoglalás

A tumorkezelés, ahogy azt eddigi tapasztalatok mutatták, rendkívül összetett feladat, a célok meghatározása és az eredmények értékelése is többszintű.

A tumor viselkedésének szimulálásához többféle matematikai modellt vettünk figyelembe, végül a Drexler-féle kétkompartment modellel dolgoztunk. Ennek alapján felépítettünk egy Markov Döntési Folyamatot reprezentáló környezetet, melyben az ágens viselkedését különböző mély megerősítéses tanulással végeztük el.

Az irodalomkutatás során különféle mély megerősítéses tanulási eljárásokkal találkozhattunk. Ezeket implementáltuk és egységesítettük az adatgyűjtő és -kiértékelő részeket. Bővítettük, finomítottuk a Drexler modellből felépülő környezetünket. Ezt követően számos szimuláció került elvégzésre, melyek eredményeiből következtetni lehetett az egyes mély megerősítéses tanulási megoldások hatékonyságára, későbbi alkalmazhatóságára.

5.1. Továbbfejlesztési és további kutatási tervek

Továbbfejlesztési, kutatási tervek tekintetében a következő irányokat határoztuk meg:

- További megerősítéses tanulási eljárások vizsgálata
- Saját megerősítéses tanulási eljárás fejlesztése, meglévők alapján
- Valóshűbb modell reprezentáció
- Elosztott rendszerek használata a tanításban

Célszerű megvizsgálni további megerősítéses tanulási eljárásokat is. Ilyen például a DDQN (Double Deep Q-Network) [30] ahol a rendszer egy Q érték kiszámítását 2 különálló neurális hálóval valósítja meg. A másik irány, ami ide tartozik az a D3QN (Duelling Double Deep Q-Network) [31] ahol két neurális háló versengve egymással oldja meg a feladatot. Mindkettőnek más -más előnye van, amiket érdemes kihasználni és kipróbálni a jövőben.

Ahogy az előző pályamunkában megjelentek a saját jutalmazási függvények, úgy itt is egy valós gondolat az, hogy saját megerősítéses tanulási eljárást fejlesszünk a meglévők előnyeinek ötvözésével. Természetesen ez egy rendkívül komplex folyamat és valószínűleg minden rendszer előnyét nem, vagy csak nehezen lehet egybe építeni,

azonban ez az irány is megér egy próbát. Számításaink szerint ez önmagában képes akkora feladat lenni, mint ez a jelenlegi kutatói munka.

A kísérletek esetén célszerű lesz a valósághoz is igazodni: modell-alapon természetesen következtethetünk a vérben levő dózismennyiségre, a valóságban ez nem megoldható. Hasonlóképp a tumor méretének megállapítása méréssel történik, amely zajt adhat a vizsgálatokhoz. Így az egyik továbbfejlesztési cél egy olyan szimulációs rendszer létrehozása, amely a valósághoz közeli, matematikai tumornövekedés-modellen alapul, és alkalmas arra, hogy egy ágens elsajátítsa az optimális kezelést ritkán, kis mennyiségekben bejuttatott dózisokkal.

Elosztott vagy felhő alapú rendszerek használata sem egy utolsó továbbfejlesztési lehetőség. Ezzel a huzamosabb, tovább tartó lassabb tanítások is kivitelezhetők, a helyi lokális rendszer leterhelése nélkül. A végső célunk az az, hogy egy paranccsal kellően standardizált rendszerben képesek legyünk kedvünkre összeválogatott, modellel, megoldással, jutalomfüggvénnyel és egyéb más paraméterekkel tanítási „csomagokat” indítani, adatait kimenteni és automatizálni ezen folyamatokat.

