EÖTVÖS LORÁND TUDOMÁNYEGYETEM

TERMÉSZETTUDOMÁNYI KAR

OPTIMÁLIS KERESKEDÉS GÉPI TANULÁSSAL

SZAKDOLGOZAT

Kőrösi Ákos

Matematikus MSc

Témavezető: Lukács András

Számítógéptudományi Tanszék

Budapest

2024

Köszönetnyilvánítás

A dolgozat alapjául szolgáló kutatást Lukács Andrással, Nagy Lóránttal és Csanády Bálinttal végeztem, így mindenekelőtt nekik szeretnék köszönetet nyilvánítani a közös munkáért, illetve Lukács Andrásnak a témavezetőként nyújtott segítségéért is. Hasonlóan hálás vagyok a családomnak és barátaimnak, akik szintén hozzájárultak ahhoz hogy ez a dolgozat létrejöhessen.

Bevezetés

A dolgozat témája két, külön-külön is sokat tanulmányozott és fontos terület találkozásából adódik.

Az egyik terület a *gépi tanulás* (*machine learning*) témaköre. Habár a terület kutatása hosszú múltra tekint vissza, a közfigyelem főleg a 2010-es évek második felében kezdett rá koncentrálódni, amikor is különféle elméleti áttörések, és a hardveres fejlődés kombinációjának köszönhetően a gépi modellek teljesítménye valóban figyelemreméltó lett. A dolgozatban egy, a gépi tanuláson belül az ún. *megerősítéses tanulás* (*reinforcement learning*) témakörébe tartozó problémával fogunk foglalkozni. Ez olyan típusú feladatokat takar, amikor a tanulóalgoritmusnak egy adott környezetben (pl. egy számítógépes játék) kell megtanulni a helyes stratégiát (viselkedést), mégpedig nagy mértékben önállóan, csak a környezettől kapott visszajelzések alapján.

A másik témakör a pénzügyi matematika, speciálisabban annak egy központi problématípusa, mégpedig hogy amennyiben egy pénzügyi eszköz (pl. részvény) árai egy megadott modell szerint alakulnak, akkor milyen kereskedési stratégiával érhető el a lehető legtöbb profit. Ennek persze jelentős gyakorlati relevanciája is van, viszont matematikailag általában elég nehéz analitikusan megadni az optimumot, néhány alapvető modell kivételével.

A kutatásunk során az árfolyamok alakulását *autoregresszív* folyamatokkal modelleztük. Ezekre a [2] cikk analitikusan megadja az optimális stratégiát. Mi azt vizsgáljuk, hogy gépi tanulással milyen mértékben közelíthető meg az optimális teljesítmény.

A dolgozat felépítése az alábbi:

Az 1. fejezetben áttekintjük a gépi tanulás elméletének legfontosabb elemeit. Először bemutatásra kerül az a formális keretrendszer amiben általában vizsgálni szoktuk a gépi tanulást, ez a *Markov döntési folyamatok* elmélete. Ezután a neurális hálók működéséről lesz szó. Majd a főbb reinforcement learning algoritmusok ismertetése következik, bemutatva az

általunk használt algoritmus, a *Proximal Policy Optimization* (PPO) működési elvét, és a megértéséhez szükséges ismereteket. Ezután a kutatásban használt modell-architektúráról lesz szó, bemutatva annak elméleti előzményeit is.

A 2. fejezetben összefoglaljuk az autoregresszív folyamatokon való kereskedésről szóló [2] cikk fő eredményeit.

A 3. fejezetben a kutatásunk folyamatának és eredményeinek összefoglalása található.

Végül a 4. fejezet rövid összefoglalással és kitekintéssel zárja a dolgozatot.

Tartalomjegyzék

1. fejezet

Gépi tanulás

Ebben a fejezetben áttekintjük a gépi tanulás alapjait. Először lefektetjük a Reinforcement Learning (megerősítéses tanulás) elméletének formális hátterét. Reinforcement Leanring alatt olyan gépi tanulási problémákat értünk, ahol az tanulóalgoritmusunk egy bizonyos szabályokszerűségek szerint viselkedő környezetben különféle akciókat, cselekvéseket választhat, és ezek függvényében jutalmakat kap, azaz visszajelzést hogy hogyan teljesít. A cél az lesz, hogy olyan stratégiát, viselkedésmódot tanuljon meg idővel, amely maximalizálja a megszerzett jutalmat.

Először fel fogjuk építeni ennek egy formális modelljét, a *Markov döntési folyamatokat*, és belátunk néhány alapvető eredményt azok tulajdonságairól.

Ezt követően a neurális hálókról lesz szó. Jelenleg ezek alkotják a gépi tanulás fő paradigmáját, az ilyen típusú modellek képesek a legjobb teljesítményre a legtöbb problématípus esetén. Bemutatjuk a neurális hálók felépítését, működését, illetve a mögöttes elmélet főbb részleteit is.

Ezután áttekintünk néhány tanítási protokollt. Először általánosan tárgyaljuk az úgynevezett *Policy Gradiens* módszereket, majd rátérünk ennek két fontos alfajára a TRPO és PPO algoritmusokra. A kutatás során az PPO algoritmust használtuk, a TRPO pedig ennek egy elméleti előzménye, amely a fontos a PPO működésének megértéséhez.

1.1. Optimális stratégiák Markov-folyamatokban

1.1.1. Markov Döntési Folyamatok

A reinforcement learning elméleti tárgyalásánál jellemzően úgynevezett **Markov döntési folyamatként** modellezzük azt a problémát, amelyre be szeretnénk tanítani az algoritmust. Ezek formális definíciója az alábbi:

1.1. Definíció. *Markov döntési folyamatnak (MDP, Markov decision process) nevezünk egy olyan* $\big(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\big)$ *négyest, ahol* $\mathcal{S}$ *és* $\mathcal{A}$ *halmazok, és minden* $a \in \mathcal{A}, s \in \mathcal{S}$*-re* $\mathcal{P}(\cdot \mid a, s)$ *egy valószínűségi eloszlás az* $\mathcal{S}$ *halmazon, továbbá* $\mathcal{R}(a, s)$ *egy valószínűségi változó* $\mathbb{R}$ *értékkészlettel.*

A definíció interpretációja a következő. Az $\mathcal{S}$ halmaz a világ (környezet) lehetséges állapotait írja le, míg $\mathcal{A}$ a világban végrehajtható akciók halmaza. A folyamat során a játékos (ágens) tesz egy lépést, amelynek hatásaként a világ állapotot vált a $\mathcal{P}$ függvény által leírt módon: ha a jelenlegi állapot s és az ágens lépése a, akkor a $\mathcal{P}(\cdot \mid a, s)$ eloszlás szerint sorsolódik a következő állapot. Az állapotváltozáson kívül az ágens "jutalmat" is kap, amelynek mennyisége a $\mathcal{R}(a, s)$ eloszlás alapján sorsolódik. Ezek után az ágens új lépést választ, és így tovább.

