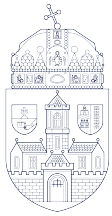


ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS INFORMATIKAI KAR

DIPLOMAMUNKA

OE-NIK Hallgató neve: **Almásy Márton György**
2022 Hallgató törzskönyvi száma: **T/008698/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Szoftvertervezés és -fejlesztés Intézet

DIPLOMAMUNKA FELADATLAP

Hallgató neve: **Almásy Márton György**
Törzskönyvi száma: T/008698/FI12904/N
Neptun kódja: DL87RU

A diplomamunka címe:

Tumor kezelési eljárások mély tanulás alapú optimalizálása
Deep learning based optimization of tumor treatment

Intézményi konzulens: Dr. Kertész Gábor, Kiss Dániel
Külső konzulens:

Beadási határidő: 2022. május 15.

A feladat

Az orvostudományban új kezelési eljárás, gyógyszer vagy épp védőoltás előzetes vizsgálata, tesztelése előbb szimulált környezetben történik, modell-alapú validációval. Ennek sikerét követően történhet annak kísérleti jellegű alkalmazása.

Ehhez természetesen a valós állapotot jól reprezentáló modellre van szükség, ezután lehetséges a tervezett kezelési paramétereket finomhangolni.

A diplomamunka célja különböző tumor kezelési eljárások vizsgálata szimulált környezetben: ehhez alakítson ki egy tumor növekedési matematikai modellen alapuló környezetet, majd mély megerősítéses tanulás alkalmazásával hozzon létre ideális kezelési eljárást!

A feladat része a releváns szakirodalom áttekintése, az elérhető keretrendszer bemutatása, a szoftveres implementáció, és végül az eredmények kiértékelése.

A diplomamunkának tartalmaznia kell:

- A szakirodalom részletes áttekintését, a választott matematikai modell bemutatását;
- A megerősítéses tanulás modern lehetőségeit, mély tanulás alapú state-of-the-art módszerek bemutatását;
- A szakirodalomban publikált hasonló célú megoldások összehasonlító elemzését;
- A választott megoldás specifikációját, részletes tervét;
- Az implementáció bemutatását;
- Az eredmények bemutatását, elemzését és értékelését;
- További kutatási-fejlesztési lehetőségek bemutatását.

ÓBUDAI EGYETEM * Neumann János Informatikai Kar * 1. Budapest

PH.

.....
Dr. Vámossy Zoltán
intézetigazgató

A diplomamunka elévülésének határideje: **2024. május 15.**
(OE TVSz 55.§ szerint)

A diplomamunkát beadásra alkalmasnak tartom:

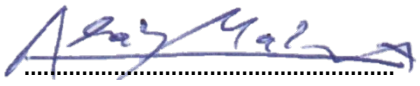
.....
külső konzulens

.....
intézményi konzulens

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2022.05.12.



.....

hallgató aláírása

Tartalomjegyzék

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, amely 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 súlyosság 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.

A diplomamunkában a kezelés folyamatára mint intelligens szabályozási eljárás van tekintve: 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).

Megerősítéses tanulás segítségével leírható az ágens és a környezet (a tumor mérhető jellemzői) közötti akción és jutalmazáson alapuló interakció. Jelen dolgozatban a létező megerősítéses tanuláson alapuló, hasonló problémákra adott megoldások kerülnek áttekintésre. Ide tartozik a különféle felépítések vizsgálata, a jutalmazási függvények elemzése, vizsgálata. A dolgozat további részében a szakirodalmi források áttekintése alapján kiválasztott megoldások implementálása, és összehasonlító elemzése kerül bemutatásra kifejezetten a tumorkezelési eljárás optimalizálásának céljával.

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 meg lehet 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 eljárás alaposan meg van vizsgálva, sok hasonlóságot lehet közte és a megerősítéses tanulás alapgondolata között találni, épp ezért a dolgozatban az van megvizsgálva, hogyan lehetne virtuális orvosoknak virtuális

pácienseken egy optimális 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 gyógyszerezési 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. Irodalomkutatás előzményei

Ezt a dokumentumot egy közös kutatás keretében alkottam meg Hörömpő András kutatótársammal. Munkánkból három darab publikáció készült, mely közül kettő a karon megrendezésre kerülő Tudományos Diákköri Konferenciához ([1] [2]) köthető, emellett pedig a Mexican International Conference on Artificial Intelligence-en (nemzetközi konferencián) ([3]) került bemutatásra. Az itt szereplő irodalomkutatás a nevezett dokumentumoknak részletes bemutatását tartalmazza.

2.2. A megerősítéses tanulás alapjai

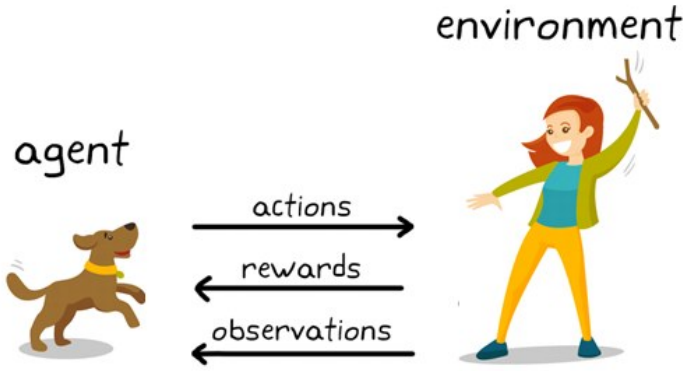
Ebben a fejezetben bemutatásra kerül nagy vonalakban az, hogy mi a megerősítéses tanulás (*reinforcement learning*, röviden: RL) [4]. Először is az alapfogalmakat, koncepciók lesznek vázolva, olyan kifejezések, szavak, amik a dolgozat hátralevő részében is előfordulnak, olyanokat, amik az RL-ben használatosak.

Fontos, hogy legelőször ketté legyen a megerősítéses tanulás és a felügyelt tanulás választva. 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 megerősítéses 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 alapfogalom kerül definiálásra és bevezetésre, 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 (műveletter)**: 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.

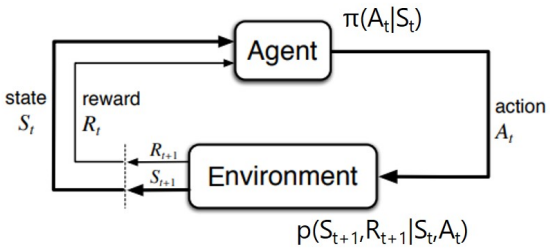


2.1. ábra. Ahogy a megerősítéses tanulást el lehet képzelni [5].

Egy konkrét, nagyon egyszerű példán keresztül mutatnám 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) [6], ahol az alább látható két feltételes valószínűség szerepel az MDP folyamatban [7]. Ezt a folyamatot, hajtja végre és oldja meg az RL rendszer. Ezt a viselkedést a következő ábra segíti megérteni:

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. ábra. A megerősítéses tanulás valószínüségekkel [8]

2.3. Tumornövekedés matematikai modelljei

A következőekben néhány általánosan használt tumornövekedési modell lesz bemutatva. Említésre kerülnek benne a modellek jellemzői, a modellek által használt leírási módok, valamint azok alkalmazhatósága.

2.3.1. Naiv megközelítések

A legelső, tumornövekedést leíró modellek több évtizedes múltra tekintenek vissza [9]. Az első megközelítések lényegében a populációdinamikában használatos megközelítéseket vették alapul, ezek közül a fontosabbakat R. K. Sachs cikke alapján lehet összefoglalni [10]. 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

$$\frac{dN}{dt} = \lambda N \tag{2.1}$$

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:

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

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.

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

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

2.3.2. 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 [11]. A modell esetén megemlítésre kerül egy *elő-angiogenezis* rész is, ahol a tumorsejtek a tömeghatás 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 logisztikus növekedési ütemet mutatnak.

A modell az alábbi differenciálegyenlet-rendszerrel írható fel:

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

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

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

$$\frac{dP}{dt} = \gamma ET - \delta P - \mu_4 P \tag{2.7}$$

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

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 tumorsejtek
- 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 hatékonynak bizonyulhat. A dolgozat esetében, a dolgozat szempontjából 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 ez a modell nem előnyös választás a probléma leírására.

2.3.3. 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 [12]. Magát a modellt rendszer- és irányításelmélet alapokra fektették, a szabályozás során ügyelve számos, kezeléssel kapcsolatosan felmerülő kérdésre. A modellt formális reakciólépésel alapján származtatott differenciálegyenletek segítségével írják fel az alábbi formában:

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

$$\dot{x_2} = n \cdot x_1 + b \cdot \frac{x_1 \cdot x_3}{ED_{50} + x_3} \tag{2.10}$$

$$\dot{x_3} = -c \cdot \frac{x_3}{K_B + x_3} - b_k \cdot \frac{x_1 \cdot x_3}{ED_{50} + x_3} + u \tag{2.11}$$

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 érné 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

$$\dot{y} = a \cdot x_1 \tag{2.12}$$

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 milligram/kilogrammal egyezik meg, ha az első

egyenletben szereplő a - n érték negatív szám, ami akkor lehet, ha a több tumorsejt pusztul el, mint amennyi újonnan, szaparodóképes állapotban van jelen. Akkor viszont nyeregpontja 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 a vizsgált probléma szempontjából sok tekintetben ideális. Az optimalizálási feladat során csupán arra van ráhatás, 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 [13]. 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.

2.4. Vizsgált eljárások

Ebben a fejezetben a megerősítéses tanuláshoz kapcsolódó eljárások kerülnek bemutatásra. Azt, hogy konkrétan egy adott MDP-nél hogyan alkalmazható általánosan az adott megoldás. Ezek az eljárások azonos, vagy közel azonos szempontok alapján történtek felkutatásra, megvizsgálásra és összehasonlításra. Ezek a szempontok a következők:

- Eljárás jellemzése:
  - Megemlítésre kerül, hogy mit jelent a neve konkrétan és hogy miért kapta azt a nevet, amit;
  - Egy egyszerű példán keresztül a működésére;
- Jellemezve lesz a működés elvi háttere;
- Említve lesznek az egyes módszerek összehasonlított előnyei és hátrányai.

Ebben a fejezetben a kiindulási pontnak a megerősítéses tanulás alapjai vannak számon tartva, annak alap problémájának egyszerű megoldásától. Ez a felvezetés egészen odáig jut el, hogy milyen új megközelítéseket, hibrid megoldásokat lehet alkalmazni annak érdekében, hogy minél pontosabb és jobb eredményt lehessen kapni, lehessen elérni.

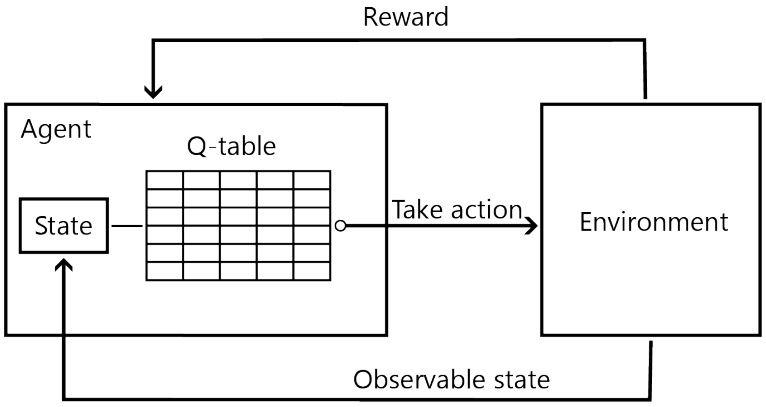
2.4.1. Q-tanulás

A megerősítéses tanítás egyik alap koncepciója a Q-tanulás (*Q-learning*) [14]. 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 ismértelve 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 tudáshalmaz 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* [15] 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 [16].

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 [17, 18, 16].

A Q-tanulás fontos eleme, hogy az egyes lépések után frissíteni szükséges a Q-táblán belüli értékeket. A Q értékek frissítése [14] két egyenlettel írható fel:

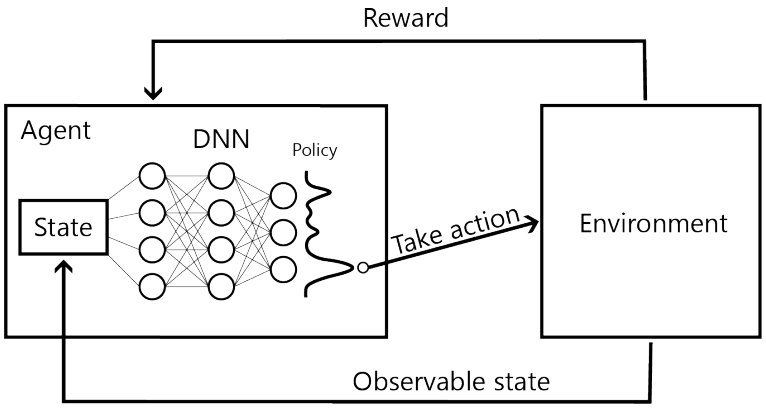
$$y = r + \gamma \cdot \max_a Q(S_{t+1}, a) \tag{2.13}$$

$$Q(s, a) = Q(s, a) + \alpha \cdot (y - Q(s, a)) \tag{2.14}$$

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 következőekben jelentkező jutalmak milyen súllyal (γ) számítsanak bele a végeredménybe. Ehhez hozzáadásra 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 van frissítve. Maguk a változtatható paraméterek lehetővé teszik a szabadabb választás jogát, hiszen ezek segítségével lehet szabályozni többek között azt, hogy mely állapotbeli jutalmak számítsanak bele leginkább az összjutalomba: a korai vagy a késői döntésekkel járó állapot-átmenetekéi. [4]

2.4.2. Deep Q-Network (DQN)

A napjainkban használt és elterjedt megerősítéses tanulási eljárások mind DQN-re épülnek. A DQN [19, 20] 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.

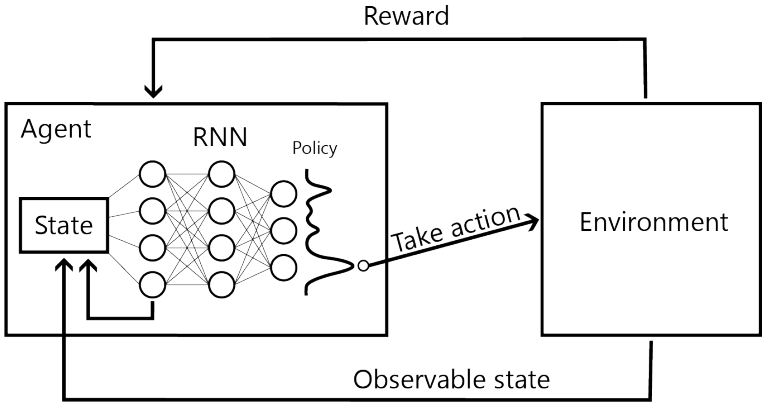
Ennek a megoldásnak az előnyei szemmel láthatóak a sima Q táblás megoldáshoz képest. Mivel itt egy DNN-el (Deep Neural Network-kel) dolgozik az eljárás, a háló képes arra hogy mintákat, összefüggéseket találjon state-ek és döntések (action-ok) között olyan komplexitással, amikre algoritmikailag szinte 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 dobni 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.

