Eötvös Loránd Tudományegyetem
Természettudományi kar

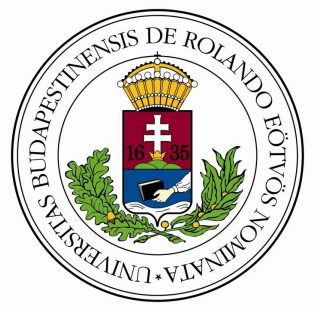
Kiss Gergő

A HEADS UP NO LIMIT TEXAS HOLD'EM JÁTÉK MODELLEZÉSE

Szakdolgozat
Matematika BSc
Matematikai elemző szakirány

Témavezető:
Csiszárik Adrián

Számítógéptudományi Tanszék



Budapest, 2020

NYILATKOZAT

Név: Kiss Gergő

ELTE Természettudományi Kar, szak: Matematika BSc

NEPTUN azonosító: LYGWKN

Szakdolgozat címe:
A Heads Up No Limit Texas Hold'em játék modellezése

A **szakdolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2020. 12. 28.

Kiss Gergő

a hallgató aláírása

Tartalomjegyzék

Ábrajegyzék

Bevezetés

A No Limit Texas Hold'em játékosok millióit vonzza magához világszerte. Számos történet szól hirtelen meggazdagodásról, hatalmas csődökről, nem egy alkalommal kapta már fel Hollywood. De mitől ennyire népszerű ez a játék?

A válasz egyszerű: a szerencsefaktor. A sakkban ha egy 1500-as ÉLŐ pontszámmal rendelkező játékos egy nagymester ellen játszik, mindig ki fog kapni. Nem így a pókerben. A rövidtávú szerencsefaktor miatt a tudáskülönbség ellenére is előfordulhat, hogy felülkerekedjen egy amatőr játékos egy profin.

Akkor a póker tulajdonképpen egy szerencsejáték? Ha azt értjük szerencsejáték alatt, ahol szerencse nélkül nem nyerhetünk (azaz a játék akármilyen stratégia mellett negatív várhatóértékű), akkor nem, korántsem. Olyan játékokkal szokták egy lapon említeni a pókert, mint a blackjack, roulette, félkarú rabló, ám míg ezek mindegyikét a ház ellen játsszuk, addig a pókert élő játékosok ellen. Az előbb említett játékokról mind megmutatható matematikai számításokkal, hogy negatív várható értékűek (azaz hosszútávon a játékos biztosan veszít), a pókert viszont megfelelő tudással, és valamilyen mértékben kihasználható ellenfelekkel lehet pozitív várható értékkel játszani.

Éppen ezért tekintik sportnak, rendeznek belőle bajnokságokat, sokan pedig hivatásként űzik, illetve maga a játék számos kutatás alapjául szolgál.

Míg például a go és a sakk esetében sikeresen írtak olyan programot, ami bármely mai játékost képes könnyűszerrel legyőzni neurális hálók segítségével [4, 14], addig a pókerről ez nem mondható el. Az az igen fontos tényező, hogy a pókerben nem rendelkezünk minden információval úgy, ahogy a fent említett játékokban olyannyira megbolondítja az egészet, hogy a mai napig is hatalmas kihívásként lebeg ez a kutatók előtt [8, 13, 5].

Dolgozatom célja, hogy bemutassam a Heads Up No Limit Texas Hold'em játék egy lehetséges modellezését egy neurális hálózatokat használó programon keresztül.

Hogy miért választottam épp ezt a témát? Lévén, hogy a jelenlegi csúcsteljesítményű mesterséges póker programok még szuperszámítógépek segítségével sem képesek megoldani a játékot, én sem ezzel a céllal vágtam neki. Ellenben szerettem volna felfedezni és megérteni, hogy hogyan közelítik meg ezt a problémát, hogyan működik egy ilyen program, mi okozza benne a nehézségeket.

Másfél éven át egy profi játékosokat összegyűjtő és támogató cég pókerjátékosa voltam, ahol 3 fős játékról tanultam napi szinten ezidő alatt. Olyan mennyiségű stratégiát tanultam, amivel önmagában meg lehetne tölteni egy dolgozatot. Munkám során ennek fényében tervezem az általam tanultakat összevetni póker programok tudásával.

Az első fejezetben bemutatom a játék két fős verzióját, a Heads Up No Limit Texas Hold'emet, majd elhelyezem a játékelméleti térképen. Beszélek a modellezési nehézségekről, és bevezetek pár nagyon fontos fogalmat ahhoz, hogy numerikus folyamatokkal közelíthessük a játék megoldását.

A második fejezetben dolgozatom fő céljaként részletesen bemutatok egy programot, amit 2016-ban készítettek, majd a Science-ben publikáltak, aminek egy-az-egy ellen sikerült legyőznie profi játékosokat neurális hálózatok segítségével.

Végezetül a harmadik fejezetben a második fejezetben bemutatott programot összehasonlítom a jelenlegi legerősebb Heads Up No Limit Texas Hold'em programmal, ami neurális hálózatok nélkül működik.

1. fejezet

A No Limit Texas Hold'em (NLTH) játék modellezése

Annyiféle pókerjáték létezik, mint ahány csillag van az égen. Mégis egy fajtáját nevezték el a pókerjátékok királyának, a No Limit Texas Hold'emet (NLTH). Megjelenése óta ez a legnépszerűbb variáció. Rengetegen szerencsejátéknak tartják, rengetegen pedig játékukkal rácáfolnak erre az állításra. A szerencsejáték definíciója a Magyar értelmező kéziszótár [3] alapján:

> **Szerencsejáték**: „Az a kártyával v. más eszközzel űzött s pénz kockáztatásán alapuló játék, amelynek kimenetele nem a játékosok következtető képességétől, hanem pusztán a szerencsétől függ; hazárdjáték."

Ez a definíció pedig egyértelművé teszi, hogy a póker semmiféleképp nem szerencse-, hanem egy tudásalapú játék.

A következőkben a két fős változatot fogom bemutatni, ugyanis azonkívül, hogy jómagam is erről tanultam a legtöbbet, a kutatások is ezzel foglalkoznak jellemzően, ugyanis három vagy több játékos még túl nagy falatnak bizonyul a mai számítógép kapacitások mellett.

1.1. A Heads Up NLTH játék szabályai

A Heads Up NLTH egy kétszemélyes pókerjáték, amit klasszikus 52 lapos franciakártyával játszanak. A két játékos *leosztásokat* (angolul *hand*) játszik egymással ismétlődve, váltogatva, hogy ki az *osztó*. A cél, hogy a *leosztások* során a lehető legtöbb *zsetont* (chip) nyerjük el az ellenfelünktől.

Egy *leosztás* négy szakaszból vagy *utcából* áll, ezek a következők: *preflop, flop, turn, river*. Mindegyik során lapokat osztanak, majd ezt követően licitálnak. A *preflop* során 2-2 *saját lapot* kap mindkét játékos, amit a másik nem lát, a további 3 *utcán* pedig *közös lapokat* osztanak felfordítva. A *flopon* hármat, *turn-ön* és *riveren* pedig egyet-egyet.

A licitálás során 3 féle akció van: *dobás, megadás* és *emelés*. A *dobás* azt jelenti, hogy a játékos lapjait eldobja, a feltett tételeket (*pot*) pedig a másik játékos nyeri meg, és új *leosztás* következik. A *megadás* az előtte történt *emelés* mértékének *potba* tételét, az *emelés* pedig új tét behelyezését jelenti. Az *emelés* mértéke minimálisan mindig legalább az aktuális licitkörben az azt megelőző tét, vagy minimum a *nagyvak* mértéke. A *licitkör* elején, amikor még nincs tét, a *megadást check*-nek hívják, ez annyit tesz, hogy az adott játékos nem emel. Azt az *emelést*, amikor egy játékos az összes *zsetonját* felteszi, *all-in*-nak hívjuk. Ha mindkét játékos *check*-elt, vagy egy *emelést* megadtak, akkor a következő *utca* jön (feltéve, hogy mindkét játékosnak volt lehetősége licitálni).

A *leosztás* elején az *osztó* tétként beteszi a *kisvakot* (ezért rá *kisvakként* és *osztóként* is hivatkozhatunk), a másik játékos pedig ennek kétszeresét, a *nagyvakot*. Ezután a *preflop* során mindkét játékos kap 2 *saját lapot*, és licitálni kezdenek. A *kisvak* játékos kezd, aki ekkor vagy *dob*, vagy *megadja* a *nagyvak* által berakott tétet (azaz kiegészíti saját tétjét 1 *kisvakkal*), vagy *emel*. Amennyiben a *kisvak* csak megad, úgy a *nagyvaknak* még lehetősége van licitálni. Ez lehet *check*, vagy *emelés*. Innentől teljességgel igaz, hogy a *licitkör* két *check*, vagy egy *megadás* után ér véget.

Preflop után a *flop* során mint említettem 3 *közös lap* kerül lehelyezésre, amit *licitkör* követ, majd a *turn* során 1 lap és *licitkör*, *river* során szintén 1 lap és egy végső *licitkör*. A *preflop licitkörét* a *kisvak* kezdte, a *floptól* kezdve viszont a *nagyvak* az első játékos. Amennyiben az utolsó *licitkör* végéig nem történt *dobás*, úgy *mutatás* következik. Ekkor a játékosok felfordítják lapjaikat, és az nyer, aki a 2 *saját* és 5 *közös lapjából*, azaz 7 lapból a legerősebb 5 lapos *pókerkezet* tudja kiválasztani. A nyertes elviszi a *zsetonokat*, majd *osztót* váltanak, és új *leosztás* következik.