Ábrajegyzék

2.1. Ahogy a megerősítéses tanulást el lehet képzelni [2].	7
2.2. A megerősítéses tanulás valószínűségekkel [5].	7
2.3. A Q-Learning egyszerűsített architektúrája	13
2.4. Az DQN egyszerűsített architektúrája	14
2.5. A DRQN egyszerűsített architektúrája	15
2.6. A Csúszó ablakos különbözeteken alapuló DQN egyszerűsített architektúrája	16
2.7. A PPO-DQN és PPO2-DQN egyszerűsített architektúrája egy képen .	17
2.8. A QR-DQN egyszerűsített architektúrája	17
4.1. 2-es tesztalany DQN környezetben látható eredménye	29
4.2. 4-es tesztalany DQN környezetben látható eredménye	30
4.3. 5-ös tesztalany DQN környezetben látható eredménye	30
4.4. 8-as tesztalany DQN környezetben látható eredménye	30
4.5. 3-as tesztalany ARDQN környezetben látható eredménye	31
4.6. 4-es tesztalany ARDQN környezetben látható eredménye	31
4.7. 5-ös tesztalany ARDQN környezetben látható eredménye	32
4.8. 6-os tesztalany ARDQN környezetben látható eredménye	32
4.9. 7-es tesztalany ARDQN környezetben látható eredménye	32
4.10. 4-es tesztalany PPO környezetben látható eredménye	33
4.11. 7-es tesztalany PPO környezetben látható eredménye	33
4.12. 3-as tesztalany PPO2 környezetben látható eredménye	34
4.13. 4-es tesztalany PPO2 környezetben látható eredménye	34
4.14. 9-es tesztalany PPO2 környezetben látható eredménye	35
4.15. 1-es tesztalany QRDQN környezetben látható eredménye	35
4.16. 2-es tesztalany QRDQN környezetben látható eredménye	36
4.17. 3-as tesztalany QRDQN környezetben látható eredménye	36
4.18. 5-ös tesztalany QRDQN környezetben látható eredménye	36

Táblázatok jegyzéke

4.1. DQN környezetben mért tesztek eredményei	29
4.2. ARDQN környezetben mért tesztek eredményei	31
4.3. PPO környezetben mért tesztek eredményei	33
4.4. PPO2 környezetben mért tesztek eredményei	34
4.5. QRDQN környezetben mért tesztek eredményei	35

Forráskódok listája

1.	A <i>Discrete Drexler Environment</i> osztálydefiníciója és konstruktora . .	22
2.	A <i>Drexler modell</i> next függvénye	23
3.	A <i>Keras</i> alapú eljárások struktúrája, egyszerű DQN	24
4.	A <i>Torch</i> alapú eljárások struktúrája, PPO2	24
5.	A <i>Keras</i> alapú tesztelés és adatgyűjtés	25
6.	A <i>Keras</i> alapú tesztelés és adatgyűjtés	26

Irodalomjegyzék

- [1] Richard S Sutton és Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [2] Code-AI. *How to implement Q Learning algorithm in C sharp*. URL: <https://code-ai.mk/how-to-implement-q-learning-algorithm-in-c/>. (utoljára megnézve: 04.10.2021).
- [3] Richard Bellman. “A Markovian decision process”. *Journal of mathematics and mechanics* 6.5 (1957), 679–684. old.
- [4] John N Tsitsiklis. “Asynchronous stochastic approximation and Q-learning”. *Machine learning* 16.3 (1994), 185–202. old.
- [5] blackburn. *Reinforcement Learning : Markov-Decision Process*. URL: <https://towardsdatascience.com/introduction-to-reinforcement-learning-markov-decision-process-44c533ebf8da>. (accessed: 04.10.2021).
- [6] Ludwig Von Bertalanffy. “Quantitative laws in metabolism and growth”. *The quarterly review of biology* 32.3 (1957), 217–231. old.
- [7] RK Sachs, LR Hlatky és P Hahnfeldt. “Simple ODE models of tumor growth and anti-angiogenic or radiation treatment”. *Mathematical and Computer Modelling* 33.12-13 (2001), 1297–1305. old.
- [8] Hyun M Yang. “Mathematical modeling of solid cancer growth with angiogenesis”. *Theoretical Biology and Medical Modelling* 9.1 (2012), 1–39. old.
- [9] Dániel András Drexler, Tamás Ferenci, Anna Lovrics és Levente Kovács. “Modeling of tumor growth incorporating the effect of pegylated liposomal doxorubicin”. *2019 IEEE 23rd International Conference on Intelligent Engineering Systems (INES)*. IEEE. 2019, 369–000374. old.
- [10] Dániel András Drexler, Tamás Ferenci, András Füredi, Gergely Szakács és Levente Kovács. “Experimental data-driven tumor modeling for chemotherapy”. *IFAC-PapersOnLine* 53.2 (2020). 21th IFAC World Congress, 16245–16250. old. ISSN: 2405-8963. DOI: <https://doi.org/10.1016/j.ifacol.2020.12.619>. URL: <https://www.sciencedirect.com/science/article/pii/S240589632030923X>.
- [11] Christopher JCH Watkins és Peter Dayan. “Q-learning”. *Machine learning* 8.3-4 (1992), 279–292. old.
- [12] Tuomas W Sandholm és Robert H Crites. “Multiagent reinforcement learning in the iterated prisoner’s dilemma”. *Biosystems* 37.1-2 (1996), 147–166. old.
- [13] Nicolò Cesa-Bianchi, Claudio Gentile, Gabor Lugosi és Gergely Neu. “Boltzmann Exploration Done Right”. *Advances in Neural Information Processing Systems* 30 (2017), 6284–6293. old.