Az ágens célja az, hogy a lehető több jutalmat gyűjtse be a játék során. Jellemzően azonban úgy tekintjük, hogy a távoli jövőben kapott jutalmak kisebb súllyal számítanak, mint a jelen- vagy közeljövőbeliek. Ez egyfelől a probléma matematikai kezelését is megkönnyíti (például konvergenssé téve az egyébként divergens sorként viselkedő összjutalmatÖ), illetve kutatások szerint az emberek empirikusan tapasztalható preferenciái is ehhez hasonlóan működnek. Ezt a súlyozást egy $0 < \gamma \leq 1$ *diszkontfaktor* rögzítésével szokás megoldani, mégpedig olyan módon hogy a (jelentől számítva) t időpontban kapott jutalmat γ^t súllyal számítjuk be.

Jelölje a döntési folyamat során a t. időpontban kapott jutalmat az R_t valószínűségi változó (tegyük fel hogy rögzítve van hogy "hogyan játszunk", hogyan választunk akciókat az egyes állapotokban, ezt tehát külön nem jelöljük most). Tehát a maximalizálandó mennyiség a

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]$$

diszkontált várható összjutalom lesz.

1.1. Megjegyzés. Sok, természetesen felmerülő MDP esetén vannak olyan $s \in S$ állapotok, hogy s állapotból tetszőleges $a \in \mathcal{A}$ akciót lépve 1 valószínűséggel a következő állapot is s. Ekkor s-re azt mondjuk hogy **nyelő** állapot.

Ha egy MDP-ben van nyelő állapot, akkor azt mondjuk hogy **epizodikus** az MDP. **Epizód** alatt itt azt az állapotsort értjük, ami a kezdőállapottól egy nyelő állapotba való első belépésig tart. Az elnyelődés után újraindul a Markov-folyamat. Az epizodikus MDP-ket gyakran $\gamma = 1$ feltevéssel vizsgáljuk, azaz diszkontálatlan jutalmakat tekintünk (adott epizódon belül).

1.1.2. Stratégiák és policyk

Szeretnénk tehát a játék folyamán olyan módon választani a lépéseinket az egyes állapotokban, hogy maximalizáljuk a várható diszkontált összjutalmat a játék során. A játék egy adott időpillanatában az összes addig rendelkezésre álló információ a bejárt állapotok sorozata, a választott akciók sorozata és a megkapott jutalmak sorozata (az adott időpontig bezárólag). A játékosnak ezek alapján kell döntenie. Tehát a legtágabb értelemben egy rögzített "viselkedés" a játékban nem más, mint egy olyan függvény, amely az állapotok, akciók és jutalmak sorozatához rendel egy valószínűségi eloszlást az akciók halmazán (és az, hogy a viselkedés által kijelölt módon játszunk, úgy valósítható meg, hogy minden állapotban a megfelelő eloszlásból sorsoljuk ki a lépésünket).

Egy ilyen alakban megadott viselkedés azonban túl komplex, mi ennél szűkebb stratégiahalmazra szorítkozunk. Azonban később látni fogjuk hogy ez nem probléma, hiszen az optimum azon a részhalmazon vétetik fel amire szorítkozunk.

Ez a szűkebb halmaz a **stacionárius policy**ké.

1.2. Definíció. *Determinisztikus stacionárius policy-nak nevezzük a $\pi : \mathcal{S} \to \mathcal{A}$ alakú függvényeket.*

Itt tehát minden állapot esetén rögzített, hogy oda kerülve milyen akciót kell megtenni. Ennél valamival megengedőbb az alábbi definíció:

1.3. Definíció. *Sztochasztikus stacionárius policy-nak nevezzük a $\pi : \mathcal{S} \to \mathbb{P}(\mathcal{A})$ alakú függvényeket, ahol $\mathbb{P}(\mathcal{A})$ az $\mathcal{A}$-n vett valószínűségi eloszlásokat jelöli.*

Itt tehát minden állapotnál megadunk egy valószínűségi eloszlás, ami szerint kisorsoljuk a megteendő akciót.

A továbbiakban policy alatt sztochasztikus stacionárius policyt fogunk érteni, ha nem jelezzük másként. Ezek halmazát Π_{stat} fogja jelölni.

1.1.3. Értékfüggvények

Legyen adott egy $\left(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\right)$ MDP és egy $\pi : \mathcal{S} \to \mathbb{P}\left(\mathcal{A}\right)$ sztochasztikus stacionárius policy-t.

1.4. Definíció (Értékfüggvény)**.** *Legyen adott egy MDP és egy hozzá tartozó π policy. Ezekhez tartozik egy, az állapotok halmazán értelmezett $V^\pi : \mathcal{S} \to \mathbb{R}$ értékfüggvény, melyet az alábbi módon definiálunk:*

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \cdot R_t \;\middle|\; s_0 = s \right]$$

Tehát V^π a diszkontált összjutalom értékét adja meg amennyiben az $s \in \mathcal{S}$ kezdőállapotból indulunk és végig a π policyt követjük.

Definiáljuk az **optimálisérték-függvényt** is, mint az elérhető várható diszkontált összjutalmak felső korlátját:

$$V^*(s) = \sup_{\pi \in \Pi_{stat}} V^\pi(s)$$

Ha egy policyra minden $s \in \mathcal{S}$ esetén teljesül $V^\pi(s) = V^*(s)$ akkor azt mondjuk hogy π **optimális policy**.

Ezen fejezet célja igazolni, hogy minden MDP-hez létezik optimális policy, illetve karakterizálni azokat.

Ehhez vezessük be az ún. **akció-érték függvényeket**:

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \cdot R_t \;\middle|\; s_0 = s, a_0 = a \right]$$

Ezek a függvények azt adják meg, hogy mennyi a diszkontált összjutalom várható értéke, ha s kezdeti állapotból indulva az a akciót választjuk, majd azután végig a π policynek megfelelően választunk.

Definiáljuk ezen függvények szuprémumát is:

$$Q^*(s,a) = \sup_{\pi \in \Pi_{stat}} Q^\pi(s,a)$$

Használni fogjuk, hogy a V^* és Q^* függvények kielégítik az alábbi egyenleteket (ezek triviálisan beláthatóak):

$$V^*(s) = \sup_{a \in A} Q^*(s,a)$$
$$Q^*(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s' \mid a, s) \cdot V^*(s')$$

1.1.4. Bellman-egyenletek, Bellman-operátor

Ha adott egy π policy, akkor fennállnak az alábbi **Bellman-egyenletek** $\forall s \in \mathcal{S}$:

$$V^\pi(s) = r\big(s, \pi(s)\big) + \gamma \cdot \sum_{s' \in \mathcal{S}} \mathcal{P}\big(s' \mid \pi(s), s\big) \cdot V^\pi(s')$$

Az egyenletet természetesen esetszétválasztással kapjuk a $\pi(s)$ meglépése után előálló állapot szerint, a γ szorzóval megfelelően diszkontálva a jövőbeli jutalmakat.

Ez az egyenlet motiválja a π-hez rendelt **Bellman-operátor** bevezetését. Ez egy $T^\pi : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$ operátor (itt $\mathbb{R}^{\mathcal{S}}$ az $\mathcal{S} \to \mathbb{R}$ leképezések halmazát jelöli, vagyis lényegében az $\mathcal{S}$-en értelmezett *értékfüggvényeket*), melyet az alábbi képlet definiál (amelyben ráismerhetünk a Bellman-egyenletek jobb oldalán álló kifejezésre):

$$(T^\pi V)(s) = r\big(s, \pi(s)\big) + \gamma \cdot \sum_{s' \in \mathcal{S}} \mathcal{P}\big(s' \mid \pi(s), s\big) \cdot V(s')$$

A Bellman egyenlet tehát röviden úgy írható fel, hogy $T^\pi V^\pi = V^\pi$.