1.1.1. Szótár

all-in Olyan emelés, amikor a játékos az összes rendelkezésre álló zsetonját felteszi.

check Olyan megadás, ahol a tét 0 zseton volt.

dobás A játékos nem tartja a tétet, hanem kiszáll a körből. A potot a másik játékos nyeri meg.

emelés A tét megadásán felül további zsetonok berakása a potba. Egy emelés mértéke minimálisan az előző emelés mértéke és a nagyvak mértéke közül a nagyobb.

flop A 2. utca. Ekkor 3 közös lap kerül leosztásra. A nagyvak kezdi a licitkört.

kisvak A nagyvak tét fele. Hivatkozhatunk vele magára a játékosra is, aki egyben az osztójátékos. Ő kezdi a preflop licitkörét.

közös lap Azok a lapok, melyet mindkét játékos lát, és pókerkezének kialakításához használhat. Egy leosztás során maximum 5 lehet belőle.

leosztás Egy kör pókerjáték. A vakok berakásával kezdődik, a pot valamely játékos általi megnyerésével és osztócserével végződik.

licitkör Mind a négy utcán az osztást követő szakasz. Ekkor helyezhetnek tételeket a játékosok.

megadás A tét kiegészítése saját zsetonjainkból.

milli-big-blinds per game (mbb/g) Egy leosztás alatt nyert ezrednagyvak.

mutatás Ha az utolsó utcát követő licitkör végéig nem történt dobás, úgy a játékosok felfedik a lapjaikat, és az erősebb pókerkezet kialakító játékos nyer.

nagyvak A játék során a fő alaptét. A kisvak kétszerese. Hivatkozhatunk vele magára a játékosra is. A floptól kezdve ő kezdi a licitköröket.

osztó Heads Up játékban egybeesik a kisvak játékossal.

pókerkéz 5 lapból álló különböző erősségű kombinációk. Lásd: kártyakombinációk cikk a Wikipédián[1].

[1] https://en.wikipedia.org/wiki/List_of_poker_hands

pot Tétként behelyezett zsetonok, amit egy játékos megnyerhet a leosztás végén.

preflop Az 1. utca. 2 saját lap osztása történik. A kisvak kezdi a licitkört.

river A 4. utca, ahol az utolsó (5.) közös lap is leosztásra kerül. A nagyvak kezdi a licitkört.

saját lap A preflop során kiosztott lapok. Ezeket csak az a játékos láthatja és használhatja fel, akinek osztják őket.

turn A 3. utca, ahol a 4. közös lap kerül leosztásra. A nagyvak kezdi a licitkört.

utca Egy leosztáson belüli köröket nevezünk így. Ezek időrendi sorrendben a preflop (1. kör), flop (2. kör), turn (3. kör) és river (4. kör).

zseton A játékhoz, az emelésekhez használt valuta.

1.2. Játékelméleti térképen való elhelyezés

A modellezési kérdések tárgyalásához elengedhetetlen játékelméleti szempontból is jellemezni magát a játékot. Itt olyan tulajdonságokat vizsgálunk, mint például a játékosoknál lévő információk, a résztvevők száma, illetve a stratégiánk célja.

1.2.1. Definíció. (Véges játék) *Véges játéknak* nevezünk egy olyan játékot, amelyben bármely játékállásból véges sok lépésen belül befejeződik.

1.2.2. Definíció. (Nulla-összegű játék) *Nulla-összegű játéknak* nevezünk egy olyan játékot, ahol a játékosok által nyert nyeremények összege mindig nulla.

1.2.3. Megjegyzés. A versenyek során a játéktermek jutalékot vonnak le a játékosoktól, de mivel ezt nem a játékban használt zsetonból, hanem a nevezési díjból teszik, de ettől még magát a NLTH játékot nulla-összegű játéknak tekintjük.

1.2.4. Definíció. (Nash-egyensúly) Nash-egyensúlynak a résztvevő játékosok olyan stratégiaegyüttesét hívják, ahol minden játékos úgy érheti el a legjobb várhatóértéket, ha nem változtat az aktuális stratégiáján csak akkor, ha más játékos változtatott.

Később, az 1.6-os szekcióban ezt formálisan is bevezetem majd.

1.2.5. Megjegyzés. (Tiszta és kevert Nash-egyensúly) A játékelméleti irodalom megkülönböztet tisza Nash-egyensúlyt és kevert Nash-egyensúlyt, amelyek abban különböznek, hogy tiszta vagy kevert stratégiákon értelmezik a Nash-egyensúly fogalmát, ahol a kevert stratégia egy stratégiák feletti valószínűségeloszlással definiált stratégia. Dolgozatomban, ha másképp nincs jelölve, akkor Nash-egyensúly alatt kevert Nash-egyensúlyt, stratégia alatt pedig kevert stratégiát értek.

1.2.6. Tétel. *Minden véges játékban létezik kevert Nash-egyensúly. [12]*

1.2.7. Következmény. A Heads Up NLTH játék a szabályaiból adódóan véges, így létezik benne kevert Nash-egyensúly.

1.2.8. Következmény. A Heads Up NLTH játékban a játékosok a legjobb várható értéket a Nash-egyensúly segítségével érhetik el.

Még egy szempontról kell említést tenni, mégpedig arról, hogy mit jelent, hogy a póker egy nem teljesinformációs játék, és hogy milyen nehézséget okoz ez.

1.2.9. Definíció. (Teljes- és nem teljesinformációs játékok)
Információs szempontból megkülönböztetünk *teljes-* és *nem teljesinformációs játékokat.*

Egy *teljesinformációs játék* során a játékosok minden információval rendelkeznek.

Ezzel szemben egy *nem teljesinformációs játék* résztvevői közös és saját információkkal is rendelkezhetnek.

Teljesinformációs játékokra példa a sakk, a go vagy a backgammon, *nem teljesinformációsra* pedig a póker legtöbb fajtája, többek között a Texas Hold'em is. A Texas Hold'em játékban közös információ a játékosok zsetonjainak száma, a pot mérete, a közös lapok, a soron következő játékos kiléte, lehetséges akciói, a játékosok pozíciói, stb. Saját információ pedig a két saját lap, ugyanis egy játékos saját lapjait semelyik másik játékos nem láthatja.

1.2.10. Állítás. A Heads Up No Limit Texas Hold'em egy kétszemélyes, véges, nulla-összegű, nem teljesinformációs játék.

Bizonyítás. *A játék szabályaiból 1.1 triviálisan következik.*

A számítást az nehezíti meg, hogy a játékosok akciói nagymértékben függnek a saját lapjaiktól, így minden akciónál egy becsült valószínűséget is kell számolnunk.

1.3. Lehetséges célok

A NLTH okkal a világ egyik legnépszerűbb kártyajátéka. Bár egy játékként beszélünk róla, teljesen más hozzáállást, stratégiát igényel ha élőben, vagy ha online játszunk, számít, hogy hány ellenfelünk van, ahogy az is, hogy milyen gyorsan emelkednek a vakszintek. Hiába egységesek két NLTH szabályai, a változó körülmények annyiféle új játékot hoznak létre, amennyit csak elképzelni lehet.

Ebben a részben szemelvényezek röviden és általánosan az emberi stratégiaalkotás körülményeiről, majd részletesebbben a dolgozat szempontjából legjelentősebb kérdésről, hogy egy pókerprogramnak mi a célja.

1.3.1. Az emberi stratégiaalkotás sarokpontjai NLTH játékban

A NLTH játékban a célunk az, hogy a lehető legtöbbet nyerjünk, ám hogy ezt milyen stratégiával tehetjük meg, az szörnyen sok tényezőn múlik.

Nem célom teljeskörű elemzést adni arról, hogy a póker játék stratégiáit hogyan alkothatják meg emberi játékhoz, csak egy betekintést nyújtani néhány alapelvhez.

Élőben vagy online? Ami e kérdés kapcsán elsőként eszünkbe juthat — talán jogosan —, az a bizonyos „pókerarc”. Vitathatlan, hogy voltak olyan idők, amikor még releváns lehetett más játékosok arckifejezése és testbeszéde, ám ma a számítógépek világában, amikor elemzőprogramok tömegei állnak rendelkezésünkre, hogy az utolsó cseppig kipréselhessük a lehetséges legjobb várható értéket stratégiánkból, a mimika háttérbe szorult.

Nem csak az elemzőprogramok miatt persze, hanem mert mások testbeszéde szörnyen megbízhatatlan. Kismértékű energiabefektetéssel is elérhető, hogy uraljuk arcunkat, mozdulatainkat, így akár ezt megtévesztésre is könnyedén használhatjuk. Ellenben azt nem lehet eljátszani, hogy jól játszunk, ha egyébként nem tesszük.

Bár a filmek szeretik a hatalmas blöffök csatájaként beállítani, a póker valójában közel sem erről szól. Minden döntés mögött — így a blöffök mögött is — óvatos és alapos matematikai számítások vannak, amik indokolják

őket. Egy profi játékosnak nincs szüksége arra, hogy a másik testi reakcióin gondolkodjon, ehelyett az előző akcióin gondolkodik, azon, hogy az adott játékos hogyan játszotta az eddigi leosztásokat.

Tehát a kérdés, hogy élőben vagy online játszunk, nem azért fontos, hogy látjuk-e ellenfeleink arcát. Teljesen más oka van, mégpedig a teremben lévő játékosok száma. Míg egy „élő" pókerteremben egyszerre legfeljebb néhány száz játékos tartózkodik, addig egy online terem befogadóképessége kvázi korlátlan. Némely online teremben egyszerre játékosok ezrei, tízezrei, akár százezrei is tartózkodhatnak egyszerre.

Így ha online kiesünk egy versenyből, nem kell aggódnunk, ugyanis valószínűleg a következő percekben indul egy új. Ezzel szemben ha élőben esünk ki, akkor vége van. Össze kell pakolni, és hazamenni, ugyanis nincs elég játékos, akivel új versenyt lehetne kiírni. Ebből kifolyólag élőben jellemzően óvatosabb stratégiát követnek a játékosok, míg online kevésbé van erre szükség.

Egy asztal, vagy több asztal? A póker jellemzően arról szól, hogy próbálunk kevesebbet hibázni, mint a többi játékos. Megpróbálunk nem hibázni, és közben azt várjuk, hogy ők viszont hibázzanak. Természetesen minél több játékos szerepel egy tornán, annál többen fognak hibákat elkövetni. Minél több ellenfelünk van, annál türelmesebbnek kell lenni. Minden kiesett játékossal közelebb kerülünk a fizetős helyekhez, így érdemes hagyni, hogy kiejtsék magukat.

Egy egy asztalos verseny esetén ez kissé változik: itt nem végzi el más helyettünk a munkát. Tennünk kell azért, hogy a gyengébb játékosok kiessenek. Jellemzően egy asztal esetén ezért agresszívabb stratégiát szoktak választani a profi játékosok, mint több asztalnál.