2.4.3. Deep Recurrent Q-Network (DRQN)

Mielőtt meghatározásra kerülne, hogy mi is az a DRQN (Deep Recurrent Q-Network) [21, 22], 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) [23] készült megoldás főként.

Egy gyakorlati példán keresztül érzékeltethető a fennálló probléma és az arra felvázolt megoldás. 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 ágens 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 háló van használva, ami képes elraktározni a környezetünkről készített megfigyeléseket.



2.5. ábra. A DRQN egyszerűsített architektúrája

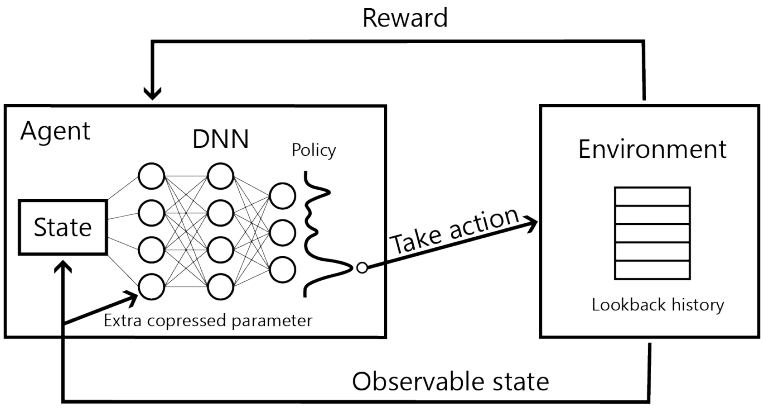
A DRQN segítségével egy adott memória ablakon belül az RNN-nel (Recurrent Neural Network) [24, 25] felvértezett ágens képes még komplexebb és pontosabb döntéseket hozni. Ilyen rekurrens hálózatoknak a használata nem elterjedt és szokványos eljárás a megerősítéses tanulás alkalmazásánál, így rengeteg 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.

2.4.4. 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 világbéli probléma nem MDP, hanem POMDP, azonban nem mindig van lehetőségünk arra, hogy DRQN-t tudjunk használni, erőforrás vagy architektúrális 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 "pillanatképet látva". A DQN-nek nincs információja a múltról 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 kerül zsugorításra. Ez az amit vissza van adva egy kumulált paraméterként a DQN ügynöknek. 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 kumulált tulaj-

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



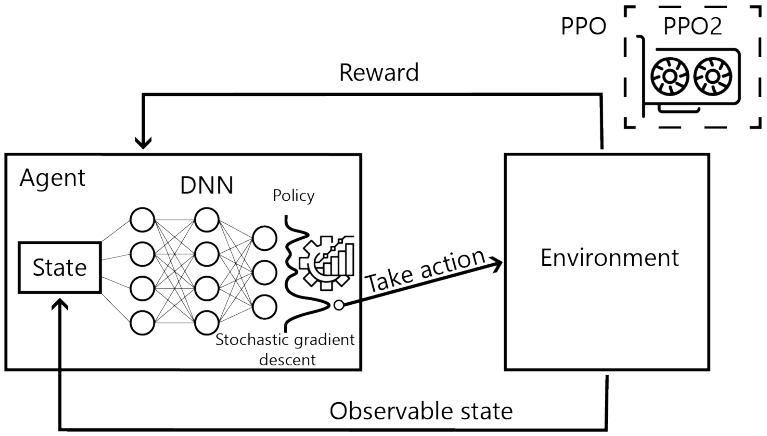
2.6. ábra. A Csúszó ablakos különbözeteken alapuló DQN egyszerűsített architektúrája

Ez azért rendkívül hatásos mivel az ügynök és a neurális háló az utolsó X darab döntésből fakadó paraméterváltozást is fogja érzékelni emiatt az új bevetett, 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.

2.4.5. 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 ugrik a policy megválasztás tekintetében hogy a tényleges megoldás átugródik, átugrásra kerül, és az idáig lévő pontosságot ezzel teljesen elrontjuk. Jóformán 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 pedig 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.

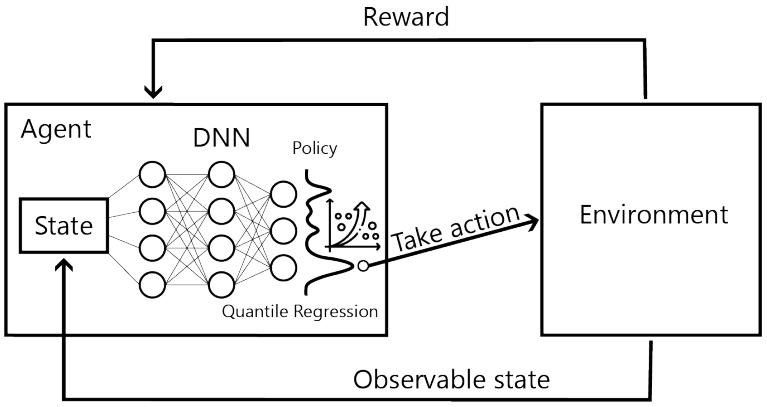
A *PPO* megalkotói hamar rájöttek arra, hogy ezt a megoldást könnyedén lehetne még gyorsabbá tenni. Így született meg ennek a verziónak az implementációja *GPU*-ra az OpenAI által. Ez a verzió vektorizált formában várja a környezeti változókat a sima *PPO* verzióval szemben az elosztottság és multiprocessing végett. Mindez még nagyobb gyorsaság elérésére képes, tesztek szerint háromszoros [26] sebességnövekedést képes produkálni.



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

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

Az eddigi megoldások mindegyike közös volt abban, hogy egy pillanat vagy egy idősor alapján hoz egy döntést [27]. 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*. 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

2.5. Vizsgált jutalomfüggvények

Ebben a fejezetben a vizsgált jutalomfüggvényekről lesz szó. A fentebb megismert jutalomfüggvény fogalma alapján ez az a része a programnak, ami osztályozza a modell döntéseit jóság alapján. A közösen végzett kutatómunkánk (Hörömpő And-

rással), egyik legnagyobb kihívása ennek a jutalomfüggvénynek / függvényeknek a megtalálása, finomra hangolása.

Mindegyik függvényt, eljárást több szempont figyelembevételével vizsgáltuk meg, amik javarészt a következők voltak:

- Mi volt a probléma, amit megoldottak vele, azaz hogy mire is használták pontosan
  - Megnevezésre kerül maga a betegség, mely kezelésének optimalizálására készült
  - Meg lesz adva, hogy sugárterápia, kemoterápia vagy esetlegesen más jellegű kezelést alkalmaztak-e
  - És milyen eredményekkel járt, miket értek el ezzel a függvénnyel
- Milyen matematikai modellre, összefüggésre alapozták
  - A modellek milyen paraméterekből építkeztek, épültek fel
  - A hosszú túlélésre törekedtek, vagy inkább a betegség leküzdésére összpontosítottak
  - Megadásra kerül a függvények felépítése, miszerint hány függvényből, illetve milyen komplexitásúakból álltak
- Ezeket összevetve hogyan tudna a jelen problémán segíteni
  - Maga az elképzelés működőképes-e vagy nem
  - Megállapításra kerül, hogy magukban is megállják a helyüket vagy módosításra van szükség egy-egy helyen
  - Esetlegesen hiányzó paraméterek esetén megoldható-e, hogy ezek a paraméterek valahogy előállíthatóak-e vagy sem.

2.5.1. Padmanabhan-féle jutalomfüggvény

Az első ilyen kutatás, amit górcső alá vettünk, a Regina Padmanabhan [28] által vezetett rákkutatás volt. Számukra a rákkezelés gyógyszer-dózisok formájában, és nem sugárkezelés útján valósult meg. Elsődleges céljuk, hogy a rákot, mint a rendellenes sejtek sokaságát, rendellenes terjedését legyőzzék, mégpedig olyan kezelés kialakításával, melynek segítségével a tumor méretének csökkenése a lehető leghatékonyabban érhető el. Megerősítéses tanítást használtak, amelyhez matematikai modellt használtak fel. Magát a tumor viselkedését differenciálegyenletek formájában fogalmazták meg. Céljuk pedig az volt, hogy ezen egyenletek ismeretében megtalálják az optimális kezelési megoldást.

A jutalomfüggvény az alábbiak szerint van definiálva:

$$r_{k+1} = \begin{cases} \frac{e(kT)-e((k+1)T)}{e(kT)}, e((k+1)T) < e(kT) \\ 0, e((k+1)T) \geq e(kT) \end{cases} \tag{2.15}$$

T: mintavételi idő
kT : aktuális időpillanat
(k+1)T: következő időpillanat

e(t): az error, hiba mértéke

Az error kategóriákat figyelembe véve különböző módon kell kezelni a betegeket, ha a páciens idős, fiatal, avagy éppen várandós. Ugyanis egy fiatal szervezetnél minél előbb le akarják csökkenteni a tumor méretét, mivel náluk gyorsabban terjed a betegség. Ezen error értékek különböző tartományokra vannak osztva, melyek az s_k állapottól függően mindig más-más tartományba esnek. Megemlítendő, hogy a kutatók a várandós nők és a fiatalok error értékét egy tartomány alá veszik, míg az idősek külön csoportot alkotnak.

Az error function ezen matematikai egyenlet segítségével írható le:

$$e(t) = \beta x_2(t) + (1 - \beta)[1 - x_1(t)] \tag{2.16}$$

$x_2(t)$ a t-edik időpillanatban jelen lévő rákos sejtek mennyisége
β súlyozási tényező, mely a kutatásban kezdetben 0.9 értékű

A függvények alkalmazhatóságával kapcsolatban vegyes gondolatok merültek fel bennünk. Ugyanis egyértelműnek bizonyult, hogy hasznosítható ezen jutalomfüggvényből paraméter egy bizonyos szempontból. Megvizsgálható ugyanis a tény, hogy valamilyen módon lehet-e következtetni a rákos sejtek mennyiségéből a rákos sejt térfogatára, illetve visszafelé, a térfogatból mennyiségre. Amennyiben igen, és sikerül egyfajta megközelítő becslést kialakítani, úgy a jutalomfüggvény implementálása megpróbálható. Az error függvény implementálása ezeken felül kihívást jelenthet, hogy a vizsgált kísérleti állatokon, egereken hogyan lehet azt megállapítani, hogy a kezelt egér éppen fiatal vagy idősebb. Egy adott módszer kitalálása mellett ugyanakkor helye lehet ennek is, így a Padmanabhan-féle megközelítés egészének is.

2.5.2. Yazdjerdi-féle jutalomfüggvény

Yazdjerdi [29] és más kutatók által vizsgált „Anti-angiogenic" terápia egy molekuláris szintű terápia, ami kemoterápiával és radioterápiával tarttatja egy bizonyos szinten a rákos daganatot (de nem eliminálja azt teljesen). Ennek is a célja az volt, hogy mikrodózisokkal nagyon finoman szabályozott kezelést kapjon a páciens annak érdekében, hogy az élete minél inkább meg legyen hosszabbítva. A szóban forgó reward function amit alapul vettek:

$$r(k+1) = \begin{cases} \frac{e(k)-e(k+1)}{e(k)}, e(k+1) < e(k) \\ 0, e(k+1) \geq e(k) \end{cases} \tag{2.17}$$

$$e(k) = |X_1(kT) - X_d| \tag{2.18}$$

T: mintavételi idő
X_d: kívánt tumor méret (mm^3)
X_t: tumor méret (mm^3)

Ez a függvény egy másik függvényt használ fel, amit hiba függvényként neveztek el. Ez a hiba függvény az elvárt tumor mérettől való távolságot adja vissza abszolútértékben. A jutalomfüggvényben pedig ez a hiba van felhasználva és van összehasonlítva a jelenlegi időpillanatban lévő és az ezt követő időpillanatbéli hibával.

Ebből a működésből levonható pár konklúzió:

- Nincs konkrétan számontartva, hogy a páciens mennyi ideig élt
- Nincs jutalmazva az az esetet, ha sikerült mégiscsak felszámolni a tumort
- Csak a tumor méretét veszi figyelembe, kemoterápia / sugárterápia dózisokat pedig nem, ami szintén nem elhanyagolható

Ezt az egész függvényt könnyedén ráilleszthet a taglalt problémára, problémánkra, hiszen van hozzá megfelelő paraméterünk, hogy meg lehessen valósítani. A fentiekből következik azonban, hogy valamiféle változtatásra szükségünk lenne, hogy figyelembe vegyük, ha nem is mindenféleképpen a túlélési időt, de legalább a vérben lévő kezelő anyag koncentrátumát, hiszen a kemoterápiát vagy sugárterápiát sem adagolhatjuk túl. Ezen felül, ahogy említésre került, az is egy opció lenne, ha a túlélési időt is valahogy bele lehetne vinni ebbe a függvénybe.

2.5.3. Hassani-féle jutalomfüggvény

Hassani [30] és társa is majdnem pont ugyan azt a célt tűzte ki, mint ami a kutatás célja is volt, mégpedig hogyan lehet minimalizálni a tumor méretét a lehető legkevesebb beadott gyógyszer mellett. Előre haladott stádiumú rákos emberek kezelését optimalizálták, akik a kemoterápiát infúzión keresztül kapták. Ők nem tértek ki a rák pontos típusára, így nem kell attól tartani, hogy a megoldásuk és a jutalom függvény rák-specifikus lenne. Véleményük szerint a megerősítéses tanuláson alapuló rákkezelés optimalizálás a meglévő megoldások következő lépcsője, és kiválóan használható egyedi karakterisztikákkal rendelkező páciensek pontosabb kezelésére.