Vezessük be most a $T^* : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$ alakú **Bellman optimalitási operátort**:

$$(T^* V)(s) = \sup_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} \mathcal{P}\big(s' \mid a, s\big) \cdot V(s') \right\}$$

1.1. Állítás. A V^* optimálisérték-függvény fixpontja a T^* operátornak, azaz $T^*V^* = V^*$.

Bizonyítás. A $T^*V^* = V^*$ egyenlet kibontva:

$$V^*(s) = \sup_{a \in A} \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}\big(s' \mid \pi(s), s\big) \cdot V^*(s') \right\}$$

Ez pedig ugyanolyan meggondolás alapján könnyen látható mint T^π, V^π-re vonatkozó egyenlet. □

1.1.5. Optimális stratégiák karakterizációi

Szeretnénk karakterizációkat adni egy π stratégia optimalitására. Ehhez azonban előbb alaposabban meg kell értenünk a Bellman-operátorok viselkedését.

1.2. Állítás. Tekintsünk egy tetszőleges π policyhez rendelt T^π operátort. Tegyük fel, hogy a tekintett MDP-ben a γ diszkontfaktorra teljesül $0 < \gamma < 1$. Ekkor T^π γ-kontrakció a $||\cdot||_\infty$ maximumnormára nézve, azaz tetszőleges $V, V' \in \mathbb{R}^{\mathcal{S}}$-re:

$$||T^\pi V - T^\pi V'||_\infty \leq \gamma \cdot ||V - V'||_\infty$$

Bizonyítás.

$$(T^\pi V)(s) = r\big(s, \pi(s)\big) + \gamma \cdot \sum_{s' \in \mathcal{S}} \mathcal{P}\big(s' \mid \pi(s), s\big) \cdot V(s')$$

$$(T^\pi V')(s) = r\big(s, \pi(s)\big) + \gamma \cdot \sum_{s' \in \mathcal{S}} \mathcal{P}\big(s' \mid \pi(s), s\big) \cdot V'(s')$$

$$(T^\pi V - T^\pi V')(s) = \gamma \cdot \sum_{s' \in \mathcal{S}} \mathcal{P}\big(s' \mid \pi(s), s\big) \cdot \big(V(s') - V'(s')\big)$$

Ebből háromszög-egyenlőtlenséggel:

$$\big|(T^\pi V - T^\pi V')(s)\big| \leq \gamma \cdot \sum_{s' \in \mathcal{S}} \mathcal{P}\big(s' \mid \pi(s), s\big) \cdot \big|V(s') - V'(s')\big|$$

Tudjuk, hogy $\big|V(s') - V'(s')\big| \leq ||V - V'||_\infty$, ezt az előbbi egyenlőtlenséggel kombinálva (és kihasználva hogy az együtthatóként szereplő valószínűségek összege 1) kapjuk hogy:

$$(T^\pi V - T^\pi V')(s) \leq \gamma \cdot \big|V(s') - V'(s')\big|$$

Mivel $||V - V'||_\infty$ éppen a $\big|V(s') - V'(s')\big|$ abszolút különbségek szuprémuma, azért ebből következik az is, hogy:

$$(T^\pi V - T^\pi V')(s) \leq \gamma \cdot ||V - V'||_\infty$$

Viszont $||T^\pi V - T^\pi V'||_\infty$ pedig éppen az ezen egyenlőtlenség bal oldalán szereplő kifejezések szuprémuma, ezért kapjuk:

$$||T^\pi V - T^\pi V'||_\infty \leq \gamma \cdot ||V - V'||_\infty$$

amit bizonyítani szerettünk volna. □

Hasonló állítás teljesül T^*-ra is.

1.3. Állítás. Ha $0 < \gamma < 1$ akkor tetszőleges $V, V' \in \mathbb{R}^{\mathcal{S}}$-re:

$$||T^* V - T^* V'||_\infty \leq ||V - V'||_\infty$$

Azaz T^* is γ-kontrakció a maximumnormára nézve.

Bizonyítás. Idézzük fel, hogy a T^* operátor definíciója:

$$(T^* V)(s) = \sup_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} \mathcal{P}\big(s' \mid a, s\big) \cdot V(s') \right\}$$

A szuprémumra általánosságban igaz, hogy

$$\big|\sup_{a \in \mathcal{A}} f(a) - \sup_{a \in \mathcal{A}} g(a)\big| < \sup_{a \in \mathcal{A}} \big|f(a) - g(a)\big|$$

A T^π-re vonatkozó tétel bizonyításában látottakhoz hasonlóan a háromszög-egyenlőtlenség használatával kapjuk, hogy:

$$||(T^* V)(s) - (T^* V')(s)|| \leq \gamma \cdot \sup_{(s,a) \in \mathcal{S} \times \mathcal{A}} \sup_{s' \in \mathcal{S}} \mathcal{P}(s,a,s') \cdot |V(s') - V'(s')| \tag{1.1}$$

$$\leq \gamma \cdot \sum_{s'} \mathcal{P}(s,a,s') \cdot ||V - V'||_\infty = \gamma \cdot ||V - V'||_\infty \tag{1.2}$$

□

A következő tételek kimondása előtt emlékeztetünk a Banach-tételre és néhány közvetlen következményére (a [7] könyv alapján):

1.1. Tétel. *Legyen $(X, |\cdot|)$ egy (teljes) normált tér. Tegyük fel hogy adott a téren egy $T : X \to X$ leképezés, amely kontrakció, azaz valamely $0 \leq q < 1$ konstanssal teljesül bármely $x, y \in X$-re hogy $|x - y| < q \cdot |T(x) - T(y)|$ Ekkor a T operációnak létezik $T(z) = z$ fixpontja. Ezenfelül ez a fixpont egyértelmű, továbbá tetszőleges $x \in X$ pontra az $x, T(x), T(T(x)), \ldots$ sorozat konvergál a fixponthoz.*

Vázlat. A bizonyítás úgy megy hogy tetszőleges $x \in X$ pontot választva képezzük az $x, T(x), T(T(x)), \ldots$ sorozatot. Mivel T kontrakció, azért ez a sorozat teljesíti a Cauchy-kritériumot. Ezért, és mivel teljes térben vagyunk, a sorozatnak van limesze. Könnyen belátható hogy ez a limesz fixpont, kihasználva hogy T folytonos, ami következik a kontrakció voltából.

Most tegyük fel hogy x és x' különböző fixpontok. Ekkor

$$|x - x'| = |T(x) - T(x')| < |x - x'|$$

itt az egyenlőséghez x, x' fixpont voltát használtuk ki, az egyenlőtlenséghez pedig azt, hogy kontrakció. Láthatóan ellentmondást kaptunk, tehát nem lehet több különböző fixpont. □

Korábban láttuk, hogy $T^\pi V^\pi = V^\pi$ illetve $T^* V^* = V^*$. Mivel a T^* és az összes T^π operátorok kontrakciók, kapjuk hogy amennyiben $0 < \gamma < 1$, úgy ezek (rendre V^π és V^*) az egyetlen fixpontjai a megfelelő operátoroknak. Ez azért igaz, mert ha két fixpontja lenne egy kontrakciónak, akkor ezekre alkalmazva az operációt, a képpontoknak egyfelől közelebb kéne kerülniük egymáshoz mint az eredeti pontoknak (mivel az operátor kontrakció), másfelől ugyanolyan távol kellene maradniuk (mivel mindkettő fixen marad). Ez ellentmondás, tehát legfeljebb egy fixpont lehet.