Milyen gyorsan nőnek a vakszintek? Vajon ki van jobb helyzetben egy leosztás során? Akinek kellett tétet betennie anélkül, hogy látta volna a lapjait, vagy az, aki ingyen nézhette meg őket? Természetesen az utóbbi. Nem szeretünk vakokat befizetni, főleg nem akkor, ha ezek magasak.

Minél gyorsabban nőnek a tétek, annál kevesebb időnk van arra, hogy az ellenfeleink hibáit várjuk. Pókeres körökben ezt a folyamatot úgy nevezik, hogy „megesznek a vakok". Ezért rövid vakszinteknél is agresszívabb stratégiára van szükség, míg hosszúaknál lehetünk türelmesebbek.

Hányan ülnek a mi asztalunknál? A stratégiánk egyik alapja, hogy a lehetséges lapkombinációk mekkora részével vagyunk hajlandóak játszik, mik azok, amiket biztosan eldobunk. Ha 10 játékos ül az asztalnál, úgy

egy 'A5' kombináció sokkal gyengébbnek számít, mintha 2 ellenfelünk lenne, lévén, hogy nagyobb eséllyel kapott valaki jobb lapot, ha sok játékos van.

Így minél kevesebb játékos ül egy asztalnál, annál több lappal játszhatunk. Ha egyetlen egy ellenfelünk van, sokszor érdemes minden lappal játszani. Ez az ellenfelünk erősségétől függ, ami a dolgozat szempontjából a legérdekesebb kérdés.

1.3.2. Egy pókerprogram célja

Milyen erősek az ellenfeleink? Attól függően, hogy hol és mit játszunk van rá esély, hogy megválaszthatjuk ellenfeleinket, vagy ha azt nem is, de tudjuk befolyásolni azt, hogy milyen erősségű ellenfelekkel játsszunk, például azzal, hogy a nap mely szakában ülünk asztalhoz.

Adódik hát az a kérdés, hogy milyen erős ellenfél ellen szeretnénk játszani? Ahhoz viszont, hogy erre választ kapjunk, újabb kérdésre kell válaszolnunk: miért játszunk?

Ha mi szeretnénk a legjobbak lenni, akkor érdemes minél erősebb ellenfeleket keresni. Ahhoz viszont, hogy valaki megélhessen a pókerből, nincs szükség arra, hogy ő legyen a legjobb. Bőven elég, ha jobb, mint az átlag, és tudja, hogyan maximalizálhatja várható értékét az átlagos játékosok ellen.

Ha a statisztikákat nézzük, úgy bizonyítást nyer az egyébként logikus megállapítás: ha gyengébb ellenfelek ellen játszunk, úgy sokkal többet nyerhetünk, mint erős ellenfelek ellen. Egy profi pókerjátékos számára ebből kifolyólag jobban megéri a gyengébb játékosok elleni stratégiáját fejleszteni, hisz a profitjának nagy része onnan származik.

Ismét adódik egy kérdés. Ha egyből a legjobb játékost tanuljuk meg legyőzni, úgy nem ütöttünk két legyet egy csapásra? Nem. Teljesen más stratégiát érdemes alkalmazni profi és hobbi játékosok ellen. Ebből a szempontból van egy fontos kulcsfogalom: a *kihasználhatóság*. Ez annyit tesz, hogy mennyire könnyű észrevenni az adott stratégia gyengeségeit, és azokkal visszaélve legyőzni azt. Vajon egy stratégia annál jobb, minél kevésbé kihasználható? Nem feltétlenül. Ugyanis míg egy profi játékos könnyebben ismeri ki más játékosok stratégiáját, addig a hobbijátékosokra ez a legkevésbé sem jellemző. Éppen ezért, ha profi játékosok ellen játszunk, a legfontosabb az, hogy kiismerhetetlenek legyünk. Hobbijátékosok ellen viszont úgy érhetünk el a lehető legjobb várható értéket, ha a lehető legkeményebben használjuk ki az ő hibáikat akár annak az árán is, hogy a mi stratégiánk is könnyen kiismerhető lesz. Hosszútávon ezt szokták a legjobb stratégiának tartani.

Fontos látni, hogy ez a kettő nem megy együtt. Nem lehet tökéletesen kihasználni a másik hibáit, és mellette tökéletesen kiismerhetetlennek lenni.

1.3.1. Példa. *Ha azt látjuk, hogy ellenfelünk az esetek 70%-ban dob, ha flopon emelünk, akkor könnyen adódik, hogy emeljünk mindig flopon. Ekkor viszont ő észreveheti ezt, és módosíthatja a stratégiáját. És itt jön a fontos kérdés, hogy milyen erős az ellenfelünk? Ha erős, akkor természetesen hiába látjuk, hogy sokat dob, nem engedhetjük meg, hogy 100%-ban emeljünk, mert azt ő azonnal kihasználja majd. Viszont egy gyengébb játékos ellen hiba nem 100%-ban emelni, mert attól gyenge játékos, hogy kevésbé tud alkalmazkodni, így nem kell attól tartanunk, hogy kihasználja ezt a gyengeségünket.*

Teljesen más a kérdés, ha egy programot szeretnénk készíteni. Jellemzően az online pókertermek tiltják a programok általi játékot élő játékosok ellen, így megélhetésre nem lehet egy ilyet használni. (A szerzőnek nincs tudomása olyan teremről, ami ne tiltaná.) Kutatási szempontból annál inkább. A póker játék megfejtése hatalmas segítség lehet más nem teljesinformációs területeken is, mint a tőzsde, hírszerzés, háborúk, kommunikáció, politika, stb.

A kutatók célja jellemzően a legjobb játékos legyőzése szokott lenni. Ha pedig a legjobbak kívánunk lenni, úgy adja magát a cél, hogy egy Nash-egyensúlyi állapotot közelítsünk. Ha jobban tudjuk ezt közelíteni, mint ellenfelünk, akkor ő nehezebben tudja majd kiismerni a stratégiánkat, mint mi az övét.

A későbbiekben bemutatok majd egy programot, ami a jelenlegi tudásunk szerint eddig a legjobban közelítette a Nash-egyensúlyt a pókerprogramok közül.

1.4. Kiértékelés

Nehezíti a kiértékelést, hogy a játékot nem egységes tétben játsszák. Mivel a nagyvak lehet 1 Ft, 10 Ft vagy 1000 Ft is, általában a nyereményeket nagyvak/játék alakú mértékegységgel jellemzik az összehasonlíthatóság végett.

A pókeres közösségekben a legelterjedtebb mértékegység a bb/100, (bb, mint big blind, azaz nagyvak), ami azt mutatja meg, hogy 100 leosztás alatt hány nagyvakot nyert a játékos. A kutatók ezzel szemben az *mbb/g (mili-big-blind per game)* mértékegységet használják.

1.4.1. Definíció. (Mili-big-blind per game (mbb/g)) Azt mutatja meg, hogy ezer leosztás alatt átlagosan hány nagyvakot nyer egy játékos. (Vagy egy leosztás alatt hány ezrednagyvakot.)

Ha minden leosztásban automatikusan eldobjuk lapjainkat, az átlagosan

−750 mbb/g, ugyanis 500-at veszítünk kisvakként, 1000-et pedig nagyvakként.

Pókerversenyzői körökben azt mondják, hogy 50 mbb/g már marginális különbség két játékos között.

1.5. Modellezési nehézségek

Ahhoz, hogy egy valóéletbeli problémára matematikai megoldást találjunk, elsők között modelleznünk kell az adott feladatot, azaz lefordítani azt a matematika nyelvére.

Egy játék modellezésének kulcsfontosságú lépése ábrázolni az állapotterét. Ezt körökre osztott játékoknál legegyszerűbben gráfok segítségével tehetjük meg. A póker szabályzatából következik, hogy az ábrázoló gráf véges lesz, és nem fog köröket tartalmazni.

Az állapottér gráfot célszerű faként ábrázolni. Így kapjuk meg egy játék játékfáját.

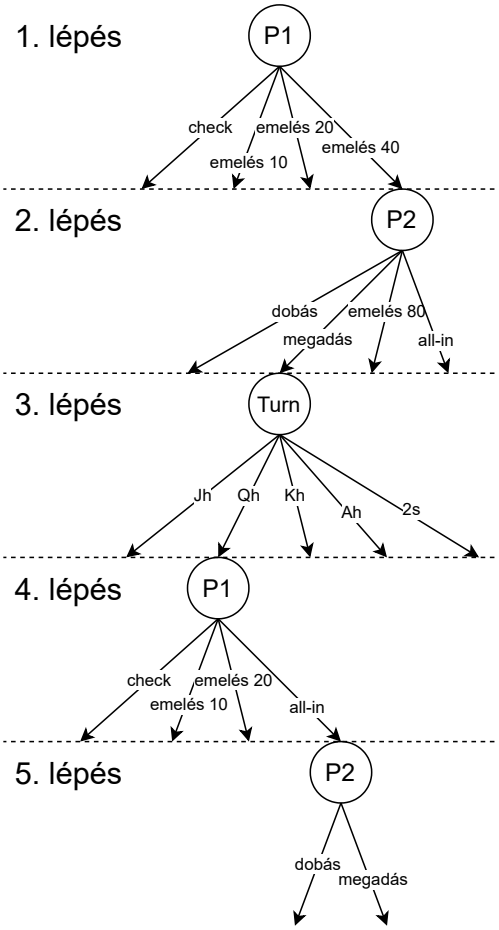
1.5.1. A Heads Up NLTH játékfája

1.5.1. Definíció. Egy olyan gráfot, ahol a fa csúcsai a játék lehetséges állapotait, a csúcsokból induló élek pedig az adott állapotból végrehajtható akciókat jelöli, ahol a lehetséges akciókat három csoportra oszthatjuk:

- a saját akciónk (saját stratégiánk szerint),
- az ellenfelünk akciója (ellenfelünk stratégiája szerint),
- a közös lapok kiosztása (véletlen szerint),

a Heads Up No Limit Texas Hold'em játéfájának hívunk.

A 1.1 ábrán egy leosztás játékfájából kiragadott rész látható. 1. lépés: épp a flopon vagyunk, az első játékos (P1) jön. Ebből az állapotból annyi állapot következhet, ahányféle akciót hajthat végre a játékos. Ezek közül 4-et tüntettem fel, de természetesen sokkal több van. (Dobás, checkelés és az összes lehetséges emelés.) Itt P1 40-et emelt. 2. lépés: a második játékos (P2) jön, akinek hasonlóan sok akciója van. P2 megadta az emelést. 3. lépés: a turn következik, ahol a lehetséges akciók a pakliban lévő lapok, amik leosztásra kerülhetnek. Esetünkben a kőr dáma kerül leosztásra, amit az angol Qh-nak rövidít (Queen of heart). 4. lépés: P1 all-in megy. 5. lépés: P2



1.1. Ábra. Példa egy játékfa részletére a pókerben.