A jutalom függvény, amit használtak, meglepően egyszerű:

$$r_{k+1} = \log\left(\frac{T_k}{T_{k+1}}\right) \tag{2.19}$$

T_t: tumor sejt populáció (sejt szám)

A tumor sejt populáció változásának a hányadosát majd ennek a tízes alapú logaritmusát vették. Tehát a fókuszban náluk a tumor sejt populáció állt. Ebből egy nagyon egyszerű összefüggés azonnal észrevehető:

- ha $T_k > T_{k+1}$ akkor a jutalom pozitív
- ha $T_k < T_{k+1}$ akkor a jutalom negatív

Magyarul, ha nő a tumor sejt populáció, akkor negatívan jutalmazunk, míg ha csökken, akkor pedig pozitív érték kerül megállapításra. A megerősítéses tanuláson alapuló megoldásuk "action"-jei diszkrét normalizált ellenszer adagolásból álltak (0.1, ... 1) közötti értékekkel.

A fentebb taglalt függvényt viszonylag egyszerűen be lehet illeszteni a feladatba abban az esetben, ha egy egyszerűsítéssel élünk, és az kerül megállapításra, hogy a tumor sejt populáció egyenlő a tumor méretével ebben az esetben. Ez a módosítás szükséges, mivel nem lehet megmérni a tumor sejt számát, legfeljebb becsülni viszonylag nagy hibával a tumor méretéből, így nyugodtan lehet dolgozni a tumor méretével egyből. Ez mind szép és jó, viszont a másik legnagyobb problémát, a gyógyszer koncentrációt, nem veszi figyelembe. Ez így önmagában azt eredményezné, hogy rengeteg gyógyszert adna az ágens a páciensnek, amibe bele is halna. A

konklúzió az, hogy ezt a megoldást lehetne alkalmazni némi módosítással, miszerint figyelembe van véve a jutalom kiszámításánál a vérben lévő ellenanyag-koncentráció is.

2.5.4. Zhao-féle jutalomfüggvény

A Zhao és társai [31] által kutatott terület szintúgy kapcsolódik a vizsgált témához, számukra is a cél az életet közvetlenül veszélyeztető betegségek, mint például a rák elleni küzdelem, melyhez a megerősítéses tanítást használták fel. Igyekeztek a lehető legjobban optimalizált megoldást, stratégiát találni, melyekhez klinikai adatokat használtak fel közvetlenül mindamellett, hogy a megfelelően pontos matematikai modell megtalálására kisebb hangsúlyt fektettek. Megjegyzendő tanulság részükről, hogy „nagy potenciál rejlik" a megerősítéses tanításnak a gyógyászatban történő hasznosítására, amellett is, hogy a rendszer „folyamatosan tanul", még a „késleltetett hatások" figyelembe vételét is, még akkor is, ha az egyes „cselekvések és eredmények közti kapcsolat nem teljesen ismert".

Az általuk használt jutalomfüggvények az alábbiak szerint definiálhatóak:

$$R_{t,1}(D_t, W_{t+1}, M_{t+1}) = -60, \text{ amennyiben a páciens meghalt} \tag{2.20}$$

$$R_{t,2}(W_t, D_t, W_{t+1}) = \begin{cases} 5 & W_{t+1} - W_t \leqslant -0.5 \\ -5 & W_{t+1} - W_t \geqslant 0.5 \\ 0 & \text{egyéb esetben} \end{cases} \tag{2.21}$$

$$R_{t,3}(M_t, D_t, M_{t+1}) = \begin{cases} 15 & M_{t+1} = -0.5 \\ 5 & M_{t+1} - M_t \leqslant 0.5, M_{t+1} \neq 0 \\ -5 & M_{t+1} - M_t \geqslant 0.5 \\ 0 & \text{egyéb esetben,} \end{cases} \tag{2.22}$$

ahol $R_{t,1}(D_t, W_{t+1}, M_{t+1})$ a túlélési mutatóból, $R_{t,2}(W_t, D_t, W_{t+1})$ a betegen észlelhető *wellness*, "jólléti" hatásból, $R_{t,3}(M_t, D_t, M_{t+1})$ a tumor méretéből adódik. Ezek három részösszeg adja a végső jóságértéket, melyekben W_t az adott időpillanatbeli wellnessértéket mutatja, M_t az adott időpillanat-beli tumorméretet mutatja, D_t pedig az adott időpillanat-beli dózismennyiséget mutatja.

Ahogy azt a rész-jutalomfüggvények is mutatják, amennyiben egy adott határértéken túlmutatnak az egyes paraméterváltozások, úgy azt annak megfelelően pluszjutalommal illeti, ha az a betegség gyógyulása szempontjából pozitív hatással, illetve negatívjutalommal illeti, ha az a gyógyulás szempontjából negatív hatással van a páciensre. Megemlítendő tényező, hogy amennyiben a következő, *t+1*-edik időpillanatban a tumor mérete beáll egy adott *-0.5*-ös értékre, úgy abban az esetben eredményeződik a harmadik rész-jóságfüggvényben a legmagasabb jutalom, hogy ezt maximalizálni lehessen, fontos mellé, hogy a páciens *wellness*-e minél jobb legyen, azaz egy adott időpillanatból ezen *t+1*-edikbe lépve ne változzon negatív irányba radikálisan.

Kapcsolópontja ezen megközelítésnek illetve a diplomamunka témájának, hogy ezen függvény és megközelítés számára is a túlélés a legfontosabb szempont, melyben döntő szerepe van a besugárzott dózisnak valamint a tumor méretének is. A függvény lehetőséget ad súlyozásra is, melyik számít az összeredménybe és -jutalomba nagyobb mértékben. Ezen értékek számunkra is adottak és ismertek, így figyelembe vehető, alkalmazható függvény lehet.

2.5.5. Yauney-féle jutalomfüggvény

A Yauney [32] által vezetett kutatás szintén rávilágított arra, hogy megfelelő, jól meghatározott jutalomfüggvény nélkül a megerősítéses tanítás nem működhet kellően hatékonyan olyan területeken, mint az egészségügy és a gyógyítás. Holott ennek meghatározásával klinikai teszteket megelőző dózisadagolásokat lehet meghatározni, melyek később megerősített eredményével akár tényleges gyógyításra lehet fordítani. Fő célpontja, illetve jutalomfüggvényének alapja, hogy az átlagos tumor átmérő (mean tumor diameter – MTD) a lehető legkisebb legyen, miközben a páciens sugárkezelést kap. Természetesen nem minden áron történő csökkentésről van szó, figyelembe veendő a páciensnek adott dózismennyiség is, a túlzottan nagy mennyiség ugyanis károsító, mintsem gyógyító hatással van a kezelt betegre.

A Yauney-féle jutalomfüggvény az alábbiak szerint definiáltható:

$$R = c(MTD_t - MTD_{t'}) - penalty \cdot concentration \tag{2.23}$$

c: egy konstans érték, a gyakorlatban 1-es értékkel, azaz 1-es súllyal
MTD_t: a t-edik (előző) pillanatban vett tumorátmérő
$MTD_{t'}$: a *t+1*-edik (jelenlegi) pillanatban vett tumorátmérő
penalty: a büntetés mértéke, jellemzően 1 és 5 közötti konstans érték, ahol az érték növelésével arányosan növekszik a büntetés mértéke is
concentration: a t-edik pillanatban adott dózis mennyiségét mutatja (0, ha abban a pillantban nem történt adagolás)

A jutalomfüggvényről megállapítható, hogy törekszik a minél kisebb tumorátmérő elérésére, ugyanakkor figyelembe veszi negatív súllyal, ha dózist kapott a kezelt páciens. Adott pillanatban optimális megoldásnak tűnik, ha a tumorméret csökken egy adott mértékben, lehetőség szerint minél jobban, amellett, hogy nem történt dózisadagolás.

Az utolsó epizódban történő jutalomfüggvény-érték a végső eredményt jobban jellemzi, hiszen abban megjelenik, mekkora tumorméret-csökkenést sikerült elérni a folyamat során.

$$R_{final} = c_{final}(MTD_{initial} - MTD_{final}) \tag{2.24}$$

c_{final}: konstans érték, a gyakorlatban 10-es értékkel
$MTD_{initial}$: a kezelés előtt megfigyelt tumorátmérő
MTD_{final}: a kezelés befejeztével megfigyelt tumorátmérő

Megállapítható, hogy jelen diplomamunkában is szerepe lehet ennek a jutalomfüggvénynek, illetve annak implementálásának. Meg lehet figyelni ugyanis a tumor méretét, közvetlen paraméter tartozik hozzá. Továbbá lehet azt is ellenőrizni, mikor történt a páciens számára dózis adagolás. Mivel mindkettő érték figyelembe vehető és lehet vele dolgozni, felhasználható a függvény készítéséhez, így megfigyelendő, milyen hatása lehet a problémánra. A paraméterek változtathatóak, így azok finomhangolásával az eredmények javulása várható.

2.5.6. Jalalimanesh-féle jutalomfüggvény

A Jalalimanesh [33] vezette kutatás alap, ugyanakkor helyt álló kijelentése, hogy a rákgyógyítás egy hatásos ellenszere a sugárkezelés, mely kezelés megannyi kezelt páciensnél effektívnek bizonyul. Ugyanakkor szükséges egyfajta optimalizálás annak érdekében, hogy a sugárkezelés hatására az egészségre gyakorolt negatív hatások minél kisebbek, kevésbé számottevőbbek legyenek. Itt jön képbe számukra is a megerősítéses tanítás alapelve, megbecsülni azt, mikor és milyen módon lenne legjobb a pácienst kezelés alá vetni, mikor kellene újabb sugárkezelést adni számára.

A Jalalimanesh-féle jutalomfüggvény az alábbiak szerint definiálható:

$$r = N_C - N_H - \alpha \cdot A_I \cdot I_R \tag{2.25}$$

N_C: a kiölt tumorsejtek mennyiségét mutatja meg
N_H: a kiölt egészséges sejtek mennyiségét mutatja meg
α: konstans, súlyérték
A_I: a sugárkezelés alá esett területet mutatja meg
I_R: a sugár erősségét mutatja meg, 3 értékkel fordul elő: 2.5 Gy (gyenge), 3 Gy (normál) és 3.5 Gy (intenzív, erős)

A jutalomfüggvény szempontjából kiindulási érték, hogy adott epizód során mennyi rákos sejtet sikerült kiölni. Ugyanakkor figyelembe veszi azt is, hogy ezen sugárkezelés hatására mennyi egészséges sejtet ölt ki ezek mellett, továbbá, szintúgy negatív súllyal, szerepel az is, hogy a kezelés során mekkora terület volt kitéve sugárkezelésnek, arányosan a sugárzás intenzitásával. A büntetés összességében nem tekinti elfogadottnak, ha nagy terület van sugárkezelés alá véve nagy sugárzással, mivel annak már egészségre gyakorolt hatása inkább hátránnyal járna összességében. Optimális eset, ha minél több rákos sejtet sikerül kiölni a legkevesebb egészséges sejt elvesztésével, valamint minél alacsonyabb sugárerősség és érintett terület figyelembe vételével.

A jutalomfüggvénynek nehezen van helye a probléma előrehaladásában. Jelen esetben ugyanis hiányoznak olyan paraméterek, melyek a Jalalimanesh-féle jutalomfüggvényben felhasználásra kerülnek, ilyen például a sugárkezelt terület vagy éppen sugárzás erőssége. Az is nehezen lenne meghatározható, hogy egy tanítási epizód alatt mennyi egészséges sejtet sikerült kiölni.

2.5.7. Humphrey-féle jutalomfüggvény

Humphrey munkájában Q-learning [34], illetve megerősítéses tanulási algoritmusokat használt fel annak érdekében, hogy olyan megoldást határozzon meg, melynek segítségével folyamatos kezelést lehet adni egy adott típusból rákkal küzdő betegek számára. A megközelítése értelmében Q-függvényeket becsült meg többféle szerkezetet figyelembe véve, kipróbálva. Figyelembe vette azt is, hogy nem minden beteg reagál ugyanarra a kezelésre ugyanolyan módon, így egy általános megoldásra törekedett.

A Humphrey-féle jutalomfüggvény a következő, *t+1*-edik időpillanat tekintetében az alábbiak szerint írható le:

$$R_{t+1} = \begin{cases} \log(S_t + \sum_{j=0}^{t} j) & S_t \leq 1 \text{ vagy } t = 2 \\ 0 & \text{egyébként} \end{cases} \tag{2.26}$$

Maga a jutalomfüggvény sajátos módján összetettnek tekinthető. Az S_t nem más, mint egy páciens t időpillanatig történő túlélési rátáját mutatja. Abban az esetben, ha 1-nél nem nagyobb ez az érték, akkor azt jelenti, hogy a beteg nem élt túl még egy hónapot. Amennyiben a beteg túlélte az elvárt kezelési idő végét jelentő 2 hónapot (azaz $t = 2$), akkor a jutalom logaritmikus alapon számítódik ki a szimulált időből, valamint az eltelt hónapok mennyiségéből adódóan.

A túlélési időket az alábbi exponenciális eloszlásból generálták.

$$S_t \sim Exp(\lambda_t(M_{t+1}, W_{t+1})) = Exp(exp(-5.5 + W_{t+1} + 1.2M_{t+1} + 0.75W_{t+1}M_{t+1})) \tag{2.27}$$

M_{t+1} a következő hónapra vonatkozó tumor tömegét adja meg W_{t+1} a következő hónapra vonatkozó toxicitást adja meg D_t a t-edik hónapban lévő dózist adja meg λ_t pedig a túlélési eloszlást adja meg az (M_{t+1}, W_{t+1}) figyelembe vételével.

A túlélést befolyásoló tényezők ennél a kutatásnál határozottan kijelenthetők, figyelembe veszi a tumor méretét, a szervezetben jelen lévő toxicitás mértékét és az adagolt dózist. Maga a toxicitás nem jelent mást, mint az emberi életre mért negatív *jólléti* és minőségi hatásai, amitől "nem érzi túl jól magát" a páciens. A toxicitás és a tumortömeg egyenletek segítségével ugyanúgy meghatározhatóak, mint ahogy az volt a túlélési ráta esetén:

$$W_t^* = 0.1M_{t-1} + 1.2(D_{t-1} - 0.5) + W_{t-1} \tag{2.28}$$

$$W_t = \begin{cases} W_t^* & W_t^* > 0 \\ 0 & W_t^* \leq 0 \end{cases} \tag{2.29}$$

$$M_t^* = [0.15W_{t-1} - 1.2(D_{t-1} - 0.5) + M_{t-1}]I(M_{t-1} > 0) \tag{2.30}$$

$$M_t = \begin{cases} M_t^* & M_t^* > 0 \\ 0 & M_t^* \leq 0 \end{cases} \tag{2.31}$$

Általános megállapítás a szerző szerint, hogy az alacsony szinten tartott egészségi szint (azaz a magasabb W_{t-1} nagyobb tumornövekedést eredményez, míg az magasabb dózisszint (azaz a D_{t-1} adott szint felett tartása) csökkenést eredményez. Persze, ha ez az érték nem éri el a küszöbnek számító *0.5*-ös értéket, szintén tumornövekedést okoz. A tumorméret természetesen nemnegatív értéket vehet csak fel.