1.2. Tétel. *Annak hogy π optimális stratégia, ekvivalens karakterizációja az alábbi állítások mindegyike:*

1. *π V^*-mohó, azaz $T^\pi V^* = V^*$*
2. *$\sum_{a \in \mathcal{A}} \pi(a, s) \cdot Q^*(a, s) = V^*(s)$*

3. *π Q^*-mohó, azaz tetszőleges s állapot esetén a $\pi(\cdot, s)$ eloszlás tartóját olyan akciók képezik, amelyek maximalizálják $Q^*(a, s)$-t*

Bizonyítás. Idézzük fel, hogy a π policy optimalitásának definíciója az, hogy $V^\pi = V^*$. Ez könnyen látható módon ekvivalens az 1. állítással. Valóban, az 1. állítás azt mondja ki hogy V^* fixpontja a T^π operátornak. Azonban láttuk azt, hogy V^π is fixpontja, illetve a Banach-tételnél beláttuk hogy egy kontrakciónak pontosan egy fixpontja van. Márpedig T^π kontrakció voltát is beláttuk. Ezért kapjuk, hogy a $T^\pi V^* = V^*$ egyenlőségből adódik $V^* = V^\pi$, vagyis π optimalitása. A fordított irány pedig triviális.

Az, hogy a 2. állítás ekvivalens az optimalitással, igazolható úgy hogy bemutatjuk ekvivalenciáját az 1. állítással. Felhasználva a Q^* és V^* korábban tárgyalt összefüggéseit:

$$\sum_{a\in\mathcal{A}} \pi(a,s)\cdot Q^*(a,s) = \sum_{a\in\mathcal{A}} \pi(a,s)\big(r(a,s) + \gamma\cdot \sum_{s'\in\mathcal{S}'} \mathcal{P}(s,a,s')\cdot V^*(s')\big) =$$
$$= r(s,\pi(s) + \gamma\cdot \sum_{s'\in\mathcal{S}'} \mathcal{P}(s,\pi(s),s')\cdot V^*(s')\big) = (T^\pi V)(s)$$

Innentől triviálisan ekvivalens az állítás 1.-gyel.

Végül pedig a 3. állítás a 2.-ból olvasható le. Tudjuk hogy $V^* = \sup_{a\in\mathcal{A}} Q^*(a,s)$, vagyis $Q^*(a,s) \le V^*(s)$. Ebből $\sum_{a\in\mathcal{A}} \pi(a,s)\cdot Q^*(a,s) \le V^*(s)$ következik, és egyenlőséggel csak akkor teljesülhet, ha csak optimális Q^*-értékű akciók kapnak nemnulla súlyt. Ebből adódik az állítás. □

A tételben elmondottakból világossá válik az a fejezet elején tett állítás, hogy elegendő a stacionárius policy-k körére szorítkozni az elképzelhető viselkedés teljes halmaza helyett. Valóban, a tételből leolvasható hogy az optimalitáshoz szükséges hogy minden egyes állapotban csakis a Q^*-mohó akciók közül válasszunk - azok közül pedig a jutalom szmepontjából indifferens hogy hogyan választunk. Ezért az optimumot valóban elérhetjük egy stacionárius policyvel, jogos tehát ezek vizsgálatára szorítkoznunk.

Megemlítendő, hogy bár a fejezetben szereplő tételek és állítások formalizmusa a diszkrét esetnek felel meg (pl. szummák integrálok helyett), az eredmények néhány, meglehetősen gyenge folytonossági kritérium feltevése mellett folytonos terekben is igazak maradnak.

1.2. Neurális hálók

1.2.1. Gépi tanulás és neurális hálók

A neurális hálók olyan számítási modellek, amelyek a valóságos biológiai agyak működését igyekeznek sematikusan modellezni. Bár az alapötlet már régóta ismert, sokáig nem tudtak olyan teljesítményt produkálni, ami ami miatt az elméleti érdekességükön túlnyúlt volna a jelentőségük. Azonban a 2010-es években kitalált új kulcsötletek, illetve a hardverek fejlődése megsokszorozta e modellek erejét. Azóta több területen is közel-emberi, vagy akár azt meghaladó teljesítményt tudtak nyújtani.

Matematikailag a neurális hálók nem mások, mint affin függvények és köztes nemlinearitások egymást követő alkalmazása a háló által kapott x inputra. Adottak tehát $A_1, \ldots, A_n$ affin transzformációk, melyek (affin transzformáció lévén) az alábbi alakba írhatók

$$A_j(x) = W_j \cdot x + b_j$$

Itt W_j mátrix, b_j pedig vektor, alkalmas dimenziókkal. A W_j mátrixok elemeit *súlyok*nak szokás nevezni, a b_j vektorok (kifejező magyar fordítás hiányában) angol megnevezése pedig: *bias* vektor.

Azonban persze lineáris leképezések komponálásával csak lineáris leképezések lennének megkaphatók, ezért minden lineáris "réteg" után beiktatunk egy nemlineáris "aktivációs függvényt". Ez a gyakorlatban jellemzően a $ReLU(x) = max(x, 0)$ függvény szokott lenni, mivel empirikus tapasztalatok alapján ez nyújtja a legjobb teljesítményt.

Természetesen a W_j mátrixokról és b_j vektorokról azt is megköveteljük hogy az egymásra következő A_j leképezések, melyeket képzünk belőlük, komponálhatóak legyenek, tehát.

A teljes neurális háló tehát az

$$f(\theta, x) = ReLU \circ A_n \circ \cdots \circ ReLU \circ A_1$$

leképezésnek felel meg, ahol a θ egy olyan, sokelemű paramétervektor, amely felsorolja az összes W_j mátrix és a b_j vektor összes elemét. (A fentebbi képletben a ReLU függvényt többdimenziós vektorokra értelemszerűen komponensenként alkalmazzuk).

Praktikus feladatokra a neurális hálókat úgy használjuk, hogy a valódi inputot (kép, szöveg, hang) valamilyen módon valós vektorként reprezentáljuk, erre a vektorra alkalmazzuk a hálót, majd a háló outputját a kapott valós vektorból visszaalakítjuk a kívánt formáturma.

A megoldandó feladat tehát a hálók esetén az, hogy a θ paramétert, azaz a mátrixok elemeit úgy állítsuk be, hogy az inputokra minél megbízhatóbban a kívánt outputokat adja

Jelölje $g(x)$ az x inputra elvárt outputot. Bevezetünk egy $L(\cdot, \cdot)$ ún. *loss függvényt*, amely azt méri hogy két vektor "mennyire különböző", ez a probléma típusától függvően többféle is lehet.