állapotában vagyunk, aki dönthet, hogy dob, vagy megad. A játék dobás esetén végetér, és P1 megnyeri a potot. Megadás esetén a river lap következik, majd licitkör híján mutatás, és az erősebb kéz nyer.

A játékfa szélességét az egy állapotból végezhető lehetséges akciók száma, illetve a lehetséges közös lapok befolyásolják, míg a mélységét az egymást követő licitek, licitkörök.

Összességében a kétszemélyes NLTH egy nagyságrendileg 10^{160} méretű döntési fával büszkélkedhet, ami azt jelenti, ennyiféle döntési helyzet merülhet fel egy leosztás során. A japán go hasonló, 10^{170} méretű fával rendelkezik, ám hatalmas különbség, hogy míg az egy teljesinformációs játék, addig a póker nem. Ez számos problémát felvet.

1.5.2. Nem teljesinformációs játékok modellezése

Egy teljesinformációs játékra igaz az, hogy egy adott állapotában az optimális stratégiát nem befolyásolja az, hogy hogyan jutottak el a játékosok az állapotba.

1.5.1. Példa. 1.5.2. Példa. (Sakk) *Az 1.2 ábrán látható sakkjátszma során ennek az állapotnak az értéke „világos mattot ad 2 lépésben" minden tényezőtől függetlenül.*

1.2. Ábra. Egy sakk állás értéke. Világos lép, mattot ad két lépésben.

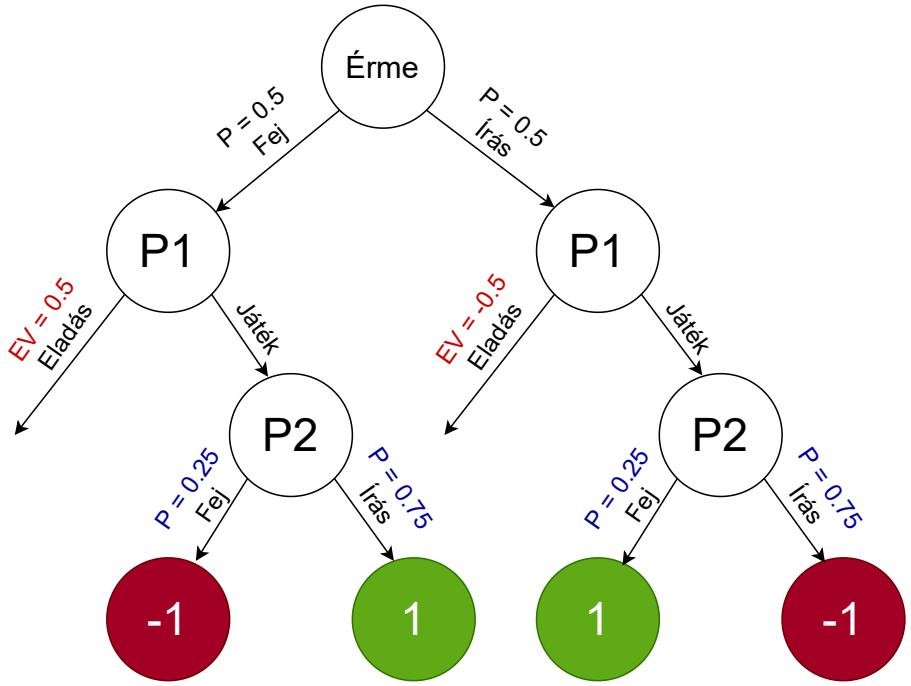
Nem teljesinformációs játékoknál ez nem így működik. Itt az optimális stratégiára olyan részei is hatással lehetnek a játékfának, ahová nem jutottunk el a játék során. Maga a tény, hogy eljuthattunk volna, súlyos tényező.

1.5.3. Példa. (Pénzfeldobás+) *Az 1.3 és az 1.4 ábra egy pénzfeldobáson alapuló játékot szemléltet, amit Noah Brown mutatott be a Libratus nevű pókerprogramot bemutató videójában.*[2]

A játék menete a következő. Az 1-es játékos (P1) feldob egy pénzérmét, amit P2 nem lát. Ezután P1 dönthet, hogy eladja az érmét, vagy tovább játszik. Amennyiben eladja, úgy egy olyan aljátékba jut, aminek elég tudnunk a várható értékét. Fej esetén ez $+0.5$*, míg írás esetén* -0.5*.*

[2]https://www.youtube.com/watch?v=McV4a6umbAY

Ha viszont játszik, akkor P2 következik. Meg kell tippelnie, hogy az érme fej, vagy írás lett az alapján, hogy tudja, hogy P1 a játékot választotta az eladás helyett. Ha mindig ugyanazt választja P2, akkor P1 könnyedén ki tudja használni ezt úgy, hogy mindig ő nyerjen, ám ha az 1.3 ábra alapján keveri őket, akkor P2 egy Nash-egyensúlyi állapot szerint játszik.

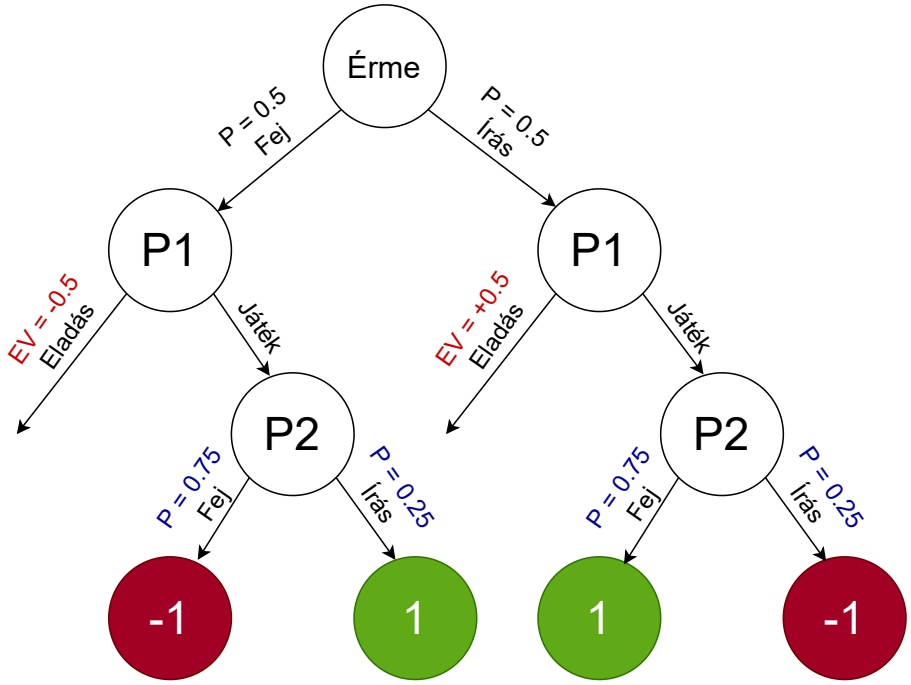


1.3. Ábra. Egy pénzfeldobás alapú játék Nash-egyensúlya, ahol fej esetén az eladás várható értéke $+0.5$, az írásé pedig -0.5. Ekkor P2 játékos Nash-egyensúly szerint játszik, ha 25%-ban választ fejet és 75%-ban írást.

Ha az eladás aljáték várható értékeit az 1.4 ábra szerint megcseréljük, és a fej várható értéke lesz -0.5, az írásé pedig $+0.5$, akkor az egyensúlyi állapot is annyiban módosul, hogy fejet kell majd 75%-ban választani, írást pedig 25%-ban.

Ez azt jelenti — és ez a legfontosabb része a példának —, hogy a nem teljesinformációs aljáték optimális stratégiája úgy változott, hogy egy olyan aljáték várható értékét módosítottuk, amihez el sem jutottunk az adott játék során. Magyarul más aljátékok kihatással vannak ennek az aljátéknak az optimális stratégiájára. Ez teljesinformációs játéknál nem történik meg.

Másik különbség, hogy a teljesinformációs játékokban egy állapotnak egyértelmű „értéke” van. A sakkban ezt valós számmal jelzik, és arra utal, hogy



1.4. Ábra. Az 1.3 ábrán bemutatott pénzfeldobás alapú játék Nash-egyensúlya, amennyiben felcseréljük a várható értékeket a fej és írás eladása esetén, azaz a fej eladása -0.5, az írásé pedig $+0.5$. Ekkor P2 játékos akkor játszik Nash-egyensúly szerint, ha 75%-ban választ fejet és 25%-ban írást.

az egyik fél hány „gyalognyi” előnyben van. Egy +5.0-s állás például arra utal, hogy az állapot olyan, mintha világosnak 5 gyaloggal vagy esetleg egy bástyával több figurája lenne, mint sötétnek. (Fontos megjegyezni, hogy ez az érték csak „utal”, ugyanis a bábuk aktuális pozíciója is beleszámít az értékbe, tehát anyagi egyenlőség mellett is előfordul +5.0-s állás.)

Ezzel ellentétben a nem teljesinformációs játékok állapotainak értéke mindig függ az ellenfél vélt vagy tudott stratégiájától. Ezáltal nem jellemezhetőek egy általános értékkel, hisz számolnunk kell azzal, hogy az ellenfelünk bármikor adaptálódhat a mi stratégiánkhoz.

Ezek mellett a nehézségek mellett a kétszemélyes NLTH játékfája túl nagy ahhoz, hogy teljesinformációs játékokhoz hasonlóan megoldható legyen, ezért a fa méretét leggyakoribb megoldásként csökkenteni szokták.

1.5.3. A játékfa csökkentése

A kétszemélyes NLTH játékfájának csökkentésére két bevett módszer az absztrakció és mélységi korlátozás.

Absztrakció Annyit tesz, hogy a valós játék helyett egy hozzá hasonló, ám lényegesen kisebb játékot oldunk meg, majd ezt az eredményt használjuk fel a valós játék megoldásához.