A tumorméret és toxicitás jellemzésére szolgáló eloszlás is megfogalmazásra került, az alábbi módon:

$$\lambda_t(M_{t+1}, W_{t+1}) = exp(-5.5 + W_{t+1} + 1.2M_{t+1} + 0.75W_{t+1}M_{t+1}) \tag{2.32}$$

Összességében elmondható a megoldásról, hogy noha nem tűnik kivitelezhetetlennek, nem-implementálhatónak, annak toxicitásra vonatkozó kijelentései mellett alkalmazására feltehetően nem kerülhet sor. A páciensek *jóllétét* sajnos nem lehet megfelelő módon megmérni, nem áll rendelkezésre mint paraméter, így használni sem lehet. Ugyanakkor, több egyenletben is megjelenik mint számottevő faktor, így annak hanyagolása egy merőben más megoldást eredményezne.

2.5.8. Tseng-féle jutalomfüggvény

A Tseng [35] és csapata által vizsgált terület és az alkalmazott paraméterek nagy különbséget mutatnak az általánosan megfigyelt eljárásokhoz képest. Alapelképzelésük szerint, noha mély megerősítéses tanítást alkalmaztak, olyan kezelési terveket dolgoztak fel, melyek már *lezajlottak*, azaz nem egy, a jelenben és jövőben nem létező megoldást *találna ki* a rendszer, hanem a megfelelő hiperparaméterek finomhangolásával, a megfigyelt információ segítségével alakítják azt ki. Elsődlegesen tűdőrákhoz, nevezett *non-small cell lung cancer (NSCLC)* kutattak, ahol célként a tumor irányíthatóság maximalizálását tűzték ki, mégpedig a besugárzott *radiation pneumonitis grade 2 (RP2)* minimális szinten tartása mellett. Azaz, hogy minél inkább kordában legyen tartva a tumor és annak mérete, a sugárzás minimálisan tartása mellett.

A Tseng-féle jutalomfüggvény az alábbiakat fogalmazza meg egy adott állapot jutalmaként:

$$R(s) = \frac{1}{2}\sqrt{Prob(LC|s)} \cdot (1 - \omega \cdot Prob(RP2|s)) \cdot (1 + sgn(17,2\% - Prob(RP2|s))) \tag{2.33}$$

R(s) az adott állapot jutalmának értéke
Prob(LC|s) az adott állapot szerinti tumor irányíthatóság feltételes valószínűsége
Prob(RP2|s) az adott állapot szerinti besugárzott dózis feltételes valószínűsége
ω a súly, jelen esetben 0.8 értékkel

A szemléltetett jutalomfüggvény tagadhatatlanul egy igen jó függvény, ugyanakkor sajnos észrevehető benne egy olyan probléma, hogy nem feltétlenül tud kellő mértékben illeszkedni. Tumor irányíthatóságot ugyanis semmilyen körülmények között nem lehetséges mérni, csupán tumor méretet. További fontos különbség, hogy ezen megoldás elsődlegesen tüdőrák kezelésére lett optimalizálva, a megadott típusú sugárzás alkalmazásával, a jelen dolgozatban taglalt megfigyelések azonban nem elsődlegesen erre irányulnak, hanem a szervezeten belül máshol elhelyezkedő ráktípusra. Ugyanakkor levonható az a következtetés ebből is, amelyet a többi függvényből már számos helyen lehetett látni, hogy a bevitt sugárdózis minimalizálása mellett igyekszik a lehető legjobb eredményt elérni.

2.5.9. Petousis-féle jutalomfüggvény

Petousis [36] és néhány társa is használt megerősítéses tanuláson alapuló megoldást a rákkezelésre. Ez több szempontból is különleges mivel ők az úgynevezett IRL (Inverse Reinforcement Learning) technikát alkalmazták amivel gyakorlatilag jutalomfüggvényt generáltak az emlő és tüdőrák méréseihez tartozó POMDP-khez (Partially Observable Markov Decision Process). Ezeket szakértők által mért és meghozott döntések alapján az emlő és tüdőrák kezelésében alkalmazták, melyek során a kezeléseken sugárterápiát alkalmaztak.

A megoldás különlegességéből fakad az alábbi, viszonylag egyszerűnek tűnő jutalom függvény:

$$r(\tau, \theta) = \theta^T f_\tau = \sum_{s \in \tau} \theta^T f_s \tag{2.34}$$

f_s: mért tulajdonságok vektora
θ: súlyok vektora

τ: mért paraméterek

Ez a jutalomfüggvény tulajdonképpen a mérendő paramétereket összegzi, súlyok alapján. Ezen megoldás különlegessége a súly paraméterek meghatározásában rejlik.

Összességében ez a megvalósítás teljesen eltér a szokványos megerősítéses tanuláson alapuló kezelés optimalizálástól. Így ezt a jutalom függvényt a megfelelő súlyok ismerete nélkül lehetetlen használni és implementálni. Ez a koncepció azonban megérne egy külön kutatást.

2.6. Konklúzió

A modellek tekintetében tumorokra egyértelműen látszik az az állítás, miszerint vannak olyan matematikai modellek, amik bizonyos paramétereket nem vagy nem úgy vesznek figyelembe, ahogy az kívánatos, túl egyszerűsítettek vagy túl komplexek és összetettek ahhoz, hogy kellően hatékonyan lehessen implementálni és integrálni egy megerősítéses tanuláson alapuló rendszerbe. A Drexler-féle modell a valóságban is könnyen mérhető paramétereket tartalmaz, ezeket egyrészt relatív könnyű mérni, másrészt egy RL megoldás is fel tud használni bemenetként. Nem elhanyagolható ok a választás mellett az sem, hogy az Óbudai Egyetem Kiberorvosi Kompetencia Központjában zajló kutatáshoz később így jól kapcsolható a munka ezen része.

2.6.1. Eljárás-orientált megközelítés konklúziója

A megerősítéses tanulás tekintetében az eljárások közül egyértelmű, hogy a sima Q-Learning sajnos nem elég robosztus ahhoz, hogy alkalmazásra kerüljön sor. Ezzel szemben alkalmazni lehet az alábbi eljárásokat azok megfelelő komplexitása miatt:

- Deep Q-Network (DQN)
- Csúszó ablakos különbözeten alapuló Q-Network (Compressed Lookback Difference DQN)
- Deep Recurrent Q-Network (DRQN)
- Proximal Policy Optimization 2 Deep Q-Network (PPO2-DQN)
- Quantile Regression Deep Q-Network (QR-DQN)

Mindegyik megoldás másban jó, mivel más az architektúrájuk, ezért erre egyértelmű válasz nincs, hogy melyik a tökéletes. Minden megközelítést érdemes kipróbálni, majd pedig azokat adott szempontok szerint összehasonlítani, hogy erre egy közelítő választ lehessen adni. Az eljárások megfigyelése célszerűen a DQN alapú megoldással kezdődne, mivel minden eljárás szempontjából megfelelő *baseline*-t, kiindulási és összehasonlítási alapot tud adni. A csúszó ablakos különbözetű DQN ennek egy továbbfejlesztett változata lenne, ami *rekurrens-szerű* viselkedést igyekszik elérni a múltban mért adatok jelenben történő felhasználása szempontjából, emellett gyorsabb mint a *DRQN*. Ennek egy továbbfejlesztése, state-of-the-art megközelítése magától adódóan figyelembe vehető a DRQN hálózat, mivel az teljesen rekurrens és a múltbéli adatok teljes spektrumával képes dolgozni. A *policy* optimalitálására megfelelőnek bizonyulhat a *PPO2*-es hálózat, melynek párja, a *PPO*

amiatt nem kerül részletesen tesztelésre, mivel nem GPU-optimalizált, így igen lassú tanítási folyamatot eredményez. A komplex tendenciák figyelembe vétele miatt a *QRDQN* is egy jó megoldás lehet nem csak a *DRQN*.

2.6.2. Jutalomfüggvény-orientált megközelítés konklúziója

A jutalomfüggvények felől szintén látszik az, hogy vannak bizonyos paraméterek, amiket lehet mérni és vizsgálni a Drexler modellből (tumor térfogat, beadott gyógyszer mennyiség) és vannak, amiket például nem. Ezek a megkötések szinte egyértelműen definiálják azokat a jutalomfüggvényeket, amik kipróbálhatóak és érdemesek is erre. Ezek a jutalomfüggvények a következők:

- Yazdjerdi-féle jutalomfüggvény
- Hassani-féle jutalomfüggvény
- Zhao-féle jutalomfüggvény
- Yauney-féle jutalomfüggvény

Némelyiket át kell alakítani, módosítani kell egy kicsit ahhoz, hogy használható vagy jobban használható megoldást adjon.

3. fejezet

Módszertan

A következőekben össze lesz foglalva, hogy a diplomamunka pontosan milyen részekből tevődik össze, kitérve természetesen magára az implementációra is. Említésre kerülnek az alkalmazott, irodalomkutatásból következtetett eljárások, továbbá konkrét forráskóddal demonstrálva lesz a rendszer jelenlegi állapota. A következőekben össze lesz foglalva, hogy a kutatás pontosan milyen részekből tevődik össze, kitérve természetesen magára az implementációra is. Említésre kerülnek az alkalmazott, irodalomkutatásból következtetett eljárások, továbbá konkrét forráskóddal lesz demostrálva a rendszer jelenlegi állapotát.

Fontos kiemelni itt azt hogy az itt taglalt módszertanok a kutatási munka egyik része. A módszertanok másik része kutatótársam Hörömpő András diplomamunkájában található meg.

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 említésre is került már, hogy meg kell valósítani 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 alapból *pythonban* készült, így adta magát az hogy olyan csomagok legyenek keresve, olyan mély megerősítéses tanulás eljárások, amik ebbe az architektúrába illeszkednek. Ezen felül a mély tanuláshoz *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 amik az

implementáció során használva lettek:

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

Az alábbi állítások tehetőek, 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. Az világos, ahogy mindenhol hogy a magasabb szint limitációval, limitációkkal jár, és itt sincs másképp. Többek között a komplexebb, rekurrens változatú hálózati felépítéseket megerősítéses tanulás tekintetében a jelen dolgozat problémájára nem kezeli megfelelően jól, emiatt ilyen típusú hálózatokat nem lehet alkalmazni ezen csomagon keresztül.

Emiatt is indult el a kutatás 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.

A konklúzió az, hogy mindkettő megoldásnak megvan a maga szerepe és helye a maguk megfelelő érveivel. Amiatt, hogy a kutatást minél szélesebb körben el lehessen végezni, mindkettő kipróbálásra, alkalmazásra és a tanulási folyamat végén tesztelésre is kerül.

3.1.1. Használt modellek bemutatása

A környezet egy igen fontos része az annak megfelelő modell is, amelyen a megerősítéses tanulás elvégezhető. Ahogy az irodalomkutatás végén meg lett fogalmazva, elsődlegesen a legfrissebbnek tartott Drexler modellt lesz alapul véve. Ehhez kétféle modellimplementáció is figyelembe lett véve. 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 mennyiségből) 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 *compartment*-ben mért dózis mennyisége milligramm/kilogrammban;

- a külső, perifériku*s compartment*-ben mért dózis mennyisége 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 került megalkotásra. Jelen dolgozatban saját munkát jelentenek a jutalomfüggvények és megerősítéses tanulási eljárások irodalmának áttekintése, vizsgálata, implementációja és a különböző megközelítésekből származó eredmények kiértékelése. 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. Legelőször a Drexler Environment implementációja, majd a csúszó ablakos különbözeteken alapuló Drexler Environment. Itt fókuszáltan és kiemelve ez utóbbi sajátosságait. Ezután egy pár implementált jutalomfüggvény bemutatása következik, amit az egészet összefogó futtató környezet bemutatása fog zárni.

3.2.1. Környezetek implementációja

Ha megnézzük a Diszkrét Drexler Environment kódját, akkor jól látszik, hogy maga a környezet implementálja az *OpenAI Gym Env* ősosztályából 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 modell kerül felhasználásra
- *reward*: milyen típusú jutalomfüggvényt kerül felhasználásra

```
class DiscreteDrexlerEnv(Env):
    def __init__(self, action_num=2, dose_unit=8, max_episode_length=50,
                 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
        if model == "twocompartment":
            self.dm = DrexlerTwoCompartmentModel(
                max_episode_length=max_episode_length)
        elif model == "original":
            self.dm = DrexlerModel(max_episode_length=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]),
                                     np.array([self.dm.volume_lim,
                                               self.dm.dose_lim]),
                                     dtype=np.float32)
        self.action_space = Discrete(self.action_num)
        self.reset()
```

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 az implementált rendszert a sok fajta kísérlet kivitelezésére. A példakódból jól látszik, hogy egy ősosztályon keresztül bármilyen tetszőleges jutalomfüggvény osztályt is meg lehet adni. Ami még kiemelendő 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 kerül felhasználásra. 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* ősosztály miatt)

3.2.2. Dosemax és terminálás implementációja

A dosemax és a terminálás részletes implementációját kutatótársam munkájában lehet részletesebben olvasni. Röviden egy komplex feltétel alapján dönt az ágens hogy mikor terminálja a beteget. Ebben a feltételben szerepel a *dose max* is, ami a vérben lévő maximális ellenanyag koncentrációt tartalmazza.

3.2.3. Diszkrét Drexler Environment változatának implementációja - Csúszó ablakos különbözeteken alapuló

Megfigyelve a Diszkrét Drexler Environment Csúszó ablakos különbözeteken alapuló implementációját észrevehető, hogy az imént bemutatott osztályból származik, ezt valósítja meg, konstruktorában kiegészülve egy *lookback* értékkel, ami azt adja meg, hogy meddig képes visszanézni, meddig képes emlékezni a környezet a múltbéli, megfigyelhető paraméterekre. A megfigyelhető paraméterek száma, amit a szintén *observations(self)* tulajdonság ad vissza, kiegészült 1-2 új dologgal, így összesen a következő paramétereket vagyunk képesek megfigyelni a környezetről:

- *tumor_volume*: tumor mérete
- *drug_level*: ellenanyag szintje a vérben
- *tumor_volume_diff*: tumor méretének különbsége
- *drug_level_diff*: ellenanyag mennyiségének különbsége
- *dose_sum*: összessen mennyi ellenanyagot adtunk be idáig (nehogy túladagoljuk stb.)