Klasszifikációs feladatoknál, tehát amikor az inputot adott osztályokba kell sorolni (például kutyát vagy macskát ábrázoló képekről megmondani hogy melyiket ábrázolják), gyakori választás a *cross-entropy loss függvény*. Ennek a képlete C-dimenziós $y, \hat{y}$ vektorokra az alábbi:

$$H(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

A függvény használatakor $\hat{y}$ lesz a háló outputja, y pedig az elvárt eredmény. Klasszifikációs feladatoknál a háló outputját jellemzően a

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}} \quad (j = 1, \ldots, C)$$

képletű *softmax* függvénnyel normáljuk. Ezzel azt érjük el, hogy az output komponenseinek összege 1 lesz, továbbá nemnegatívak is lesznek a komponensek, tehát egy valószínűségi eloszlást adnak meg. Az eloszlásból pedig már könnyebben választhatunk egy kívánt elemet (klasszifikáció esetén természetes választás a legnagyobb valószínűségű elem, más feladatoknál azonban például a valószínűségi eloszlásból való mintavételezés a bevett, a feladathoz adekvát módon választva).

Ha nem klasszifikációs feladatról van szó, hanem egy "folytonosabb" értékkészletű az output, akkor gyakori választás a *Mean Square Error* (átlagos négyzetes hiba):

$$MSE(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

illetve a Mean Absolute Error:

$$MAE(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Összefoglalva, szeretnénk elérni hogy a modellünk a lehető legkevesebbet tévedjen, azaz formálisan megfogalmazva feladatunk a θ paraméter olyan megválasztása lesz, amely minimalizálja a

$$\mathbb{E}\left[L(Y, f(X;\theta))\right] = \int\int L\left(y, f(x;\theta)\right) \cdot p(x,y)\, dx\, dy$$

mennyiséget (az integrál persze praktikus esetekben jellemzően egy szummaként értendő).

1.2.2. Gradient descent

Tehát egy optimalizációs probléma áll előttünk: úgy beállítani a θ paramétert, hogy a $\mathbb{E}\left[L(Y, f(X;\theta))\right]$ minimális legyen. Ez a valós alkalmazások esetén általában analitikusan nem megoldható.

Az egyik általánosan használat technika ilyenkor a *gradient descent* (gradiens leereszkedés) algoritmus. Ennek során az optimumot nagyszámú iteratív lépés során közelítjük meg, amelyek során csak kisebb, fokozatos javulásokat akarunk elérni a célfüggvény értékében. Specifikusan a gradient descent módszer esetében a célfüggvény aktuális pontbeli gradiensének az ellentétes irányába fogunk elmozdulni. Informálisan ezt az ötletet az tény motiválja, hogy ebben az irányban csökken legmeredekebben a célfüggvény értéke (lokálisan).

Ha tehát a h függvényt akarjuk minimalizálni, akkor a gradient descent algoritmus során a

$$x_{n+1} = x_n - \eta \cdot \nabla_x h(x_n)$$

rekurziót követjük (valamilyen módon inicializált kezdőpontból). Az η paraméter neve *learning rate* (tanulási ráta); ez a gyakorlatban általában nem konstans a teljes folyamat alatt, hanem jellemzően valamilyen dinamika szerint fokozatosan csökkentjük.

Elméleti megalapozottságát tekintve több konvergenciaeredmény is létezik a gradient descentről. A legalapvetőbb és legismertebb ilyen tétel (lásd pl. a [1] könyv 9.3 fejezete) azt mondja ki hogy konvex h függvény esetén az algoritmus konvergál a minimumponthoz. Azonban persze a konvexitás általában (a neurális hálók esetében pedig főképpen) túl erős feltétel. Az alaptételnek léteznek erősebb verziói, azonban teljes általánosságban nincs garancia a gradient descent algoritmus konvergenciájára. Mindennek ellenére azonban az algoritmus empirikus eredményei sok esetben kielégítőek, különösen ha megfelelő dinamika szerint alakítjuk learning rate-t és egyéb hiperparamétereket.

1.2.3. Backpropagation

Ha neurális hálókat tanítunk a gradient descent algoritmussal akkor ennek során lépésenként ki kell számolnunk a $\nabla_\theta \mathbb{E}\big[L(Y, f(X;\theta))\big]$. Persze az alkalmazások esetén véges számú adatponttal dolgozunk, vagyis

$$\nabla_\theta \ \mathbb{E}\Big[L(Y, f(X;\theta))\Big] = \nabla_\theta \left[\frac{1}{n}\sum_{i=1}^{n} L\big(y_i, f(\theta; x_i)\big)\right] = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \ L\left(y_i, f(\theta; x_i)\right)$$

alakú a képlet a várható érték kibontásával, ahol $(x_i, y_i) \quad (i = 1, \dots n)$ olyan párok amelyek egy input értékből és az arra elvárt output értékből állnak. Célunk az, hogy a háló outputjai minél kisebb várható hibával közelítsék meg az elvárt outputokat.

1.2. Megjegyzés. Gyakorlati alkalmazások esetén nem veszünk minden iterációs lépésben minden adatpontot figyelembe, ahogy azt ebben a képletben tesszük. Ehelyett véletlenszerű mintavételezéssel kiválasztjuk az adatpontok egy kisebb részhalmazát, egy ún. *minibatch*-et, és csak azokra képezzük a fenti szummát, majd annak gradiensét.

Azonban még minibatchek vételezésével is a teljes adathalmaz helyett sem oldható meg hatékonyan, hogy iterációs lépésenként több alkalommal is $\nabla_\theta \ L\big(y_i, f(\theta; x_i)\big)$ alakú gradienseket számoljunk ki a klasszikus (akár szimbolikus, akár numerikus) módszerekkel, hiszen a gyakorlati alkalmazásokban az f egy sokszorosan összetett függvény, amely továbbá függ a nagyon sok komponensű θ paramétervektortól is.

Ennek a számítási problémának a megoldására született a backpropagation algoritmus. Az algoritmus alapötlete, hogy kihasználjuk f összetett függvény struktúráját és a lánc-

szabályt, hogy a függvény különböző "rétegeiben" lévő paramétereit visszafelé haladva egymás után rekurzíven változtassuk, így valósítva meg a kívánt gradiensirányú lépést.

Emlékeztetésképp, a hálót leíró paraméteres függvény

$$f(\theta; x) = ReLU \circ A_n \circ \cdots \circ ReLU \circ A_1$$

alakú, ahol az A_i-k lineáris leképezések, ezek mátrixainak komponenseit gyűjti össze a θ paraméter.

1.3. Optimizációs algoritmusok a gépi tanulásban

1.3.1. Alapvető algoritmusok

Policy Gradient algoritmusok

A Policy Gradient algoritmusok lényegében a neurális hálókban használt gradient descent algoritmusok analogonjai a megerősítéses tanulás kontextusában. Az alapötlet itt is az, hogy a célfüggvény gradiense szerint változtatjuk a modellünk paramétereit, és ezt a lépést iteráljuk amíg kellő módon megközelítjük az optimális teljesítményt a modell részéről.

Ennek implementálásához azonban szükség lenne arra, hogy a célfüggvény gradiensét jól tudjuk közelíteni. Ehhez ad elméleti hátteret az alábbi tétel:

1.3. Tétel (Policy Gradient Theorem)**.** *Legyen adott egy Markov döntési folyamat és azon értelmezett paraméteres policyk egy családja, amelynek a paraméterezett átmenetvalószínűségfüggvénye* $\pi(a|s;\theta)$.