- [14] Leslie Pack Kaelbling, Michael L Littman és Andrew W Moore. “Reinforcement learning: A survey”. *Journal of artificial intelligence research* 4 (1996), 237–285. old.
- [15] Michel Tokic és Günther Palm. “Value-difference based exploration: adaptive control between epsilon-greedy and softmax”. *Annual conference on artificial intelligence*. Springer. 2011, 335–346. old.
- [16] Volodymyr Mnih és tsai. “Playing atari with deep reinforcement learning”. *arXiv preprint arXiv:1312.5602* (2013).
- [17] Volodymyr Mnih és tsai. “Human-level control through deep reinforcement learning”. *nature* 518.7540 (2015), 529–533. old.
- [18] Matthew Hausknecht és Peter Stone. “Deep recurrent q-learning for partially observable mdps”. *arXiv preprint arXiv:1507.06527* (2015).
- [19] Guillaume Lample és Devendra Singh Chaplot. “Playing FPS games with deep reinforcement learning”. *Proceedings of the AAAI Conference on Artificial Intelligence*. 31. köt. 1. 2017.
- [20] Christos H Papadimitriou és John N Tsitsiklis. “The complexity of Markov decision processes”. *Mathematics of operations research* 12.3 (1987), 441–450. old.
- [21] David E Rumelhart, Geoffrey E Hinton és Ronald J Williams. “Learning representations by back-propagating errors”. *nature* 323.6088 (1986), 533–536. old.
- [22] Sepp Hochreiter és Jürgen Schmidhuber. “Long short-term memory”. *Neural computation* 9.8 (1997), 1735–1780. old.
- [23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford és Oleg Klimov. “Proximal policy optimization algorithms”. *arXiv preprint arXiv:1707.06347* (2017).
- [24] Will Dabney, Mark Rowland, Marc Bellemare és Rémi Munos. “Distributional reinforcement learning with quantile regression”. *Proceedings of the AAAI Conference on Artificial Intelligence*. 32. köt. 1. 2018.
- [25] Almásy Márton György és Hörömpö András. *Tumorkezelési eljárás optimalizálása mély megerősítéses tanulás alapon*. 2021.
- [26] Parisa Yazdjerdi, Nader Meskin, Mohammad Al-Naemi, Ala-Eddin Al Moustafa és Levente Kovács. “Reinforcement learning-based control of tumor growth under anti-angiogenic therapy”. *Computer methods and programs in biomedicine* 173 (2019), 15–26. old.
- [27] A. Hassani és M. B. Naghibi.S. “Reinforcement learning based control of tumor growth with chemotherapy”. *2010 International Conference on System Science and Engineering*. 2010, 185–189. old. DOI: 10.1109/ICSSE.2010.5551776.
- [28] Yufan Zhao, Michael R Kosorok és Donglin Zeng. “Reinforcement learning design for cancer clinical trials”. *Statistics in medicine* 28.26 (2009), 3294–3315. old.
- [29] Gregory Yauney és Pratik Shah. “Reinforcement learning with action-derived rewards for chemotherapy and clinical trial dosing regimen selection”. *Machine Learning for Healthcare Conference*. PMLR. 2018, 161–226. old.

- [30] Hado Van Hasselt, Arthur Guez és David Silver. “Deep reinforcement learning with double q-learning”. *Proceedings of the AAAI Conference on Artificial Intelligence*. 30. köt. 1. 2016.
- [31] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot és Nando Freitas. “Dueling network architectures for deep reinforcement learning”. *International conference on machine learning*. PMLR. 2016, 1995–2003. old.