A konkrét implementáció, amelynél a különbségek megfigyelhetőek:

```
def observations(self):
        return (self.dm.tumor_volume, self.dm.drug_level,
                self.tumor_volume_diff, self.drug_level_diff,
                self.dose_sum)

def tumor_volume_diff(self):
        if (len(self.dm.history['tumor_volume']) < self.lookback):
                return 0
        return self.dm.tumor_volume -
                self.dm.history['tumor_volume'][-self.lookback]

def drug_level_diff(self):
        if (len(self.dm.history['drug_level']) < self.lookback):
                return 0
        return self.dm.drug_level -
                self.dm.history['drug_level'][-self.lookback]

def dose_sum(self):
        if (len(self.dm.history['actions']) == 0):
                return 0
        return sum(self.dm.history['actions'])

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

2. Listing. A *csúszó ablakos különbözetű* modellimplementáció részei

A *tumor_volume_diff(self)* és a *drug_level_diff(self)* azt az értéket adja vissza, hogy az aktuális ellenanyag és tumor mérete mennyivel tér el attól az értéktől ameddig vissza tudunk tekinteni a múltba. A *dose_sum(self)* pedig összegzi azt, hogy hányszor adtunk be ellenanyagot.

3.2.4. Eljárások implementációja jellegük szerint - keras alapú megközelítés

A *keras* alapú megközelítésről, kutatótársam munkájában lehet részletesen olvasni. Összességében ezt a megközelítést az jellemzi hogy szofisztikáltabb az eredmény kezelése egy *callback*-en keresztül, illetve fókuszában a neurális háló konfigurálása és ezen háló az ágensnek való átadása van.

3.2.5. Eljárások implementációja jellegük szerint - torch alapú megközelítés

A feljebb 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)
```

3. 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 példényosításánál 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.

```
# 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)
```

4. Listing. A *Torch* alapú tesztelés és adatgyüjtés

A *torch* esetében sajnos azoban külön tesztelő és adatmentő metódust kell implementálni, aminek az eredményeit a szintén ebben a kutatásban alkotott 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.

3.2.6. Jutalomfüggvények implementációja

Összesen 4 jutalomfüggvény lett implementálva. Ezek közül az a egy lesz itt bemutatva egy pedig kutatótársam munkájában. Olyanok lettek taglalva amiket vagy módosítani kellett, vagy pedig az implementáció során érdekesek, összetettek, említésre méltóak voltak.

A Yazdjerdi-féle jutalomfüggvény bementi paraméterei közé tartozik a kívánt tumor mérete (*desired_tumor_volume*) és a kívánt gyógyszer mennyiség a vérben (*desired_drug_level*). Itt egy pillanatra megállva, ha visszatekintünk az eredeti jutalomfüggvényre, abban egyáltalán nem is szerepelt a beadott ellenanyag koncentrációja. Az implementáció során ezt kellett bevezetni, hogy a pácienst az ügynök ne ölje meg túladagolással.

```
class YazdjerdiReward(Reward):
    def __init__(self, desired_tumor_volume=0, desired_drug_level = 0):
        self.previous_error = -1
        self.desired_tumor_volume = desired_tumor_volume
        self.desired_drug_level = desired_drug_level
        super(Reward, self).__init__()

    def get(self, params):
        current_tumor_vol = params["tumor_volume"]
        drug_level = params["drug_level"]
        life = params["life"]
        current_error = self.error(current_tumor_vol,
                drug_level, life)
        if self.previous_error == -1:
            self.previous_error = current_error
        if current_error < self.previous_error:
            calculated_reward =
                        (self.previous_error - current_error) /
                        self.previous_error
        else:
            calculated_reward = 0
        self.previous_error = current_error
        return calculated_reward

    def error(self, tumor_vol,drug_level, life):
        return np.abs((tumor_vol * life) -
                self.desired_tumor_volume) +
                np.abs((drug_level * life) -
                self.desired_drug_level)
```

5. Listing. Yazdjerdi-féle jutalomfüggvény implementációja

Ezeket a bemeneti paramétereket természetesen, ha nem adjuk meg akkor alapértelmezetten nullával inicializálódnak. Ugyan úgy ahogy timer (*sampling_time*) 1-gyel és az előző hiba nagysága pedig -1-gyel. Az egész jutalom függvény mivoltja az említett ősosztályból származó get(self, params) metódusban található. Itt a paraméter listából (amit az environment-nél megismerhettünk) kikérjük a tumor méretét és a gyógyszer koncentrációt. Ezt követően kalkulálunk egy error értéket ezek alapján, majd megvizsgáljuk, hogy a hibánk csökkent-e az előző állapothoz képest vagy nem. Ha igen akkor számítunk egy jutalom értéket az irodalomkutatásban megismert hiba hányadossal, ezzel szemben, ha pedig növekszik a hibánk akkor nem adunk jutalmat. Mindez után frissítjük az előző hiba értékét (*previous_error*), illetve az idő számlálónkon is növelünk egyet.

Az hibafüggvényben, az *error(self, tumor_vol, drug_level)*-ben jól látszik hogy ugyanazt az abszolútértékes error számítást hajtjuk végre mint amit megismerhettünk már egy előző fejezetben, annyi kiegészítéssel hogy ugyanez van implementálva nem csak a tumor méretére hanem a véráramban található ellenanyagra is. Ezért is van 2 bemeneti paramétere ennek a metódusnak.

A továbbiakban egy olyan jutalomfüggvény részletezése olvasható, melynél mindennemű módosítás nélkül, a modellből kiolvasható paraméterek felhasználásával történt meg annak implementálása. A jutalomfüggvényre a Zhao-féle jutalomfüggvényként volt hivatkozva az irodalomkutatás során. Megismételve az ott megemlítetteket, a függvény számára igen fontos szempont a páciens kezelés során történő túlélése. Így figyelembe kell venni a tumorméret-változása mellett azt is, hogy ez mennyi olyan dózissal jár, amely a szervezetben jelen van. Ennek megfelelően a jutalomfüggvény *inicializálása* során az alábbi függvény-beli változók lettek definiálva:

- *dead_punishment_const*: konstans, mely a szimuláció során "elpusztuláskor" jelentkezik
- *wellness_const*: konstans, mellyel a "jólétet" lehet súlyozni
- *tumor_const*: konstans, mellyel a tumor méretet lehet súlyozni
- *previous_wellness*: eltárolásra kerül az előzőleg mért "jólét" értéke
- *tumor_size*: megadható a tumor mérete

3.2.7. Összefoglaló implementáció

Mivel ezt egy közös kutatás keretében alkottuk meg (Hörömpő Andrással együtt) így közös volt az implementált architektúra is. Az, hogy ez hogyan épült fel és hogy miként lett felhasználva az implementáció, az alábbiakban taglaljuk.

A fentiekben bemutatott implementációkat egy .ipynb-ből (azaz IPythonNoteBookból) tudjuk felparaméterezni és használni. Ha megnézünk 1-1 ilyen .ipynb file-t, akkor látható, hogy gyakorlatilag pár sor kódból egy teljes tesztesetet tudunk összerakni. Ez is azt mutatja, hogy ha kellően átgondolt *framework* kerül létrehozásra, akkor a használata kellően egyszerű és letisztult lesz. Ez azonban azzal jár, hogy máshol összpontosul inkább a *know-how* kódszinten, ahogy az az előbbiekben be lett mutatva.

Egy ilyen file struktúrailag úgy épül fel hogy felparaméterezzük a környezetet, azt a fajtát amit éppen szeretnénk használni. Itt gyakorlatilag be lehet állítani a model és a jutalom függvény típusát. Eljárástól függően (hogy éppen *pytorch* vagy *keras* megvalósítást használunk) elláthatjuk egy neurális hálóval, majd mindkét esetben definiálhatjuk a *policy* típusát is. Végezetül összeállítódik az egész rendszer és futtathatók a tanítások, majd a tesztek. Tesztelés vizualizációja tekintetében előre definiált tesztalanyokon is ki lehet próbálni a megoldásokat amiknek az eredményeit diagrammokon képes prezentálni a rendszer (feltüntetve a tumor méretét, gyógyszer mennyiségét a szervezetben).

3.3. Végrehajtandó kísérletek

Végrehajtandó kísérletek tekintetében 2-szer 2-es megközelítés került alkalmazásra. Megkülönböztetünk kísérlet jellege szerintit, valamint végrhajtandó tesztek száma szerintit.

Jellegét tekintve a következőket lettek meghatározva meg:

- *Csúszóablakos különbözetű DQN* és *standard DQN* összehasonlítása az összes implementált jutalomfüggvénnyel
- *QRDQN*, *PPO* eljárások, *DRQN* összehasonlítása a *standard DQN*-nel, egy konkrét jutalomfüggvénnyel

Tesztek száma szerint:

- 1000 db teszt végrehajtása: túlélési idő *(steps)* és jutalom (*reward*) minimumának, maximumának és átlagának a kiértékelése
- Specifikus tesztalanyon végzett tesztelésnél, kezelési diagramok készítése és ezek elemzése a túlélés és kezelés alapján

4. fejezet

Kezdeti eredmények

4.1. Jutalomfüggvény-orientált megközelítés eredményei

Maga a fejezet 4 jutalomfüggvényről szól azonban a közös kutatás miatt (Hörömpő Andrással) kettőt itt lehet olvasni ezekből kettőt pedig kutatótársam munkájában.

A fejezetben használt elnevezések az alábbiak lesznek:

- G: utalás a Yauney-féle jutalomfüggvényre
- H: utalás a Hassani-féle jutalomfüggvényre
- Y: utalás a Yazdjerdi-féle jutalomfüggvényre
- Z: utalás a Zhao-féle jutalomfüggvényre
- TC: utalás a Drexler-féle *Twocompartment* modellre
- $TCLB$: utalás a Drexler-féle *Twocompartment csúszó ablakos különbözetű* modellre

Mivel a kísérletben egységenként három kísérlet fog megtörténni, így az alábbi jelölésmód lesz használatos:

$$< modellNeve >\_< jutalomfüggvényNeve >\_< kísérletSzáma >$$

Így például, ha az az elnevezés adott, hogy *TCLB_Z_3*, akkor annak jelentése, hogy

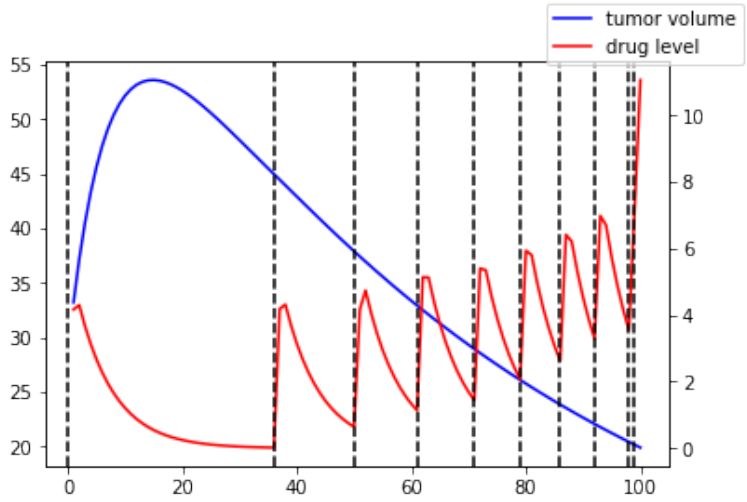
- A *Twocompartment csúszó ablakos különbözetű* modell került felhasználásra
- A Zhao-féle jutalomfüggvény került felhasználásra
- Ez a kísérlet ezzel a párossal a harmadik kísérlet

4.1.1. Eredmények Twocompartment Drexler modell esetén

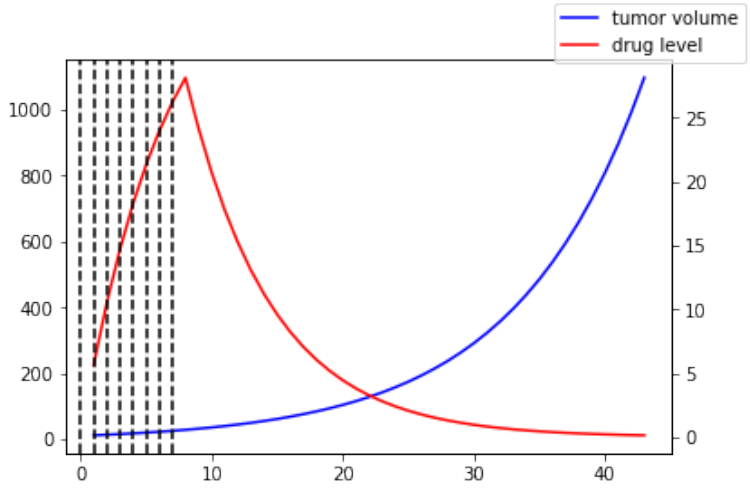
Hassani-féle jutalomfüggvény teszt

	reward	*steps*
minimum	**-5.772**	**1**
maximum	**5.77**	**100**
átlag	**0.002**	**13.829**

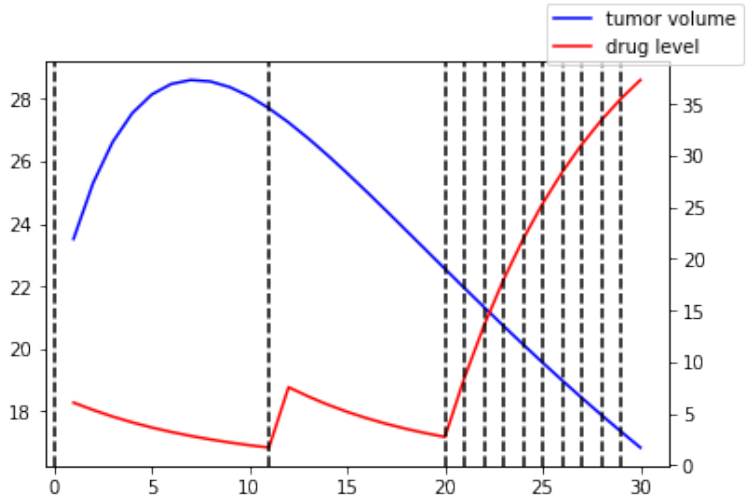
4.1. táblázat. *Twocompartment* modell Hassani-féle jutalomfüggvénnyel (*TC_H*)



4.1. ábra. TC_H 1. dedikált tesztesete, 8-as alany (*reward: -0.512, steps: 100*)