Absztrakciós lehetőségek:

- **Kártya absztrakció**: az összes lehetséges kártyakombináció figyelembevétele helyett azokat csoportokra, klaszterekre osztjuk, majd ezekkel számolunk.
- **Akció absztrakció**: bár a póker szabályai szerint a minimális emelés a nagyvak mértéke, a maximális a játékos összes zsetonja, és a két érték közt bármely egész számnyi emelés elfogadott, ezt korlátozni lehet például pot méretű, két pot méretű és all-in emelésre.
- **Állapot absztrakció**: egy teljes állapot helyett egy egyszerűbb állapot megoldása, ahol figyelmen kívül hagyunk bizonyos információkat, például az ellenfelünk múltbeli akcióit.

Mélységi korlátozás Amikor egy nagyméretű játékfán számolunk, minden egyes új szinttel exponenciálisan növekszik a számításhoz szükséges erőforrás. Ezt mind teljes és nem teljesinformációs játékoknál úgy szokták kezelni, hogy megszabnak egy határt, ami mélységen túl nem folytatják a számolást, hanem abban az állapotban egy becsült értékkel számolnak tovább. (A Deepstack program például, amit a 2. fejezetben részletesen bemutatok, 4 akciónyi mélységig számol, ezzel 10^{160}-ról 10^{17}-re csökkenti a játékfa méretét. [11])

1.6. Megoldás Counterfactual Regret Minimization segítségével

A Counterfactual Regret Minimization (CFR) (magyarul „utólagos megbánás minimalizálása”) algoritmus már több, mint tíz éve képezi az alapját a pókerprogramoknak. Az algoritmus kiszámolja, hogy különböző stratégiákat mennyire bánunk meg, majd azok közül a minimálisat kiválasztva megadja a

legjobb stratégiát az adott helyzetekben. Ahhoz viszont, hogy erről beszélhessünk, be kell vezetnünk több fogalmat. Ezt most a Zinkevich és társai által írt cikk [15] alapján teszem meg.

1.6.1. Definíció. (Extenzív nem teljesinformációs játék) Egy *extenzív nem teljesinformációs játék* fogalmát a következő elemekkel építjük fel:

- legyen N véges számú *játékosok* egy halmaza;
- legyen H az összes lehetséges véges számú akciósorozatokat tartalmazó halmaz, úgy, hogy az üres sorozat is eleme, és egy $h \in H$ akciósorozat minden prefixe is H eleme;
- jelölje $Z \subseteq H$ azon akciósorozatokat, amelyek végállpotokban érnek véget (azaz olyan sorozatokból álló részhalmaza H-nak, amik nem előtagjai egy másik sorozatnak sem);
- jelölje $A(h) = \{a : (h, a) \in H\}$ a $h \in H$ akciósorozat lejátszásának végén a lehetséges akciókat;
- legyen egy P (mint „player") játékos függvény, amely minden $h \in H \setminus Z$ sorozathoz hozzárendeli a soronkövetkező játékost az $N \cup \{c\}$ halmazból, akit $P(h)$ jelöl. Ha $P(h) = c$, akkor a véletlen dönti el a következő akciót, ami póker esetén a saját és közös kártyák kiosztását jelenti;
- egy f_c függvény, ami minden olyan $h \in H$ akciósorozathoz, amelyre $P(h) = c$, hozzárendel egy $A(h)$-n értelmezett $f_c(\cdot|h)$ valószínűségi mértéket, ahol ezek a mértékek függetlenek egymástól. ($f_c(a|h)$ tehát annak a valószínűsége, hogy a akció történik, feltéve, hogy h akciósorozat végén állunk);
- minden $i \in N$ játékoshoz tartozik egy $\mathcal{I}_i$ partíció $\{h \in H : P(h) = i\}$ elemekből, azzal a tulajdonsággal, hogy $A(h) = A(h')$ minden olyan h és h' párra, amik a partíció ugyanazon részhalmazának elemei. Minden $I_i \in \mathcal{I}_i$ esetén $A(I_i)$ jelöli az $A(h)$ halmazt, $P(I_i)$ pedig a $P(h)$ játékost minden $h \in I_i$ akciósorozatra. $\mathcal{I}_i$-t ekkor *információ partíciónak*, I_i-t pedig *információ halmaznak* nevezzük;
- minden $i \in N$ játékoshoz tartozik továbbá egy u_i hasznossági függvény, ami H végállapotaihoz (Z-khez) egy valós számot rendel.

1.6.2. Megjegyzés. Ha $N = \{1, 2\}$ és $u_1 = -u_2$, akkor egy kétszemélyes nulla-összegű játékról beszélünk.

1.6.3. Definíció. (Stratégia) Egy olyan σ_i függvényt, amely minden $A(I_i)$-n hozzárendel egy eloszlást minden $I_i \in \mathcal{I}_i$-hez i *játékos stratégiájának* nevezünk. Σ_i az i játékos összes stratégiáját tartamazó halmaz.

Stratégia profilnak nevezzük azt a σ halmazt, ami tartalmazza az összes játékos stratégiáját $(\sigma_1, \sigma_2, ...)$, σ_{-i} pedig az összes σ-beli stratégiát jelöli σ_i kivételével.

1.6.4. Jelölés. (Egy h akciósorozat valószínűsége) Jelölje $\pi^\sigma(h)$ annak a valószínűségét, hogy a h akciósorozat történik meg úgy, hogy a játékosok a σ stratégiát követik. Ekkor a π^σ szétszedhető úgy, hogy $\pi^\sigma = \Pi_{i \in N \cup \{c\}} \pi_i^\sigma(h)$, ahol $\pi_i^\sigma(h)$ az egyes játékosok választásainak a valószínűségei. $\pi_{-i}^\sigma(h)$ ekkor az i-n kívüli játékosok döntéseinek valószínűségeinek a szorzata.

$\pi_i^\sigma(I)$ és $\pi_{-i}^\sigma(I)$ a fentiekhez hasonlóan definiálható.

1.6.5. Jelölés. (Egy I információhalmaz valószínűsége) $\pi^\sigma(I)$ jelöli annak a valószínűségét, hogy elérjük az I információhalmazt egy adott σ stratégia mellett.

Egy *stratégiaprofil értékét* az i játékos számára a következő képlet írja le:

$$u_i(\sigma) = \sum_{h \in Z} u_i(h) \pi^\sigma(h).$$

Azaz minden végállapotra szummázuk a hasznosságát beszorozva az odajutás valószínűségével.

Ezen definíciók és jelölések segítségével az 1.2 alszekcióban már definiált Nash-egyensúly a következőképp írható le formálisan:

1.6.6. Definíció. (Nash-egyensúly)

$$u_1 \geq \max_{\sigma_1' \in \Sigma_1} u_1(\sigma_1', \sigma_2),$$

$$u_2 \geq \max_{\sigma_2' \in \Sigma_2} u_1(\sigma_1, \sigma_2').$$

Ezek után bevezethetjük a *regret*, vagyis *megbánás* fogalmát. Ez azt mutatja meg, hogy mennyire bántuk meg egyes múltbeli döntéseinket. Ezek alapján szeretnénk a jelenben és jövőben olyan döntéseket hozni, hogy ez a megbánás minimális legyen.

1.6.7. Definíció. (i játékos átlagos megbánása T időpillanatban)

$$R_i^T = \frac{1}{T} \max_{\sigma_i^* \in \Sigma_i} \sum_{t=1}^{T} (u_i(\sigma_i^*, \sigma_{-i}^t) - u_i(\sigma^t)).$$

Tehát vesszük annak a különbségét, hogy mi egy adott stratégiát, rajtunk kívül mindenki pedig a σ^t stratégiát játssza, azzal, hogy velünk együtt mindenki a σ^t stratégiát játssza. Ezt szummázzuk minden időpillanatban, majd ezek közül kiválasztjuk a maximálisat, és hogy átlagot kapjunk, osztjuk a vizsgált időpillanatok számával.

Mostmár bevezethetjük a *counterfactual regret*, vagyis az *utólagos megbánás* fogalmát. Az a koncepció, hogy a fent kapott T időpillanatbeli megbánást felosztjuk kisebb, összeadható részekre, és azokat minimalizáljuk. Ezeket különálló információ halmazokon definiáljuk.

Kezdésnek veszünk egy $I_i \in \mathcal{I}$ információhalmazt, és az azon történt akciókat.

1.6.8. Jelölés. (h akciósorozat hasznossága) Legyen $u_i(\sigma, h)$ a várható hasznosság h-ban, amikor mindenki a σ stratégia szerint játszott.

1.6.9. Jelölés. (h'-be való jutás valószínűsége) Legyen $\pi^\sigma(h, h')$ a valószínűsége annak, hogy σ stratégia mellett h-ból h'-be jutunk.

Ezek segítségével definiálhatjuk a *counterfactual utility*-t, azaz az *utólagos hasznosságot*.

1.6.10. Definíció. (Utólagos hasznosság) Legyen $u_i(\sigma, I)$ az *utólagos hasznosság* az I információhalmazon, amikor minden játékos a σ stratégia szerint játszott, kivéve i, aki minden döntését úgy hozta, hogy I-t érjük el. Értékét a következő képlet adja meg:

$$u_i(\sigma, I) = \frac{\sum_{h \in I, h' \in Z} \pi^\sigma_{-i}(h)\pi^\sigma(h, h')u_i(h')}{\pi^\sigma_{-i}(I)}.$$

Végül definiáljuk $\sigma|_{I \to a}$-t úgy, hogy ez megegyezzen a σ stratégiával, azt kivéve, hogy I-t elérve i játékos mindig az a akciót fogja választani. Ekkor az azonnali utólagos megbánás a következőképp írható fel:

1.6.11. Definíció. (Azonnali utólagos megbánás)

$$R^T_{i,imm}(I) = \frac{1}{T} \max_{a \in A(I)} \sum_{i=1}^{T} \pi^{\sigma^t}_{-i}(I)(u_i(\sigma|_{I \to a}, I) - u_i(\sigma^t, I)).$$

Tehát annak a hasznosságnak, hogy I-ben a-t lépünk vesszük a különbségét azzal a hasznossággal, hogy a σ^t stratégia szerinti akciót hajtjuk végre, ezt megszorozzuk azzal a valószínűséggel, hogy i játékoson kívül mindenki σ stratégiát játszva I-be jutottunk, ezeket szummázzuk, kiválasztjuk a lehetséges akciók fényében a maximális hasznosságot, majd osztva az időpillanatok számával átlagot kapunk.