Vezessük be a

$$J(\theta) = \sum_{s\in\mathcal{S}} d^{\pi}(s)V^{\pi}(s) = \sum_{s\in\mathcal{S}} d^{\pi}(s)\sum_{a\in\mathcal{A}} \pi_{\theta}(a|s)Q^{\pi}(s,a)$$

függvényt. Itt π *a megfelelő stacionárius eloszlást jelöli, erről lásd a Tétel utáni megjegyzést.*

Ekkor:

$$\nabla_{\theta} J(\theta) = \sum_{s\in\mathcal{S}} d^{\pi}(s)\sum_{a\in\mathcal{A}} Q^{\pi}(s,a)\nabla_{\theta}\pi_{\theta}(a|s)$$

1.3. Megjegyzés (A stacionárius eloszlás)**.** Ha rögzítjük a π_θ policyt a Markov döntési folyamatra nézve, akkor egy Markov-láncot kapunk az állapotokon. Ennek a Markov-láncnak van egy *stacionárius eloszlása*, amely egy olyan eloszlást jelent, ami invariáns arra hogy teszünk egy lépést a láncon. Ezt jelöli a képletben π. A stacionárius eloszlásoknak számos szép tulajdonsága van, nekünk főként az lesz figyelemreméltó hogy (elfajuló eseteket leszámítva) ez az eloszlás létezik, egyértelmű, és értékei éppen a látogatási valószínűségeknek felelnek meg.

vázlat. Először írjuk fel a V^π gradiensét tetszőleges s állapotot véve:

$$\begin{aligned}
\nabla_\theta V^\pi(s) &= \nabla_\theta \Big(\sum_{a\in\mathcal{A}} \pi_\theta(a|s) Q^\pi(s,a) \Big) \\
&= \sum_{a\in\mathcal{A}} \Big(\nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) + \pi_\theta(a|s) \nabla_\theta Q^\pi(s,a) \Big) \\
&= \sum_{a\in\mathcal{A}} \Big(\nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) + \pi_\theta(a|s) \nabla_\theta \sum_{s'} \mathcal{P}(s'|s,a) \Big(r(s,a) + V^\pi(s') \Big) \Big) \\
&= \sum_{a\in\mathcal{A}} \Big(\nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) + \pi_\theta(a|s) \sum_{s'} P(s'|s,a) \nabla_\theta \Big(r(s,a) + V^\pi(s') \Big) \Big) \\
&= \sum_{a\in\mathcal{A}} \Big(\nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) + \pi_\theta(a|s) \sum_{s'} P(s'|s,a) \nabla_\theta V^\pi(s') \Big)
\end{aligned}$$

Az első egyenlőség nyilvánvaló, a második a szorzat deriválási szabályából adódik, a harmadik Q^π felbontása a következő állapot szerint, a negyedik és ötödikszintén triviális (felhasználva, hogy a $\mathcal{P}$ valószínűségek és r jutalmak nem függnek θ-tól).

Összefoglalva, azt vezettük le, hogy:

$$\nabla_\theta V^\pi(s) = \sum_{a\in A} \Big(\nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) + \pi_\theta(a|s) \sum_{s'} P(s'|s,a) \nabla_\theta V^\pi(s') \Big)$$

Vegyük észre, hogy ugyanez a felbontás alkalmazható a szummabeli $V^\pi(s')$ értékekre is. Ezt fogjuk rekurzívan megtenni, azonban előtte vezessünk be néhány jelölést a követhetőség érdekében. Legyen

$$\varphi(s) = \sum_{a\in A} \nabla_\theta \ \pi_\theta(a|s) Q^\pi(s,a)$$

Ez a fenti felbontás "első fele". Jelölje $\rho_\pi(s_1 \to s_2, k)$ annak a valószínűségét hogy π szerint haladva s_1-ből pontosan k lépés után s_2-be érünk. Persze $\rho_\pi(s_1 \to s_2, 1) = \mathcal{P}(s_2 \mid s_1)$. Az is könnyen látható, hogy $\rho^\pi(s \to x, k+1) = \sum_{s'} \rho^\pi(s \to s', k) \rho^\pi(s' \to x, 1)$ (a k. állapot szerinti felbontással).

$$
\begin{aligned}
\nabla_\theta V^\pi(s) &= \phi(s) + \sum_a \pi_\theta(a|s) \sum_{s'} P(s'|s,a) \nabla_\theta V^\pi(s') \\
&= \varphi(s) + \sum_a \sum_{s'} \pi_\theta(a|s) P(s'|s,a) \nabla_\theta V^\pi(s') \\
&= \varphi(s) + \sum_{s'} \rho^\pi(s \to s', 1) \nabla_\theta V^\pi(s') \\
&= \varphi(s) + \sum_{s'} \rho^\pi(s \to s', 1) \left[\varphi(s') + \sum_{s''} \rho^\pi(s' \to s'', 1) \nabla_\theta V^\pi(s'') \right] \\
&= \varphi(s) + \sum_{s'} \rho^\pi(s \to s', 1) \varphi(s') + \sum_{s'} \rho^\pi(s \to s', 1) \sum_{s''} \rho^\pi(s' \to s'', 1) \nabla_\theta V^\pi(s'') \\
&= \varphi(s) + \sum_{s'} \rho^\pi(s \to s', 1) \varphi(s') + \sum_{s''} \rho^\pi(s \to s'', 2) \nabla_\theta V^\pi(s'')
\end{aligned}
$$

A végtelenségig hasonlóan folytatva a kibontást, kapjuk hogy:

$$
V^\pi(s) = \sum_{s^* \in S} \sum_{k=0}^{\infty} \rho^\pi(s \to s^*, k) \varphi(s^*)
$$

Vegyük észre hogy a $\sum_{k=1}^{\infty} \rho^\pi(s \to s^*) = d_\pi(s^*)$.

Tehát megkaptuk, hogy

$$
\nabla_\theta V^\pi(s_0) = \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \varphi(s) = \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \nabla_\theta \ \pi_\theta(a|s) Q^\pi(s,a)
$$

Ez egy s_0-tól független formula, ezért $\nabla_\theta J(\theta)$ egyenlő lesz ezzel, hiszen azt úgy kaphatjuk meg, hogy s_0 mintavételezése után kiszámoljuk ezt a (konstans) értéket.

□

1.4. Megjegyzés. A tétel gyakran az alábbi formában fordul elő:

$$
\nabla_\theta J(\theta) = \mathbb{E}_\pi \big[Q^\pi(a|s) \nabla_\theta \ln \pi_\theta(a|s) \big]
$$

Ez a tétel általunk bizonyított alakjából a $\nabla_\theta \ \ln \pi_\theta(s,a) = \pi_\theta(a|s) \cdot \nabla_\theta \ \pi_\theta(a|s)$ összefüggés kihasználásával következik, ami pedig a logaritmus deriválási szabályából adódik.

A Policy Gradient algoritmusok bővebb tárgyalásért ld. pl. a [6] könyv 13. fejezetét.

1.3.2. Trust Region Policy Optimization

A Trust Region Policy Optimization (TRPO, [4]) egy 2015-ben publikált, a Policy Gradient módszerek családjába tartozó optimizációs algoritmus, amely publikálása idején az egyik legjobbnak számított. Azóta felülmúlta a rá épülő, ezt továbbfejlesztő Proximal Policy Optimization (PPO), azonban annak megértéséhez érdemes előbb a TRPO-val is megismerkedni.

Az alábbiakban áttekintjük a TRPO háttérelméletét, majd bemutatjuk magát az algoritmust is.