4.2. ábra. TC_H 2. dedikált tesztesete, 7-es alany (*reward: 4.549, steps: 43*)

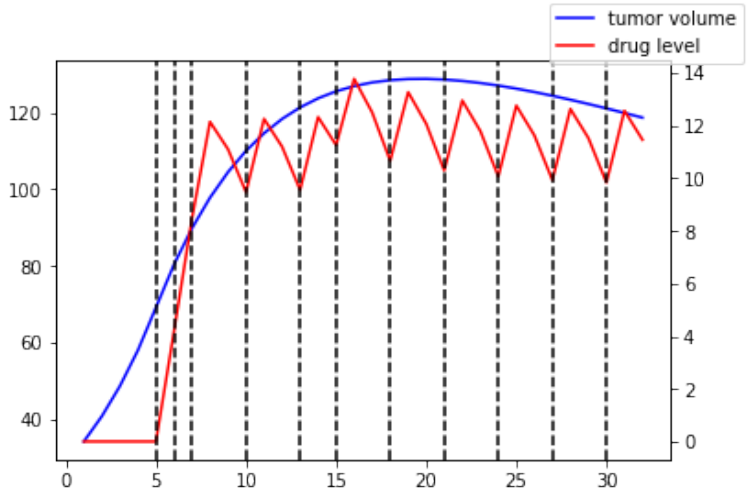


4.3. ábra. TC_H 3. dedikált tesztesete, 2-es alany (*reward: -0.334, steps: 30*)

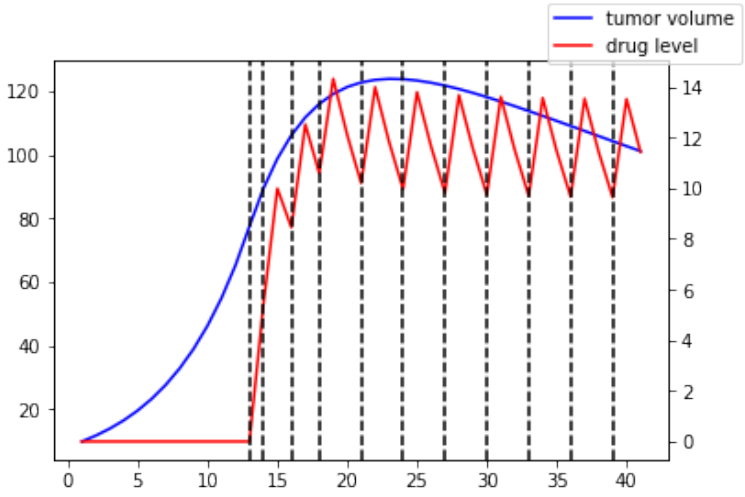
Yazdjerdi-féle jutalomfüggvény teszt

	reward	*steps*
minimum	**0**	**1**
maximum	**12.27**	**100**
átlag	**2.03**	**48.44**

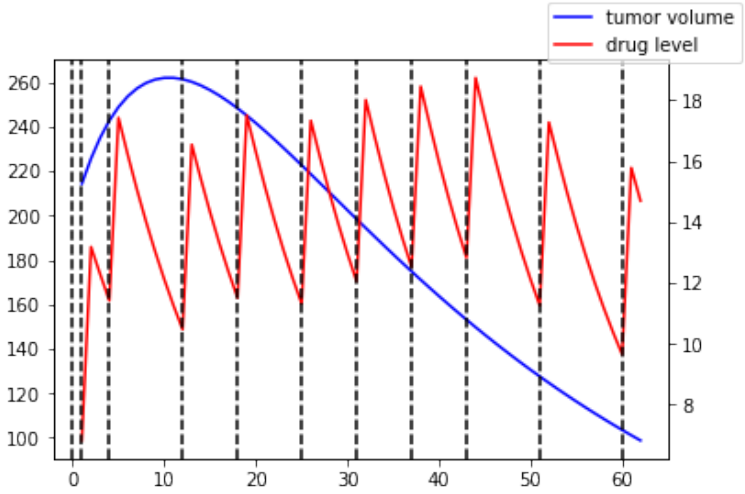
4.2. táblázat. *Twocompartment* modell Yazdjerdi-féle jutalomfüggvénnyel (*TC_Y*)



4.4. ábra. TC_Y 1. dedikált tesztesete, 8-as alany (*reward: 0.992, steps: 32*)



4.5. ábra. TC_Y 2. dedikált tesztesete, 5-ös alany (*reward: 1.013, steps: 41*)



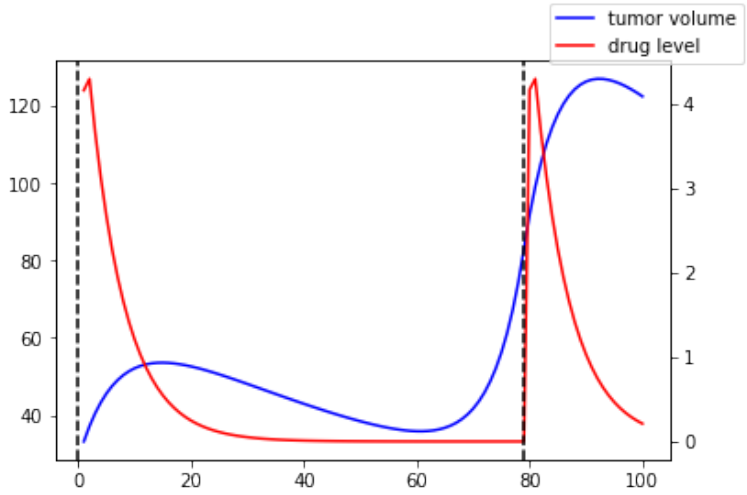
4.6. ábra. TC_Y 3. dedikált tesztesete, 4-es alany (*reward: 1.120, steps: 62*)

4.1.2. Eredmények a Twocompartment csúszó ablakos különbözetű Drexler modell esetén

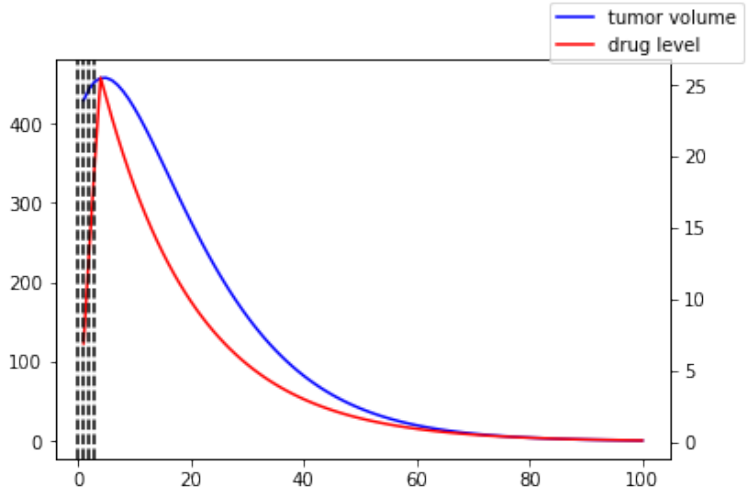
Hassani-féle jutalomfüggvény teszt

	reward	*steps*
minimum	**-13.794**	**1**
maximum	**23.767**	**100**
átlag	**-0.005**	**86.838**

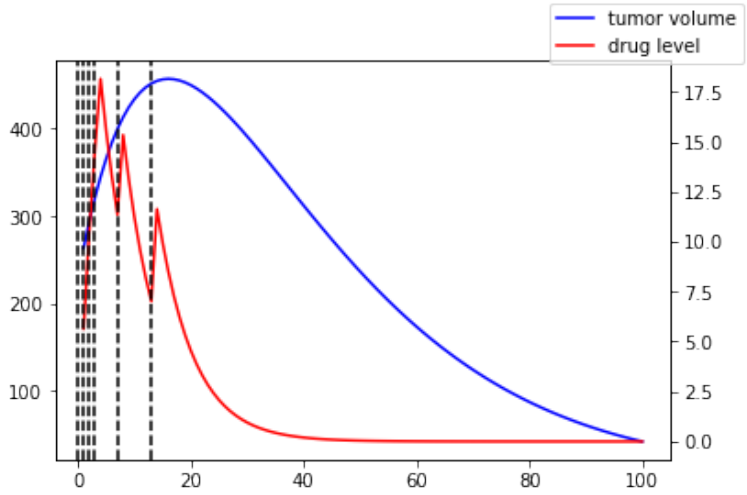
4.3. táblázat. *Twocompartment csúszó ablakos különbözetű* modell Hassani-féle jutalomfüggvénnyel (*TCLB_H*)



4.7. ábra. TCLB_H 1. dedikált tesztesete, 8-as teszt (*reward: 1.305, steps: 100*)



4.8. ábra. TCLB_H 2. dedikált tesztesete, 6-os teszt (*reward: -6.123, steps: 100*)

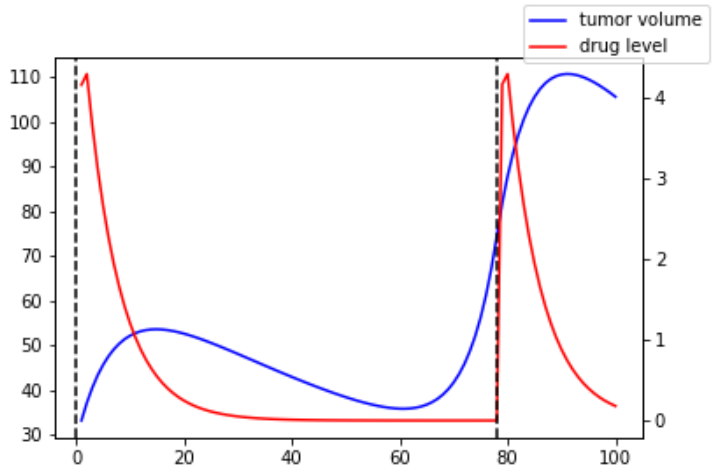


4.9. ábra. TCLB_H 3. dedikált tesztesete, 1-es teszt (*reward: -1.83, steps: 100*)

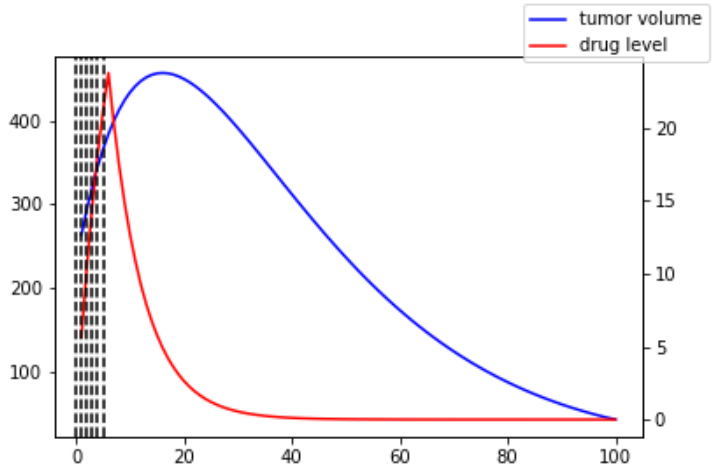
Yazdjerdi-féle jutalomfüggvény teszt

	reward	*steps*
minimum	**0**	**1**
maximum	**669.46**	**100**
átlag	**200.58**	**80.7**

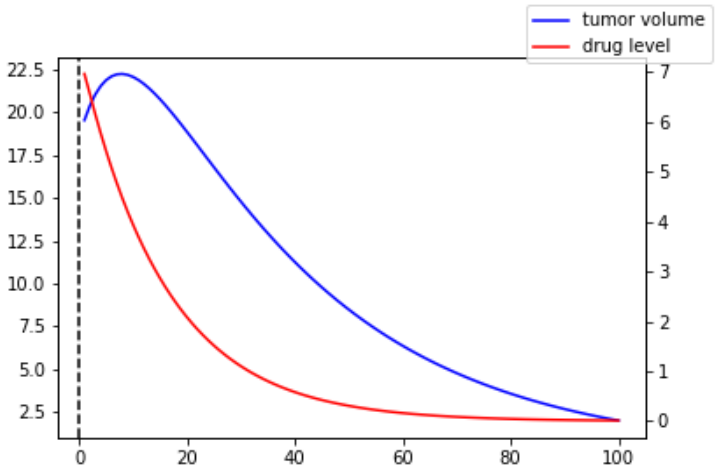
4.4. táblázat. *Twocompartment csúszó ablakos különbözetű* modell Yazdjerdi-féle jutalomfüggvénnyel (*TCLB_Y*)



4.10. ábra. TCLB_Y 1. dedikált tesztesete, 8-as teszt (*reward: 0.97, steps: 100*)



4.11. ábra. TCLB_Y 2. dedikált tesztesete, 1-es teszt (*reward: 0.752, steps: 100*)



4.12. ábra. TCLB_Y 3. dedikált tesztesete, 9-es teszt (*reward: 0.981, steps: 100*)

4.1.3. Jutalomfüggvény-orientált eredmények értékelése

Az eredmények a következőkben taglalt folyamat végén állnak elő. Legelőször virtuális alanyok kerültek generálásra random paraméterezett tulajdonságokkal, figyelve arra, hogy a valódikhoz hasonló tulajdonságokkal rendelkezzenek, csak véletlenszerűen. Ezeken a generált alanyokon zajlik a tanítás maga. Egy ciklus egy kezelést jelent, ez az *episode*, ami maga legfeljebb 100 napig tart. A *steps* nevű változó azt mutatja, 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 *reward* az egyes jutalomfüggvényekkel végzett kísérletek összehasonlításánál fontos, hogy egy-egy kezelés végén mennyi jutalmat ért el az ügynök adott környezettel és jutalomfüggvénnyel.

Tesztek struktúrája

A jutalomfüggvény-orientált tesztek során kétféle felépítés lett megvizsgálva: az egyik esetben a Drexler-féle kétkompartment modell szerint alap DQN megközelítésben, a másik esetben pedig ugyanennek a modellnek csúszó ablakos különbözeteken alapuló implementációja volt górcső alatt. Mindkét részben szerepel mind a 4-4 kiválasztott jutalomfüggvény.