1.6.12. Tétel. *Az azonnali utólagos megbánás értéknek a minimalizálásával megfelelő ismétlésszám mellett Nash-egyensúlyi állapotot közelítünk. [15]*

Tehát a CFR algoritmus megfelelő eszköz ahhoz, hogy Nash-egyensúlyi állapotot közelítsünk.

A CFR algoritmust egy játék végállapotáig futtatni minden alkalommal irreálisan drága lenne, ezért korlátozzuk, hogy milyen mélységig számolhat, ha pedig ezt a mélységet elérte, akkor ott egy becsült értékkel fog találkozni, ami a játék további részének az utólagos megbánását becsüli meg. Erre pedig egy lehetséges megoldás neurális hálózatokat használni.

2. fejezet

Deepstack — egy modell neurális hálózatokkal

A NLTH játék modellezésének célja, hogy adott rendelkezésre álló információk mellett a modell által irányított ágens a lehető legprofitábilisabb akciót hajtsa végre. Lévén, hogy modellünkben az utólagos megbánást minimalizálva keressük a legmegfelelőbb akciót, a célunk az, hogy minél pontosabb becslést adjunk a különböző akciók utólagos megbánási értékeire.

Mivel a játékfa praktikusan kezelhetetlenül nagy, ezért kimerítő leszámolások és pontos számítások helyett közelítéseket és becsléseket vagyunk kénytelenek alkalmazni. Egy lehetséges választás, hogy a modellezéshez használt hasznossági függvényeket mesterséges neurális hálózatokkal közelítsük.

Ebben a fejezetben egy ilyen mesterséges neuronhálón alapuló modellt mutatok be: a Deepstack modellt [11], amely az eddigi legalacsonyabb kihasználhatósággal rendelkező NLTH modell.

A Deepstack modellben sűrű előrecsatolt neurális hálózatokat használnak, a következő definícióban ezt vezetem be.

2.0.1. Definíció. (Sűrű, előrecsatolt neurális hálózat) Sűrű, előrecsatolt neurális hálózatnak nevezünk egy olyan $f_\theta(x) = f_{\theta_L} \circ \cdots \circ f_{\theta_2} \circ f_{\theta_1}(x)$ $\theta = (\theta_1, \theta_2, \ldots, \theta_L)$ paraméterekkel parametrizált függvénykompozíciót, ahol az f_θ függvénykompozíció elemei a hálózat rétegei, és a következő alakban adottak: $f_{\theta_i}(x_i) = \sigma(A_i x_i + b_i)$ $\forall i \in [1..L]$, ahol $x_i \in \mathbb{R}^n$ az i. réteg bemeneti vektroa, $A_i \in \mathbb{R}^{n \times m}$ az i. réteg neuronjainak súlymátrixa, $b_i \in \mathbb{R}^m$ az i. réteg neuronjainak eltolásvektora, $\theta_i = (A_i, b_i)$ az i. réteg súlyai, σ az ún. aktivációs függvény, $L \in \mathbb{N}$ pedig a hálózat rétegeinek száma. [6]

Az utólagos megbánással való modellezés során egy lehetséges út, hogy egy vagy több neuronháló végez minden becslést, azonban a gyarkolatban gyakran hatékonyabb, ha a neuronhálót csak bizonyos játékfa mélyég esetén

alkmalmazzuk. Nevezetesen akkor, amikor a CFR algoritmust már túlságosan idő- és erőforrásköltséges lenne használni.

Voltaképpen pontosságot cserélünk el időre, ám ahhoz, hogy a programunk élő játékossal élő időben tudjon játszani, ez a csere elengedhetetlen. A fő kihívás tehát az, hogy élő időben hozzon az algoritmus olyan döntéseket, ami versenyre kelhet professzionális emberi játékosokkal.

2.1. A Deepstack modell

A Deepstack modell a póker játékfáját használja modellezésre. A játékfa méretének csökkentése érdekében absztrakciót alkalmaz, a megoldás közelítéséhez, a legjobb stratégia kiválasztásához pedig mélységkorlátolt CFR algoritmust használ neurális hálózatokkal.

Az eddigi póker programokkal ellentétben a Deepstack nem előre kiszámolt stratégiát használ, hanem minden problémát akkor old meg, amikor az a játék során felmerül, és ezt mindig az összes elérhető információ tudatában teszi, nem azokból kiemelve.

Hogy időt és erőforrást takarítsanak meg, egy bizonyos mélység után abbahagyják a számolást, nem számolják ki végig a játékot, hanem egy gyors becsléssel élnek. Ezt a becslést a Deepstack „intuíciójának" nevezik, az emberi intuícióra utalva, és apellálva, hogy mesterséges neuronhálót alkalmaznak a modellben a becslésekhez. A külső hatások miatt valami úton-módon születik egy reakció. A Deepstacknél ugyanúgy, mint az embernél annál jobb az intuíció, minél több tapasztalatunk van (egy bizonyos szintig), így akár azt is mondhatnánk, hogy az intuíciót tanítani kell.

2.2. A Deepstack felépítése

A Deepstack felépítésének három sarokpontja:

- folytonos újramegoldás,
- utólagos megbánás becslése mélységkorláttal,
- a játékfa ritka modellezése.

A következőkben ezeket a modellezési elemeket mutatom be.

Folytonos újramegoldás A Deepstack nem egy előre megkonstruált és eltárolt stratégiát alkalmaz, hanem újraszámolja a stratégiát minden egyes akció előtt. Ehhez szükséges tudni a saját laptartományunkat az aktuális állapotban és egy vektort az ellenfelünk utólagos megbánási értékeivel minden lehetséges lapkombinációja szerint. Ezek az értékek felsőkorlátokként megadják a maximális hasznosságot, amit az ellenfelünk az adott lappal elérhet. Mivel a CFR algoritmus szintén utólagos megbánási értékekkel dolgozik, így megfelelő módszer az ezen állapotokban történő számolásokhoz.

Ahhoz viszont, hogy a stratégiát minden akciónál újraszámítsa, szükség van egy kezdeti stratégiára, amiből kiindulhat az algoritmus. Mivel a Deepstack nem használ előre számolt stratégiát, ezért ezt úgy hidalja át, hogy minden egyes stratégiát csak és kizárlóag az aktuális akcióhoz használ fel, majd az egészet újraszámolja módosított értékekkel.

Mivel a játék elején semmilyen információ nincs az ellenfél laptartományáról, ezért az uniform eloszlás szerint tárolja őket, és az utólagos megbánási értékeket a saját lapja ellenében vett értékekkel inicializálja minden egyes lapkombináció esetében. Ezután ha ő van soron, ezen értékek segítségével megoldja a részfát (ágat), majd az eredmény szerint meghozza a megfelelő akciót. Minden egyes játékos által tett akció után, vagy ha lekerül egy új közös lap, módosítja a tárolt értékeit a következő elvek alapján:

- Saját akció: frissíti az ellenfél utólagos megbánási értékeit a számolt stratégia alapján az új akció esetén. Frissíti a laptartományát a számolt stratégia és a Bayes-szabály alapján.
- Ellenfél akciója: sem a saját laptartományát, sem az ellenfél értékeit nem kell módosítani.
- Új lap: frissíti a saját laptartományát kivéve belőle azokat a kombinációkat, amiket az új lap lehetetlenné tett, és frissíti az ellenfél értékeit, az erre az esetre számolt stratégia alapján.

Ez a folytonos újramegoldás önmagában is megállná a helyét, de a gyakorlatban nem praktikus, ugyanis a játék végén kívül irreálisan nagy kapacitást igényelne az ágakat végigszámolni. Ennek a komplexitásának a csökkentésére egy lehetőség a mélységi és szélességi korlátozások alkalmazása.

Utólagos megbánás becslése mélységkorláttal Mint ahogy azokban a teljesinformációs játékokban is, ahol a játékfa nagysága miatt a közvetlen megoldás kiszámíthatatlan, itt is csökkenteni szeretnénk a kiszámolásra váró

fát. Ám a nem teljesinformációs játékoknál a korlát utáni ágakat nem helyettesíthetjük egyetlen előre kiszámolt értékekkel, ugyanis egy állapot értéke mindig függ az ellenfelünk aktuális stratégiájától.

A Deepstack ezért utólagos megbánási értékeket tartalmazó tanított értékfügvénnyel helyettesíti őket, ami közelíti azt a megoldást, amit akkor kaptunk volna, ha a jelenlegi laptartományok mellett megoldottuk volna azt az állapotot. Ehhez bemenetként olyan vektorokat használ, amik leírják a játék aktuális állapotát, kimenetként pedig olyan értékeket kap, amik megbecsülik a lehetséges lapok hasznosságát az aktuális állapotban.

A mélységet 4 akcióra korlátozva az egyébként 10^{160} döntési helyzetetet 10^{17}-re csökkenti. A Deepstack mély neurális hálókat használ értékfüggvény gyanánt.

A játékfa ritka modellezése A Deepstack harmadik sarokpontja a lehetséges akciók számának csökkentése. A Deepstack úgy építi fel a játékfában az ágakat, hogy csak a dobás, megadás, legfeljebb háromféle emelés, illetve all-in akciók lehetségesek. Erre azért van szükség, hogy a Deepstack emberi sebességben tudjon játszani akár egyetlen GPU segítségével is.

A ritkaságnak és mélységi korlátozásnak hála az ágak 10^7 döntési helyzetet eredményeznek, amin egy NVIDIA GeForce GTX 1080 GPU segítségével képes bármely döntést 5 másodperc alatt meghozni.

2.3. Absztrakció a Deepstackben

Bár a Deepstack is használ absztrakciót, teljesen más oldalról, mint az eddigi pókerprogramok. A Deepstack az ágak számítása során a lehetséges akciókat korlátozza, ám minden számítás a tényleges játékállapotból indul, nem egy absztrakt állapotból, így a Deepstack tökéletesen tisztában van az aktuális helyzettel.

Kártya absztrakciót is használ, ám ezeket a laptartomány klasztereket az utólagos megbánási értékek közelítéséhez használja, így nem korlátozza az információt arról, hogy a játékos milyen lapokkal rendelkezik.