Legyen adott egy $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ Markov döntési folyamat és legyen a diszkontfaktor γ. Jelöljük egy π policy által szerzett várható (diszkontált) jutalmat továbbra is $\eta(\pi)$ jelöléssel. A szokásos Q_π és V_π függvények mellett vezessük be az

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$$

előnyfüggvényt, ez láthatóan azt adja meg, hogy mennyivel járunk jobban a játék folyamán, ha a jelen s állapotban először a-t lépünk, majd végig π szerint játszunk, ahhoz képest mintha azonnal π szerint kezdtünk volna játszani.

A következő egyenlet régóta ismeretes az ezzel foglalkozó szakirodalomban. Tegyük fel, hogy adott két policy π és π'. Ekkor:

$$\eta(\pi') = \eta(\pi) + \mathbb{E}_{s_0,a_0,\ldots\sim\pi'}\left[\sum_{t=0}^{\infty}\gamma^t A_\pi(s_t,a_t)\right]$$

A várható érték indexelése azt jelöli, hogy az akciókat a π' policynek megfelelően választjuk egy adott állapotban, és eszerint alakul az s_t állapotok sorozata is természetesen. Az egyenlet tehát azt fejezi ki, hogy a két policy közti összjutalom-különbség felbontható a lépésenkénti előnyértékek összegére. Bár ez az állítás meglehetősen triviálisnak tűnik, a bizonyítása ennél technikásabb, lásd a [3] cikket.

Az egyenletet tovább alakítva kapjuk az alábbiakat:

$$\eta(\pi') = \eta(\pi) + \sum_{t=0}^{\infty} \sum_{s} \mathbb{P}(s_t = s \mid \pi') \sum_{a} \gamma^t \pi'(a \mid s) A_\pi(s,a) \tag{1.3}$$

$$= \eta(\pi) + \sum_{t=0}^{\infty} \gamma^t P(s_t = s \mid \pi') \cdot \sum_{a} \pi'(a \mid s) A_\pi(s,a) \tag{1.4}$$

$$= \eta(\pi) + \sum_{s} \rho_{\pi'}(s) \sum_{a} \pi'(a \mid s) A_\pi(s,a). \tag{1.5}$$

ahol $\rho_{\pi'}(s)$ azt jelöli hogy milyen gyakorisággal látogatja meg a folyamat az s állapotot, feltéve hogy a π' policy szerint játszunk, diszkontálva a γ faktorral, ez a $\sum_{t=0}^{\infty} \gamma^t P(s_t = s \mid \pi')$ mennyiségnek felel meg tehát.

Ebből az egyenletből olvasható le egy központi meglátás (ami még nem a cikk saját ötlete, hanem már korábban is lefektetett eredmény), miszerint ha egy $\pi \mapsto \pi'$ iterációs lépésnek minden s állapotra nézve nemnegatív várható előnye van, azaz $\sum_s \rho_{\pi'}(s) \sum_a \pi'(a \mid s) A_\pi(s,a) \geq 0$ akkor teljesül $\eta(\pi') \geq \eta(pi)$ is, sőt, az utóbbi egyenlőtlenség csak akkor teljesül egyenlőséggel, ha minden s állapotban 0 a π' várható előnye.

Ez az eredmény motiválja az ún. *egzakt policy iteráció* algoritmust, amelyet a

$$\pi_{t+1}(s) = \arg\max_a A_\pi(s,a)$$

iterációs lépés határoz meg. A belátottak szerint ameddig létezik olyan (s,a) pár, amire $A_{\pi_t}(s,a) > 0$ és $\rho_{\pi_t}(s) > 0$, addig az iteráció javítja a policyt. Ha tehát nem találunk javítást akkor, feltéve hogy a policy minden állapotot pozitív valószínűséggel látogat meg, garanciánk van arra hogy az iteráció konvergált az optimális policyhez.

Az egzakt policy iteráció a gyakorlatban azonban túl számításigényes. Emiatt közelítő algoritmusokkal dolgozunk. Először bevezetünk egy approximációt η helyett:

$$L_\pi(\pi') = \eta(\pi) + \sum_{s} \rho_\pi(s) \sum_{a} \pi'(a \mid s) A_\pi(s,a)$$

Mint látható ez alakját tekintve majdnem megegyezik az $\eta(\pi')$-re levezetett algoritmussal. A különbség annyi, hogy itt a $\rho_\pi(s)$ látogatási gyakoriságokat tekintjük. Ez gyakorlati szempontból azért jelentős könnyebbség, mert egy iterációs lépésnél a ρ_π értékek becslése az aktuális policyre nézve kézenfekvő feladat, míg az hogy a $\rho_{\pi'}$ értékeket számoljuk az összes szóba jövő π'-re rendkívüli erőforráspazarlás lenne.

Belátható egyébként, hogy paraméteres L_π és η elsőrendben megegyeznek, azaz egyrészt $L_{\theta_0}(\pi_\theta) = \eta(\pi_{\theta_0})$ (ez triviális), másfelől viszont a $\nabla_\theta L_{\theta_0}(\pi_\theta)$ és $\nabla_\theta \eta(\pi_\theta)$ deriváltak is egyenlők a $\theta = \theta_0$ pontban. Ez informálisan azt jelenti, hogy θ_0 közelében L_{θ_0} jól közelíti η-t.

A TRPO algoritmushoz szükségünk lesz egy távolságfogalomra a policy-k halmazán. Először vezessük be a teljes variációs távolság fogalmát. Ha adottak a p, q eloszlások akkor $D_{TV}(p||q) = \frac{1}{2}\sum_i \frac{\downarrow}{p_i} - q_i|$ (itt diszkrét valószínűségi mezőre írtuk fel, de természetesen ugyanígy átültethető folytonosra is, integrálással a szumma helyett).

Ezután definiáljuk a π, π' policyk távolságát az alábbi módon:

$$D_{TV}^{max}(\pi, \pi') = \max_{s \in \mathcal{S}} D_{TV}\big(\pi(\cdot, s), \pi'(\cdot, s)\big)$$

Belátható az alábbi alsó becslés:

1.4. Tétel. *Legyen* $\alpha = D_T^{max} V(\pi, \pi')$. *Ekkor:*

$$\eta(\pi_{new}) \geq L_{\pi_{old}}(\pi_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}\alpha^2$$

ahol $\epsilon = \max_{s,a} |A_\pi(s, a)|$

Az is ismert, hogy D_{TV}^{max} metrika felülről becsülhető a sokkal gyakrabban használt D_{KL} *Kullback-Leibler divergencia* segítségével. Ez is egy eloszlásokon használt távolságfogalom, melyet az alábbi képlet ad meg:

$$D_{KL}(p \parallel q) = \sum_i p_i \log \frac{p_i}{q_i}$$

illetve az ezzel analóg integrálképlet a folytonos esetben. Ismert, hogy $D_{TV}(p, q)^2 \leq D_{KL}(p, q)$ áll fenn bármely p, q eloszlásokra.