Egy adott jutalomfüggvény tesztje egy adott környezeten az alábbiakból épül fel:

- véletlenszerűen megválasztott paraméterezésű tumormodell alapján az ágens betanításra kerül;
- 1000 darab kezelés van végrehajtva, szintén virtuális, random generált egyedeken;
- 3 db, a kísérletek során nem változtatott, fix paraméterezésű virtuális alany 1-1 kezelésének részletes bemutatása

Az 1000 darab kezelésből származó információk, az egyes esetek elején, táblázat formájában vannak összesítve, míg a 3 fix paraméterezésű egyed kísérletét a diagramok szemléltetik. A fix paraméterezés azt jelenti, hogy egy-egy sorszámozott alany tulajdonságai, tumorviselkedési jellemzői tesztektől függetlenül mindig ugyanazok. Emiatt egy adott egyed segítségével rendkívül jól össze lehet hasonlítani a környezetek és jutalomfüggvények hatását a kezelésre. Az eredmények összesítésekor a legenerált tesztalanyok közül a későbbiekben 8-as tesztalanyként hivatkozott egyedre történik egy-egy összehasonlító jellemzés.

8-as tesztalany eredményei

A Hassani-féle jutalomfüggvény esetén viszont a 8-as alanyra igen ígéretes megoldás mutatkozik. Egyrészt megállapítható, hogy a jutalom értéke közel esik az átlagoshoz, körülbelül 10%-os eltérés jelentkezik mindössze. Általánosan szembetűnik, hogy a tumor mérete egy kezdetleges emelkedést követően, bizonyos nap elteltével csökkenésnek kezd. Idővel szükséges plusz dózist a szervezetbe juttatni, de amíg csökken a méret, addig ez az érték is egy adott szintű határon belül mozog. Ami viszont nem várt viselkedésre utal, hogy a teszt végén egymás után adagol több dózist is, a dózismennyiség így egy nagyobb ugrást hajt végre, ami veszélyezteti az alany túlélését.

A Yazdjerdi-féle jutalomfüggvény során is ígéretes eredmény került megállapításra, ugyanakkor leellenőrizve a *steps* értékét, rá lehet jönni, hogy a teszt nem élte végig túl a tesztelési időt, annak nagyjából harmadánál tovább nem haladt. Nem is azonnal, hanem bizonyos idő elteltével kerül adagolásra, egymás után több adagnyi dózis is, majd egy periodikát követve, időszakosan újabb adag kerül a tesztbe. Egy adott szint így megmarad, sőt, maga a tumorméret is csökkenni kezd. Ugyanakkor a 32. lépés után nem folytatódik a kezelés. Ennek oka az, hogy a kezelés során az ágens elérte a maximális beadható dózismennyiséget, így az terminált. Javítandó és finomítandó a modell, ugyanis jól kivehető az is, hogy a tumor már reagált a kezelésre, az ágens viszont továbbra is az adagolás mellett döntött.

Érdekes észrevételként mutatkozott, hogy a Hassani-féle jutalomfüggvény esetén ugyanilyen megoldás került meghatározásra, mint a Yauney-féle esetén. Ez mutathat arra példát, hogy a Yauney- és Hassani-függvényekre ezen teszt hasonlóképpen reagál, továbbá ugyanazzal a modellel került betanításra a rendszer, amely rendszer figyeli az előző állapotokban mért értékeket is. Ugyanúgy túléli a teszt egészére vonatkozóan az alany, ugyanúgy kétszer kap csak dózist, és ugyanúgy, *reward*-jának értéke közel áll az átlaghoz. A megállapítások így egyezhetnek a Yauney-függvénynél tapasztaltakkal és megfogalmazottakkal.

A Yazdjerdi-jutalomfüggvénynél tapasztaltak hasonlítanak a fentebbiekhez: kevés dózis beadása mellett az alany hosszan túlélt, ennek megfelelően e jutalomfüggvény is alkalmazhatónak tűnik ha idősoros vagy lookback típusú felépítés a kiindulási alap.

Konklúziók a jutalomfüggvényekről

Összefoglalva a jutalomfüggvényekkel tapasztaltakat az alábbiakat lehet megfogalmazni:

- *TC_H*: A tesztek során kiszámításra került átlag *step* érték igen alacsony, amely arra utal, hogy általánosan kevés ideig élnek a páciensek. Ennek oka kettős lehet, vagy hirtelen kap nagyobb dózist a páciens, vagy pedig lecsökkent annyira a tumor mérete, hogy elérje a 0-át. A teszteredményeket alátámasztó diagramok mindkét eshetőséget alátámasztják, a függvény finomhangolása szükséges.

- *TC_Y*: Az átlag *step* érték megfelel az összesen elvégzett 100 lépés számtani közepével, így közepesnek tekinthető maga a megoldás is. A minimumérték ugyanakkor 1, azaz van olyan eset, amely rögtön terminálással végződik. Azon esetek viszont, melyek működőképesek, megállapítható, hogy az adagolt dózis mennyiségét egy adott határon belül tartják, nem lépnek át bizonyos értéket. Emellett általánosan megfigyelhető a tumorméret csökkenése is ezen esetek megfigyelésekor, úgyhogy a továbbiakban is figyelembe veendő párosról van szó.

- *TCLB_H*: Az imént említett Yauney-függvény mellett ezen függvény és modellpáros is hasonló értékeket produkál a *steps* értékére vonatkozóan, hiszen a magas átlagérték mellett akadnak hirtelen elpusztulásra kerülő alanyok. A tumorméret folyamatos csökkenő tendenciát mutat, a modell helyességét pedig alátámasztja, hogy a *8-as tesztalanyra* vonatkozó eredmények megegyeznek.

Az egymás után gyakran beadott dózisok kezelése feladat, általánosan elfogadott megoldásról van szó.

- *TCLB_Y*: Hasonlóan a többi *lookback* alapon implementált megoldáshoz, itt is stabil viselkedés állapítható meg, gyakori túléléssel, relatíve kevés beadott dózissal.

Eredmények összegzése

Összességében azt az aspektust figyelembe véve, hogy az ügynök mikor tudta a 100-hoz leginkább közel vinni a túlélési időt amellett, hogy minél kevesebb dózis ellenanyagot vigyen be az egyed szervezetébe, a csúszó ablakos különbözetű környezettel került elérésre leginkább a Yauney- , Hassani- (4.7 ábra) és a Yazdjerdi-féle (4.10 ábra) jutalomfüggvények esetén, a 8-as tesztalanyt tekintve. Ilyen kevés ellenanyag adagolást és túlélési időt egyik alkalommal sem sikerült elérni a sem a *twocompartment* eljárás esetén, sem pedig a fentebb említett csúszóablakos különbözetű eljárás esetén egy, akár másik jutalomfüggvénnyel kombinálva. Ezzel szemben természetesen láthattunk olyan megoldást is, ami az egyik környezet esetében adott megoldást, nem feltétlenül a legjobbat, de legalább adott, míg a másik környezet esetében nem. Ilyen volt például a Zhao-féle megoldás, amely vegyes eredményeket produkált a *lookback* alapú környezet esetén.

4.2. Eljárás-orientált megközelítés eredményei

A közös kutatás miatt és amiatt mivel a DQN-t vettük alapul az összehasonlításoknál így ez az eljárás mindkét dolgozat esetében meg fog jelenni. További 2 eljárásról viszont kutatótársam dolgozatában lehet olvasni.

A fejezetben használt elnevezések az alábbiak lesznek:

- *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

Az eljárás-orientált megközelítés szerint egységenként több kísérlet is fog megtörténni, az alábbi jelölésmód lesz alkalmazva az eredmények későbbi pontos értékeléséhez:

$$< környezetNeve >\_< módszerNeve >\_< jutalomFüggvényNeve >\_< kísérletSzáma >$$

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

Az eredmények ismertetése során minden teszteredmény esetén

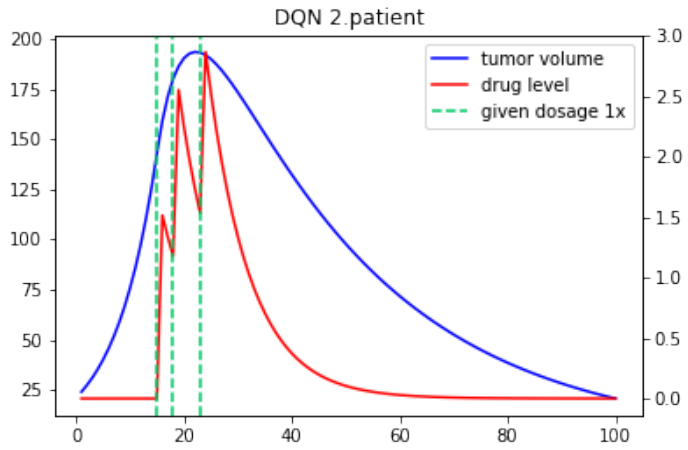
- a *Twocompartment* Drexler modell volt a viselkedést leíró matematikai modell, továbbá
- a *Long Living Linear* jutalomfüggvényt használtuk fel,

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

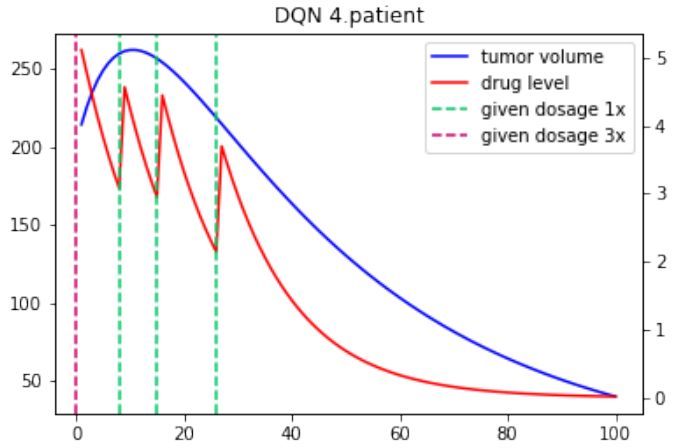
DQN teszt

	reward	*steps*
minimum	**-23830**	**1**
maximum	**515000**	**100**
átlag	**361750**	**87.273**

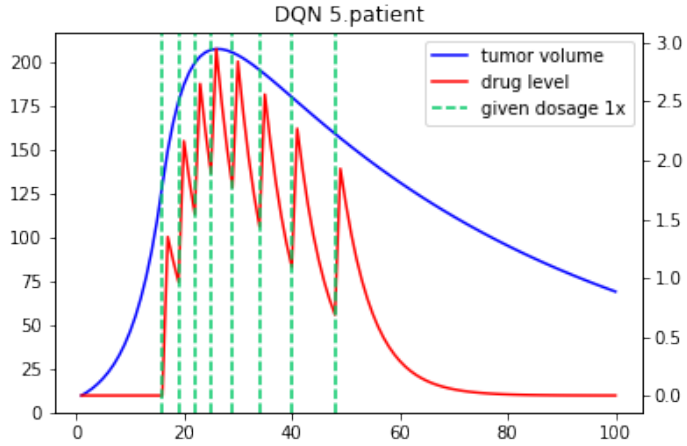
4.5. táblázat. DQN környezetben mért tesztek eredményei



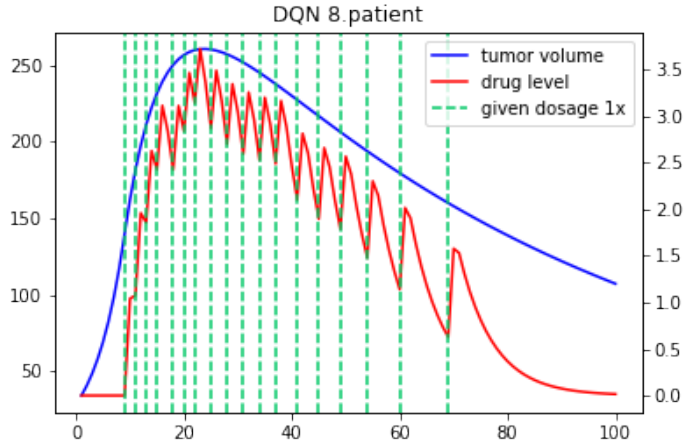
4.13. ábra. 2-es tesztalany DQN környezetben látható eredménye



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



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

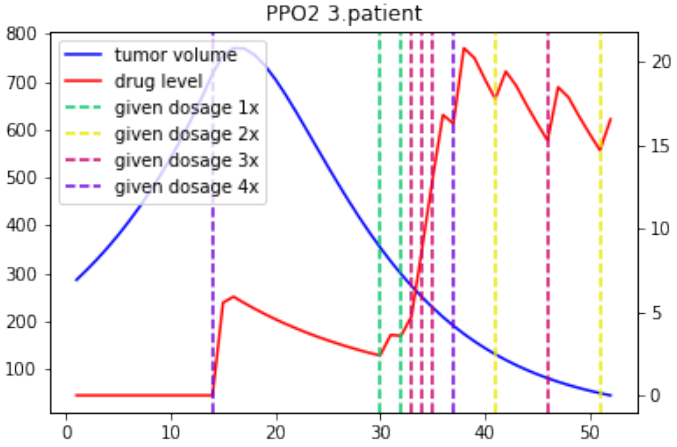


4.16. ábra. 8-as tesztalany DQN környezetben látható eredménye

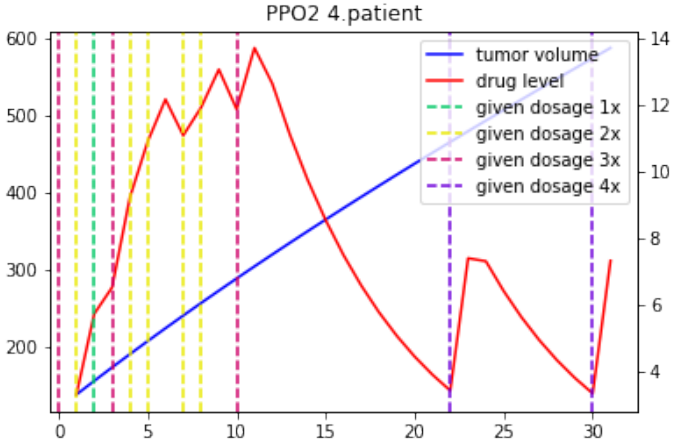
PPO2 teszt

	reward	*steps*
minimum	**-21034.69**	**0**
maximum	**440764.87**	**100**
átlag	**16845.85**	**20.95**

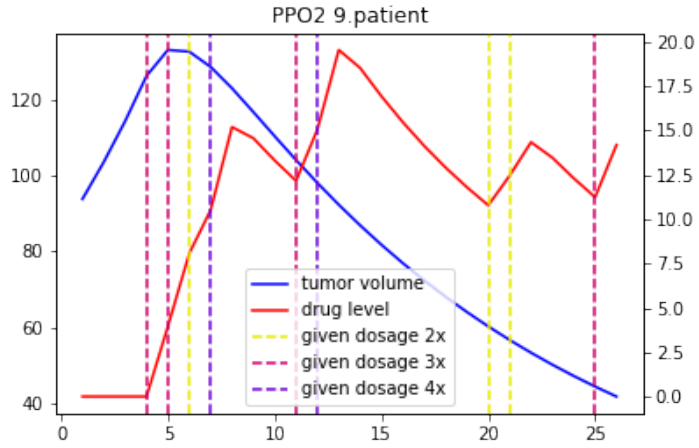
4.6. táblázat. PPO2 környezetben mért tesztek eredményei