Ezen felül pedig az algoritmusnak sosincs szüksége az ellenfél tényleges akcióihoz ahhoz, hogy nyomonkövethesse és tárolhassa az utólagos megbánási értékeket tartalmazó vektorát és a laptartományait, így nem szükséges azzal vesződni, hogy ezeket az akciókat is számontartsa az algoritmus.

Ezek a különbségek alacsonyabb kihasználhatóságot eredményeznek, ahogy a cikkjükben ezt bemutatják.

2.4. Neuronhálók a Deepstack modellben

A Deepstack modellben 3 különböző neuronhálót tanítanak be. Egyet preflop akciókhoz, egyet flophoz, egyet pedig turn-höz. Riverre már nem, mert a river állapotterén olcsón lehet tényleges megoldást számolni, lévén, hogy a mélység a leosztás végére jelentősen lecsökken.

Minden akció előtt végrehajtódik a rekurzív CFR számítás egy adott mélységig, ott pedig a megfelelő neuronháló által adott becsült értéket veszi, és annak segítségével állapítja meg az utólagos megbánási értéket. Mint már említettem, riveren neuronháló híján CFR segítségével történik végig az ekkor már pontos számítás.

A hálók működési elve viszonylag egyszerű. Vektorokként megkapják a megfelelő bemeneti értékeket, ezeken végrehajtanak egy függvényt, majd kimenetként visszaadják az utólagos megbánási értékeket.

Ahhoz, hogy ezek az értékek megfelelő közelítések legyenek, a hálókat mind tanítani kell. A neuronhálók tanításához a standard tanítási módszertant alkalmazzák, azaz a sztochasztikus gradiens ereszkedő módszer egy változatát, az Adam optimizáló algoritmust [9] használják Huber veszteségfüggvénnyel [7].

2.4.1. Az alkalmazott architektúrák

A DeepStack egy standard előrecsatolt hálózatot használ 7 sűrű réteggel, mindegyikben 500 neuronnal, parametrikus ReLU aktivációs függvénnyel.

Ez az egész bele van ágyazva egy külső hálóba, ami garantálja a nulla-összegűséget. A külső számítás ezután veszi a megbecsült utólagos megbánási értékeket, és súlyozottan szumázza őket a játékosok laptartományai szerint, ami a játék becsült értékét eredményezi külön-külön a két játékos számára. Ezután az összeg felét kivonja a két játékos becsült utólagos megbánási értékéből.

A neuronháló bemenetei:

- a pot mérete az összzsetonszámhoz arányosan viszonyítva,
- a játékosok laptartományai a közös lapokkal együtt. A laptartományok 1000 vödörbe vannak csoportosítva, és egy a vödrökön vett valószínűségek vektoraként szerepelnek a bemenetben.

A neuronháló kimenete:

- az utólagos megbánási értékek mindkét játékosra és lapjaikra, a pot méretéhez mérve arányosan megadva.

2.4.2. Turn-háló

Tanítás A turn-háló 10 millió random generált turn leosztás megoldásával lett tanítva. Ezek a leosztások random generált laptartományokkal, közös lapokkal és pot mérettel rendelkeztek. Az utólagos megbánási értékek minden tanuló leosztáshoz úgy lettek generálva, hogy 1000 CFR$^+$ iterációval megoldották a leosztásokat a játékosok akcióit limitálva dobásra, megadásra, pot méretű emelésre és all-inra, lap absztrakció nélkül. Ehhez 6144 CPU magot használtak a Calcul Québec MP2 klaszteren, 175 core year számítási idő alatt. (Ez annyit tesz, hogy 1 mag használata esetén 175 évbe telt volna a számítás).

Laptartomány reprezentáció A játékosok laptartományainak könnyebb általánosíthatósága miatt a lehetséges különböző leosztásokat vödrökbe csoportosítják, klaszterezik. Ezek a vödrök klaszterező absztrakcióval lettek létrehozva úgy, hogy a stratégiailag azonos lapokat k-közép segítségével és earth mover's távolsággal használva szétválogatták. Mind a turn-hálóhoz, mind a flop-hálóhoz 1000 darab vödröt használtak.

Pontosság A turn-háló a tanuló halmazon 0.016 potnyi Huber veszteséget ért el, a validációs halmazon pedig 0.026-ot.

2.4.3. Flop-háló

Tanítás A flop hálót hasonlóan tanították 1 millió leosztással, itt viszont az utólagos megbánási értékeket a mélység-korlátolt CFR-rel és a tanult turn-hálóval generálták. Ehhez 20 GPU-t és fél év GPU számítási időt használtak.

Laptartomány reprezentáció A turn-hálónál tárgyaltak alapján történik, azzal azonosan.

Pontosság A flop-háló a turn-hálóénál sokkal kisebb tanulóhalmazon 0.008 potnyi Huber veszteséget ért el, a validációs halmazon pedig 0.034-et.

2.4.4. Preflop-háló

Tanítás A preflop hálót szintén 10 millió leosztással tanították. Itt az utólagos megbánási értékeket úgy generálták, hogy mind a 22.100 lehetséges flopon átlagolták flop-háló által adott utólagos megbánási értékeket.

Laptartomány reprezentáció Megemlítendő különbség, hogy a többi hálóval ellentétben ennek a bemenetéhez nem volt szükség klaszterezésre, ugyanis a játék elején kapott két lapból álló kombinációkat csupán 169 (azaz 13^2) féleképpen lehet stratégiailag megkülönböztetni egymástól, így olcsóbb őket közvetlenül bevinni.

Pontosság A tanulóhalmazon 0.000053 potnyi Huber veszteséget ért el, a validációs halmazon pedig 0.000055-öt.

2.4.5. Egyéb technikai információk

A tanítás részletei Minden hálózatot a Torch könyvtárral tanítottak Adam optimalizálóval. A mini-batch méret 1000 volt, a tanulási ráta 0.001, amit az első 200 epoch után 0.0001-re csökkentettek. A hálózatokat nagyságrendileg 350 epochig tanították, ami egyetlen GPU-n két napos tanítást jelentett. A végső modellhez a legkisebb epoch végi validációs hibával rendelkező állapotot választották.

Rejtett rétegek száma Már korai kísérletek során látták, hogy a rejtett rétegek számának növelésével alacsonyabb validációs hibát tudnak elérni. A kísérletek hatására végül 7 réteg mellett döntöttek pontosságot sebességre és erőforrásra cserélve ezzel.

Mélyebb vizsgálatok viszont később megmutatták, hogy a turn-háló esetében például a validációs hiba az 5. rétegtől kezdve nem csökken, így a 7 réteg valójában több, mint szükséges.

Nem lenne nagy meglepetés azt látni, hogy nagyobb tanulóhalmaz mellett még tovább csökkenne a validációs hiba.

2.5. Kihasználhatóság

A kihasználhatóság, angolul exploitability azt jelenti, hogy mennyire egyszerű feltárni egy adott stratégia, vagy játékos hibáit, és azokat kihasználni.

2.5.1. Példa. (Kihasználhatóság a pókerben) *Nézzünk egy olyan stratégiát, ami minden kezdőlap esetén emel, ám visszaemelésre azonnal dob. Ha*

ezt nem vesszük észre, rengeteget veszíthetünk mindazon alkalommal, amikor mi döntöttünk a dobás mellett, így akár egy jó stratégia is lehetne, de ha észrevesszük, és alkalmazkodunk hozzá, akkor minden leosztás során mi nyerünk.

A hobbi és a profi játékosok közötti egyik legnagyobb különbség az, hogy ki mennyire képes észrevenni és kihasználni az ellenfele hibáit. A Deepstack fő célja egy Nash-egyensúly közelítése volt, azaz az, hogy minimalizálják a kihasználhatóságot.

LBR Bár a NLTH játékban a pontos kihasználhatóság kiszámíthatatlan, egy LBR nevezetű local best-response technikával[10] kapható egy alsó becslés az egyes stratégiák kihasználhatóságáról. Ehhez teljes hozzáférésre van szüksége az akciók valószínűségét tartalmazó vektorhoz, ami segítségével le tudja generálni a stratégia teljes tartományát bármely lehetséges állapothoz, ami közös lapokat is tartalmaz. Ezeket a tartományokat aztán arra használja, hogy feltételezve, hogy nem lesz több licitkör a játék során, meghozza a lehető legjobb válaszakciót egy előre meghatározott halmazból.

Eredmény Míg az ACPC-n szereplő absztrakció alapú programok az LBR szerint rettentően kihasználhatóak (négyszer rosszabb eredményt produkálnak, mintha minden leosztásban automatikus eldobnák a lapjaikat), addig a Deepstacket képtelen volt kihasználni az LBR. Mi több, átlagosan több, mint 350 mbb/g értékes veszteséget ért el az LBR a Deepstack ellen.

A szerzők értékelése szerint vagy egy sokkal szofisztikáltabb módszer szükséges a Deepstack hibáinak felderítéséhez, vagy valóban sokkal kevésbé kihasználható, mint elődei.

2.6. Kiértékelés

A Deepstack kiértékelését 30 professzionális játékos segítségével végezték, akiket az Internation Federation of Poker világszervezeten keresztül kértek fel. Ezektől a játékosoktól azt kérték, hogy játsszanak 3000 leosztást a Deepstackkel 4 hét leforgása alatt 2016 novemberében, ám ezt mindössze 11-en teljesítették.

A játékosokat pénzjutalommal ösztönözték. Az első 3 helyezést elérő játékos $5000, $2500 és $1250 kanadai dollárban részesült.

Kiértékelési nehézségek A póker játék népszerűségét adó rövidtávú variancia egyszerre áldás és átok. Áldás, mert játékosok millióit vonzza az asztalhoz, viszont átok, mert egy profi játékossal is megtörténhet (és meg is

történik), hogy egy hónapban, vagy akár hónapokon keresztül veszteséges a rövidtávú variancia miatt. A Claudico meccs[1] során látható volt, hogy 80.000 leosztás sem feltétlenül elég ahhoz, hogy marginális statisztikai különbséget tegyenek két eltérő tehetségű játékos között. Erre többféle megoldást is alkalmaznak a játékosok, amikre most nem térek ki.

AIVAT A kiértékeléshez az AIVAT módszert[2] használják, ami egy bizonyítottan torzítatlan, alacsony varianciás technika, ami megfelelő becslést ad nem teljesinformációs játékokon végzett teljesítményre gondosan konstruált kontroll változók segítségével.