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



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

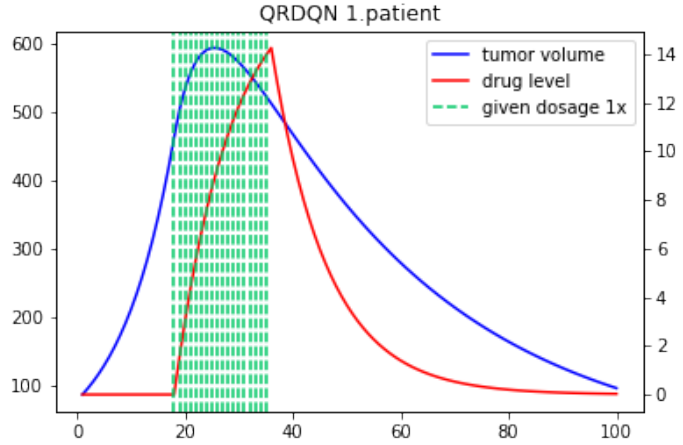


4.19. ábra. 9-es tesztalany PPO2 környezetben látható eredménye

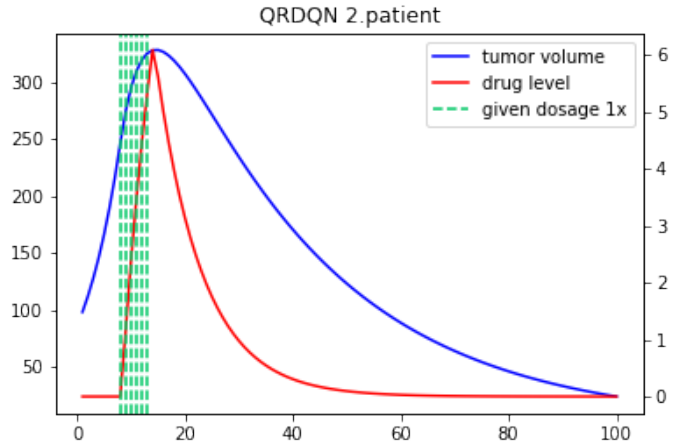
QRDQN teszt

	reward	*steps*
minimum	**-23400**	**0**
maximum	**515000**	**100**
átlag	**387620**	**82.37**

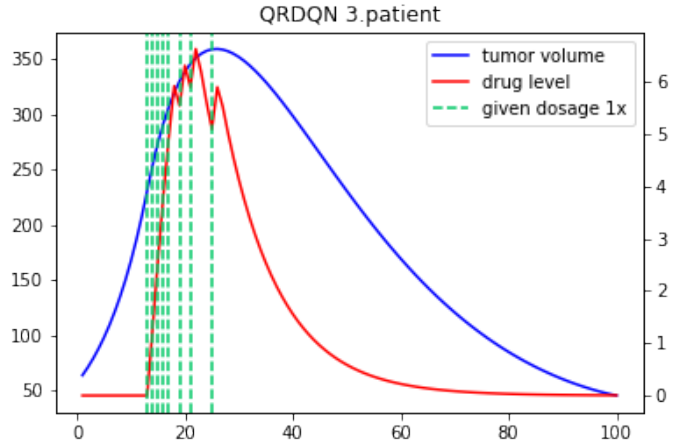
4.7. táblázat. QRDQN környezetben mért tesztek eredményei



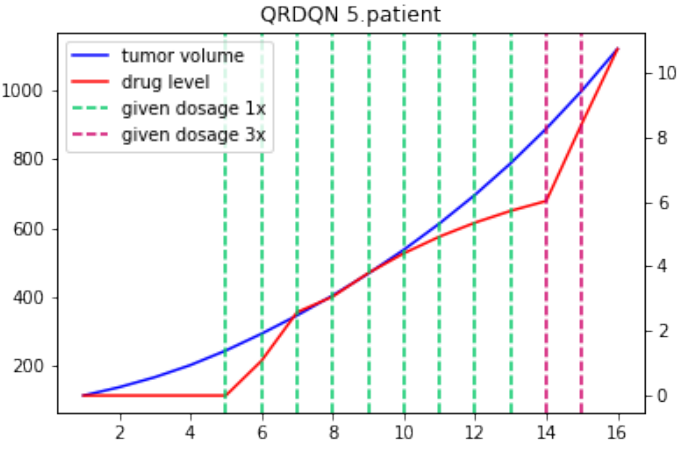
4.20. ábra. 1-es tesztalany QRDQN környezetben látható eredménye



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



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



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

4.2.1. Eredmények értékelése

Az eredmények értékelése folyamán kiindulási alapként a jutalomfüggvény-orientált megközelítésnél is alkalmazott módszer kerül felhasználásra. Erről röviden tehát azt kell tudni, hogy a valódi egyedekhez hasonló tulajdonságú, véletlenszerűen felparaméterezett esetek kerülnek generálásra, melyek legfeljebb 100 napig élhetnek túl. Elemzéshez használatos a változók közül a *steps*, azaz a túlélési idő, valamint a *reward*, azaz a jutalom. Utóbbi következetesen a jutalomfüggvény lineáris voltából, minél magasabb értéket vesz fel, annál inkább valószínűsíthető és állapítható meg jobb, hatékonyabb eljárási módszer, kezelés.

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 tesztelései.

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 tesztegyedek rendelkeztek. A fix paraméterezés azt jelenti, hogy egy-egy sorszámozott alany tulajdonságai, tumorviselkedési jellemzői tesztektől függetlenül mindig ugyan azok. 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.

Módszertanok összehasonlító elemzése

Az eljárások összehasonlító elemzése során alapul a DQN módszer által adott eredmények tekintendőek. A DQN eljárásban rejlő potenciált korábbi TDK munkánk során részleteit tekintve górcső alá vettük [1], mely eredmények jelen diplomamunkában is taglalásra kerültek. A környezeti implementáció, eljárás-orientált megközelítésnek módszertani részletezése során említésre került, hogy ezen tesztesetekben már 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 4.5 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 gyógyszermennyiséget 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 & 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 adatok kerülnek figyelembe vételre, 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 kialakítania. Jellemzőek az indokolatlan döntések mindamellett, 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 gyógyszermennyiséget, 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 4.23. 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 megfigyeltekhez, 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 az a 4.7. 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 gyógyszerszint 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 gyógyszermennyiséget, amennyit akar adagolni, ellenben a DQRN a tumor térfogatot és annak esetleges újbóli növekedését figyelembe véve dönt újbóli adagolásról

- A DRQN és a QRDQN (4.7) táblázatait 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.

Konklúziók a módszertanokról

Összefoglalva az egyes módszertanokat, az alábbiakat lehet megállapítani:

- *DQN*: A kiindulási pontnak tekinthető DQN eljárás az egyes fejlesztések alkalmazása után is megfelelő eredményt volt képes produkálni. A maximálisan adható gyógyszermennyiséget 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.

- *PPO-PPO2*: Ha kontextusba helyezzük a többi, eljárás-orientált részen 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 gyógyszerszint 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.

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 megemlítendő, 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 a diplomamunka készülése során nem megfelelően kerültek illesztésre megírásának pillanatáig, és noha felhasználja a *multi-action* megközelítést, nem számítanak megfelelőnek a kiolvasható értékek, még az alapnak szolgáló DQN-hez képest 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 mély megerősítéses tanulással végeztük el.

Az irodalomkutatás során vizsgált hasonló megoldásokban többféle jutalmazással és eljárással is találkoztunk, melyek közül a problémához és a modellhez illeszkedőeket kiválasztottuk, és implementáltuk. Ezt követően számos szimuláció került elvégzésre, melyek eredményeiből következtethetni lehet az egyes megoldások hatékonyságára, későbbi alkalmazhatóságára.

5.1. Továbbfejlesztési és további kutatási tervek

Továbbfejlesztési, kutatási tervek tekintetében a következő irányok kerültek meghatározásra:

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

Célszerű megvizsgálni további megerősítéses tanulási eljárásokat is. Ilyen például a DDQN (Double Deep Q-Network) [37] 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) [38] 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 tumornövekedésmodellen 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.

Kutatótársammal, Hörömpő Andrással igyekeztünk maximális odafigyeléssel és körültekintéssel készíteni kutatásunkat, egyben diplomamunkánkat, hogy a lehető legjobb eredményeket tudjuk megállapítani az optimális kezelésre vonatkozóan. Közös irodalomkutatásunk kijelölte számunkra a fejlesztés lépéseit, magát az egész folyamatot. Folyamatos egyeztetéssel a jutalomfüggvény-orientált megközelítésből végeztem a Hassani- és Yazdjerdi-jutalomfüggvények implementációját, tesztelését, és hasonlóképp jártam el az eljárás-orientált megközelítéseknél a baseline-nak tekinthető DQN melletti QRDQN és PPO2 módszerek optimalizálására.

Ábrajegyzék

Táblázatok jegyzéke

Forráskódok listája

Irodalomjegyzék

[1] Almásy Márton György és Hörömpő András. *Tumorkezelési eljárás optimalizálása mély megerősítéses tanulás alapon*. 2021.

[2] Almásy Márton György és Hörömpő András. *On-policy mély megerősítéses tanulás tumorkezelés optimalizálására*. 2022.

[3] Márton György Almásy & András Hörömpő & Dániel Kiss & Gábor Kertész. "A Review on Modeling Tumor Dynamic and Agent Reward Functions in Reinforcement Learning Based Therapy Optimization". *Journal of Intelligent & Fuzzy Systems*. 2022.

[4] Richard S Sutton és Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[5] 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 megtekintve: 04.10.2021).

[6] Richard Bellman. "A Markovian decision process". *Journal of mathematics and mechanics* 6.5 (1957), 679–684. old.

[7] John N Tsitsiklis. "Asynchronous stochastic approximation and Q-learning". *Machine learning* 16.3 (1994), 185–202. old.

[8] blackburn. *Reinforcement Learning : Markov-Decision Process*. URL: `https://towardsdatascience.com/introduction-to-reinforcement-learning-markov-decision-process-44c533ebf8da`. (accessed: 04.10.2021).

[9] Ludwig Von Bertalanffy. "Quantitative laws in metabolism and growth". *The quarterly review of biology* 32.3 (1957), 217–231. old.

[10] 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.

[11] Hyun M Yang. "Mathematical modeling of solid cancer growth with angiogenesis". *Theoretical Biology and Medical Modelling* 9.1 (2012), 1–39. old.

[12] 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.

[13] Dániel András Drexler, Tamás Ferenci, András Füredi, Gergely Szakács és Levente Kovács. “Experimental data-driven tumor modeling for chemotherapy**This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 679681). The present work has also been supported by the Hungarian National Research, Development and Innovation Office (2018- 2.1.11-T ET-SI-2018-00007 and SNN 125739).” *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.

[14] Christopher JCH Watkins és Peter Dayan. “Q-learning”. *Machine learning* 8.3-4 (1992), 279–292. old.

[15] Tuomas W Sandholm és Robert H Crites. “Multiagent reinforcement learning in the iterated prisoner’s dilemma”. *Biosystems* 37.1-2 (1996), 147–166. old.

[16] 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.

[17] Leslie Pack Kaelbling, Michael L Littman és Andrew W Moore. “Reinforcement learning: A survey”. *Journal of artificial intelligence research* 4 (1996), 237–285. old.

[18] 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.

[19] Volodymyr Mnih és tsai. “Playing atari with deep reinforcement learning”. *arXiv preprint arXiv:1312.5602* (2013).

[20] Volodymyr Mnih és tsai. “Human-level control through deep reinforcement learning”. *nature* 518.7540 (2015), 529–533. old.

[21] Matthew Hausknecht és Peter Stone. “Deep recurrent q-learning for partially observable mdps”. *arXiv preprint arXiv:1507.06527* (2015).

[22] 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.

[23] Christos H Papadimitriou és John N Tsitsiklis. “The complexity of Markov decision processes”. *Mathematics of operations research* 12.3 (1987), 441–450. old.

[24] David E Rumelhart, Geoffrey E Hinton és Ronald J Williams. “Learning representations by back-propagating errors”. *nature* 323.6088 (1986), 533–536. old.

[25] Sepp Hochreiter és Jürgen Schmidhuber. “Long short-term memory”. *Neural computation* 9.8 (1997), 1735–1780. old.

[26] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford és Oleg Klimov. “Proximal policy optimization algorithms”. *arXiv preprint arXiv:1707.06347* (2017).

[27] 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.

[28] Regina Padmanabhan, Nader Meskin és Wassim M Haddad. “Reinforcement learning-based control of drug dosing for cancer chemotherapy treatment”. *Mathematical Biosciences* 293.11-20 (2017), 15–16. old. DOI: https://doi.org/10.1016/j.mbs.2017.08.004.

[29] 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.

[30] 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.

[31] Yufan Zhao, Michael R Kosorok és Donglin Zeng. “Reinforcement learning design for cancer clinical trials”. *Statistics in medicine* 28.26 (2009), 3294–3315. old.

[32] 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.

[33] Ammar Jalalimanesh, Hamidreza Shahabi Haghighi, Abbas Ahmadi és Madjid Soltani. “Simulation-based optimization of radiotherapy: Agent-based modeling and reinforcement learning”. *Mathematics and Computers in Simulation* 133 (2017), 235–248. old.

[34] Kyle Humphrey. “Using reinforcement learning to personalize dosing strategies in a simulated cancer trial with high dimensional data”. (2017).

[35] Huan-Hsin Tseng, Yi Luo, Sunan Cui, Jen-Tzung Chien, Randall K Ten Haken és Issam El Naqa. “Deep reinforcement learning for automated radiation adaptation in lung cancer”. *Medical physics* 44.12 (2017), 6690–6705. old.

[36] Panayiotis Petousis, Simon X. Han, William Hsu és Alex A. T. Bui. “Generating Reward Functions Using IRL Towards Individualized Cancer Screening”. *Artificial Intelligence in Health*. Szerk. Fernando Koch és tsai. Cham: Springer International Publishing, 2019, 213–227. old. ISBN: 978-3-030-12738-1.

[37] 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.

[38] 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.