A Deepstack értékfüggvénye tökéletesen megfelel az AIVAT-hoz, így a használatával teljesen torzítatlan teljesítménybecsléseket kaptak.

Az AIVAT segítségével már 3000 leosztás is elég ahhoz, hogy statisztikai szignifikanciával rendelkező eredményt kapjanak.

Eredmények Összesen 44.852 leosztást játszottak a játékosok, a 30-ból 11-en teljesítve a kért 3000 leosztást. Összességében a Deepstack 492 mbb/g átlagos nyereséget produkált. Az AIVAT segítségével látható volt, hogy a Deepstack kissé szerencsés volt, a korrigált érték végül 486 mbb/g lett. Itt érdemes újra megemlíteni, hogy a profi játékosok 50 mbb/g különbséget már jelentős különbségnek tekintenek, ám azt is, hogy ezek a játékosok nem a Heads Up játék specialistái.

A 11 játékosból, akik az összes leosztást lejátszották 10 ellen ért el statisztikailag szignifikáns győzelmet a Deepstack. Ez ellen a 11 játékos ellen az átlagos nyereség 394 mbb/g volt. Az első játékos ellen is 70 mbb/g eredményt ért el a Deepstack, ám ez nem volt statisztikailag szignifikáns különbség.

Érdekes információ még, hogy a Deepstack nem volt képes legyőzni a 2016-os éves pókerprogram (ACPC) verseny győztesét, a Tartanian 8-at. Bár fej-fej melletti teljesítményben elmaradt tőle, a kihasználhatósága minden eddigi programénál, így a Tartanian 8 modell értékénél is alacsonyabb volt.

[1] J.Wood, Doug Polk és csapata legyőzte Claudico-t, ezzel $100,000 díjazást nyerve a Microsofttól és a The Rivers Casino-tól, Pokerfuse, http://pokerfuse.com/news/media-and-software/26854-doug-polk-andteam- beat-claudico-win-100000-microsoft/ (2015).

3. fejezet

A Deepstack összevetése a Libratus-szal — egy csúcsteljesítményű modell neurális hálózatok nélkül

A Libratus[1] az első olyan modell, amely képes volt a világ legjobbjait is legyőzni kétszemélyes NLTH játékban. A Libratus nem használ neurális hálózatokat, így a dolgozat eredeti célkitűzései szerint nem szándékozom részletezni, azonban úgy vélem, hogy a két modell összevetése egy érdekes zárása a dolgozatnak. Ebben a rövid fejezetben bemutatom a Libratus működését, illetve azt, hogy miben hasonlít és miben különbözik a Deepstacktől.

3.1. A Libratus felépítése

A készítők úgy fogalmaznak, hogy a Libratus stratégiáját nem programozták, hanem inkább generálták. Ugyanúgy 3 sarokpontja van, mint a Deepstacknek, ezek pedig a következők:

1. egy Nash-egyensúlyt közelítő előre kiszámított stratégia,
2. élő megoldásszámítás,
3. önfejlesztés.

Az első modul egy kisebb, absztrakt játékhoz generál egy optimális stratégiát, ezt követi a program az első két licitkör során. A maradék kettőre pedig a második modult alkalmazza úgy, hogy a várható érték semiképp se

legyen rosszabb annál, mint amit az első modul adna. A harmadik modul azért felel, hogy finomítsa az első modult.

A következő 3 szekcióban ezeket fogom röviden bemutatni.

3.1.1. Előre kiszámított stratégia

Mint azt fent tárgyaltuk, a játék túl nagy a direkt megoldáshoz, így a Libratus is absztrakciót alkalmaz ehhez a modulhoz.

Egyrészről akcióabsztrakciót alkalmaz, mégpedig úgy, hogy a lehetséges emelések közül csak néhányat választ ki, a többi emelési értéket pedig a hozzá legközelebb lévőhöz kerekíti. Ha például a $100-onként veszi az emeléseket, akkor ha az ellenfél $201-t emel, az algoritmus úgy veszi, mintha $200-t emelt volna.

Másrészről pedig kártyaabsztrakciót, ám ezt csak a harmadik és negyedik utcán teszi. Érdekes ismét kiemelni, hogy a harmadik és negyedik utcán valójában nem is ez a modul szolgáltatja a stratégiát, csupán biztosítékként szolgál, hogy a második modul semmiképp se adjon rosszabb várhatóértékű stratégiát annál, mint amit az első modul adna.

3.1.2. Élő megoldásszámítás

Ez a modul a harmadik utcán lép színre, és nem használ kártyaabsztrakciót, és az akcióabsztrakció során is több lehetőséget hagy bent.

Bemenetként az első modul által kiszámított ehhez az aljátékhoz tartozó értéket veszi, és azt a finomabb absztrakció mellett élőben továbbszámítja, így garantálva, hogy legrosszabb esetben is az első modul által javasolt stratégiát adja, annál rosszabbat nem. Ehhez a fent ismertetett CFR algoritmus egy fejlesztett változatát, a CFR$^+$-t használják.

3.1.3. Önfejlesztés

A háttérben egy algoritmus folyamatosan figyeli az ellenfél emeléseit az első licitkör során, hogy aztán néhány ez alapján választott elemmel bővítse az első modul által használt absztrakt modellt. Ha az ellenfél gyakran alkalmaz olyan emelési méreteket, amikre az alapmodul nem számolt ki stratégiát, akkor ez az algoritmus kiegészíti ezekkel az értékekkel az absztrakciót, így pontosabb játék érhető el az adott ellenfél ellen.

3.2. A tanítás részletei

A Libratus összesen 25 millió core-hour időt használt fel a tanuláshoz. (Azaz, ha egy processzormagot használtak volna, 25 millió órába, azaz nagyjából 2850 évbe telt volna.)

Ebből kísérletekre 13 milliót, az első modul számára 6 milliót, a második és harmadik számára pedig 3-3 milliót használt.

Az első és harmadik modult 196 darab 28 magos számítógépen tanították, amik egyenként 128GB memóriával rendelkeztek, a másodikat pedig 50 darab hasonló számítógépen.

Asszimetrikus absztrakciót használtak, az ágens ugyanis csak 14 magot használt a 28-ból, hogy az ellenfélnek többféle akcióra legyen lehetősége.

3.3. Kiértékelés

2017 januárjában a Libratust egy 4 főből álló csapattal tesztelték, akik a világ legjobb Heads Up NLTH játékosok közül kerültek ki. Ők Jason Les, Dong Kim, Daniel McCauley, és Jimmy Chou.

Mindannyian $20.000-t kaptak a szereplésükért, és további $120.000 pedig a teljesítményük szerint lett szétosztva.

Libratus 147 mbb/game nyereséget produkált a játékosok ellen, 99.98%-os statisztikai significancia szinttel, 0.0002 p-értékkel.

3.4. Összegzés

Mindkét program játékfát használ a modellezéshez, és absztrakciót a játékfa csökkentéséhez. Mindkettő használja a CFR algoritmust, ám a Deepstack neurális hálózatokat is használ, a Libratus viszont nem.

A Deepstack nem tárol előre kiszámított stratégiát, míg a Libratus igen, és azt a játék során szerzett tapasztalatok segítségével finomítja.

Míg a Deepstack tanításához kevesebb, mint 200 core-year időt használtak, addig a Libratuséhoz több mint 2850-et. Ez önmagában egy jelentős különbség.

Összefoglalás

Dolgozatomban a Heads Up No Limit Texas Hold'em játék gépi tanulásos modellezésével foglalkoztam.

Az első fejezetben elhelyeztem a játékot a játékelméleti térképen, betekintést nyújtottam emberi és gépi játékhoz való stratégiaalkotásba, bemutattam a játék modellezésének a nehézségeit, azokra prezentáltam a jellemző megoldási lehetőségeket és részletesen bemutattam a pókerjátékok modellezésének egyik fontos algoritmusát, a CFR algoritmust.

A második fejezetben bemutattam egy modellt, a Deepstack-et, ami neurális hálózatok segítségével modellezi a problémát. Bemutattam a felépítését, a modell álltal használt absztrakciókat, hogy hogyan használja a neurális hálózatokat, és a Deepstack által elért eredményeket.

Végül a harmadik fejezetben a Deepstack modellt összehasonlítottam a Libratus modellel, amely jelenleg a legjobb teljesítményt nyújtja kétszemélyes NLTH játékban. Röviden bemutattam a felépítését és a működését, a tanításának részleteit, az elért eredményeit, majd összegeztem a legnagyobb hasonlóságokat és különbségeket.

Dolgozatommal képet adtam arról, hogy miként lehet modellezni a Heads Up No Limit Texas Hold'em játékot. Feltérképeztem, hogy milyen viszonyban van egymással az emberi és a gépi játékhoz történő stratégiaalkotás, ám mint kiderült, a kettőt összehasonlítani olyan, mint almát a körtével. A pitébe mindkettő jó, mégis teljesen más a kettő.

Irodalomjegyzék

[1] Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

[2] Neil Burch, Martin Schmid, Matej Moravčík, and Michael Bowling. Aivat: A new variance reduction technique for agent evaluation in imperfect information games, 2017.

[3] Országh László Bárczi Géza. *A magyar nyelv értelmező szótára.* Akadémiai kiadó, 1962.

[4] Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. Deep blue. *Artificial intelligence*, 134(1-2):57–83, 2002.

[5] Xiaotie Deng and Fan Chung Graham. *Internet and Network Economics: Third International Workshop, WINE 2007, San Diego, CA, USA, December 12-14, 2007, Proceedings*, volume 4858. Springer, 2007. pp. 57–69.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. http://www.deeplearningbook.org.

[7] Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer, 1992.

[8] Michael Johanson. Measuring the size of large no-limit poker games. *CoRR*, abs/1302.7008, 2013.

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] Viliam Lisy and Michael Bowling. Equilibrium approximation quality of current no-limit poker bots, 2017.

[11] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[12] John F Nash et al. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49, 1950.

[13] J.C. Platt, N.I.P. Systems, and Neural Information Processing Systems Foundation. *Advances in Neural Information Processing Systems 20: Proceedings of the 2007 Conference*. Advances in Neural Information Processing Systems. Neural Information Processing Systems Foundation, 2009. pp. 905–912.

[14] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[15] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 1729–1736, Red Hook, NY, USA, 2007. Curran Associates Inc.