Ezt kombinálva az előző egyenlőtlenséggel, kapjuk, hogy

$$\eta(\pi') \geq L_\pi(\pi') - C \cdot D_{KL}^{max}(\pi, \pi')$$

ahol $D_{KL}^{max}(\pi, \pi') = \max_{s \in \mathcal{S}} D_{KL}\big(\pi(\cdot, s), \pi'(\cdot, s)\big)$ és

$$C = \frac{4\epsilon\gamma}{(1-\gamma)^2}\alpha^2$$

Ezen előkészületek után, a TRPO által javasolt policy iteráció az alábbi:

$$\pi_{i+1} = \arg\max_{\pi} \left[L_{\pi_i}(\pi) - C D_{KL}^{\max}(\pi_i; \pi) \right]$$

Belátjuk hogy ez az iteráció monoton javítja az η célfüggvényt. Vezessük be az

$$M_i(\pi) = L_{\pi_i}(\pi) - C \cdot D_{KL}^{max}(\pi_i, \pi)$$

jelölést, ez az η-ra fentebb adott alsó becslés a π_i policy-t alapul véve. Tudjuk tehát, hogy $\eta(\pi_{i+1}) > M_i(\pi_{i+1})$ (hiszen alsó becslés). Azt is tudjuk, hogy $\eta(\pi_i) > M_i(\pi_i)$. Ez azért igaz, mert $\eta(\pi_i) > L_{\pi_i}(\pi_i)$ és könnyen meggondolható, hogy $D_{KL}^{max}(\pi_i, \pi_i) = 0$.

Ezen két egyenlőtlenséget kombinálva:

$$\eta(\pi_{i+1}) - \eta(\pi_i) \geq M_i(\pi_{i+1}) - M_i(\pi_i)$$

Ebből viszont az következik, hogy amennyiben π_{i+1}-et úgy választjuk meg, hogy maximalizálja M_i-t akkor ezzel garantáljuk hogy η nem csökken.

Ezzel kaptunk egy olyan algoritmust, amely monoton növelni képes a célfüggvényt. Valójában viszont TRPO alatt nem pontosan a fenti képletű algoritmust értjük, hanem a

$$\begin{aligned} &\max && L_{\theta_{\text{old}}}(\theta) \\ &\text{amelyre} && D_{KL}^{\max}(\theta_{\text{old}}, \theta) \leq \delta. \end{aligned} \tag{1.6}$$

korlátozott optimalizálási problémát. Ezt főleg az a gyakorlati meggondolás motiválja, hogy az eredeti képlettel túl lassan konvergál az iteráció.

A valóságban persze nem pontosan a képletek szerint folytatjuk le az algoritmust, hanem becslésekkel, közelítésekkel dolgozunk. Néhány ilyet áttekintünk az alábbiakban. Először is, jellemzően a D_{KL}^{max} helyett D_{KL} átlagát szokás tekinteni. Erre is igaz, ami a legtöbb várható értéket vagy átlagot tartalmazó képletre is, hogy a gyakorlati megvalósításokban ezt jellemzően egy kisebb mintán (*minibatch*) vett átlaggal közelítjük. A legtöbb

más érték közelítésére, amelyek a képletekben szerepelnek, pedig **Monte-Carlo módszereket** használunk. Ez a megnevezés módszerek egy tág családját takarja, amelyeknek az az alapötlete, hogy (jellemzően meglehetősen nagy számú) szimuláció elvégzése után a szimulált eredmények eloszlásából igyekszünk megérteni a vizsgált jelenséget. Itt ez a megközelítés úgy valósítható meg, hogy egy $\pi_t \mapsto \pi_{t+1}$ policy update után megadott lépésszámig futtatjuk az aktuális policyt, ezzel egy $s_0, a_0, s_1, a_1, \ldots$ sorozatot generálva, majd ebből a sorozatból számoljuk ki a minket érdeklő értékeket (tartózkodási valószínűségek, $A_\pi(s,a)$ előnyértékek, stb.). Ha a környezet lehetőséget ad erre, akkor gyakran egyszerre több futási trajektóriából is mintavételezünk egy Monta-Carlo szimuláció keretében.

1.3.3. Proximal Policy Optimization

A Proximal Policy Optimization (PPO, [5]), a TRPO algoritmus egy olyan módosítása, amely a gyakorlati tapasztalatok szerint lényegesen jobb eredményeket produkál.

Mint láttuk, a TRPO algoritmus alapgondolata az volt hogy ha az iterációs lépés során úgy keresünk új policy-paramétereket, hogy "korlátozzuk" a régi és új algoritmus "távolságát" (vagy ténylegesen egy korlátozott optimalizációs feladatot megoldva, vagy pedig bizonyos együtthatóval büntetve ezt a távolságot), elméleti garanciáink vannak arra, hogy monoton javulni fog a célfüggvény. A PPO algoritmus esetén is hasonlóra törekszünk, azzal a különbséggel hogy itt a jutalomfüggvényt fogjuk "levágni". Ennek a hatása hasonló lesz a TRPO-ban látott korlátozásokhoz. Intuitíve ugyanis a levágott (konstanssá tett) részen a függvény gradiense eltűnik, ami mintegy "elveszi az ösztönzést" arra hogy az iterációs lépés során messzire updateljünk a jelen policy kellően közeli környezetén kívülre.

Először vezessük be az alábbi jelölést. Tegyük fel hogy adott egy θ_0 policy-paraméter, ezen szeretnénk egy iterációs lépés során javítani. Legyen tetszőleges θ paraméterre legyen

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_0}(a_t|s_t)}$$

a *valószínűségi arány* (*probability ratio*).

A TRPO az alábbi célfüggvényt igyekszik optimalizálni egy $\theta \to \theta'$ update során:

$$L(\theta) = \mathbb{E}_t\left[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} A_t\right] = \mathbb{E}_t\left[r_t(\theta')A_t\right]$$

Ez valóban ekvivalens a TRPO-nál tekintett célfüggvénnyel, ezt könnyedén beláthatjuk, először kicserélve A_t-re a Q_t-t (ez csak egy konstansnyi különbséget eredményez) bővítve azt a formulát egy $\frac{\pi(a|s)}{\pi(a|s}$ tényezővel.

A PPO ehelyett az alábbi célfüggvényt javasolja:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) A_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) A_t \right) \right]$$

Az ϵ itt egy adott hiperparaméter, míg $clip(x, a, b) = min\big(b, max(a, x)\big)$ ha $a \leq x \leq b$.

1.4. Alapvető architektúrák és modellek

Ebben a fejezetben bemutatunk néhány, a reinforcement learning területén alapvető modelltípust. Először a *Q-learning*ről lesz szó, ami az egyik legalapvetőbb modell a reinforcement learning területén, mind korai felfedezése, mind egyszerűsége folytán. Ezután az *Actor-Critic* architektúrákról lesz szó, ez volt a kutatásunk során is használva.

1.4.1. Q-learning

A Q-learning algoritmus alapötlete, hogy a tanulás során egy úgynevezett *Q-táblázatot* tárolunk, folyamatosan frissítve azt. A Q-táblázat lényegében egy olyan függvényt amely $(s, a) : s \in \mathcal{S}, a \in \mathcal{A}$ egy párokhoz rendel egy $Q(s, a)$ valós értéket. Ennek a számnak tulajdonképpen az a célja, hogy megbecsülje a $Q^*(s, a)$ értéket, tehát a várható diszkontált összhasznot, feltéve hogy s állapotból indulva a akciót teszünk, majd utána optimálisan választunk (persze ezt az optimális stratégiát nem kell ismernünk, mindössze arról van szó hogy egy ilyen becslésként interpretáljuk az értéket).

Az algoritmus során először inicializáljuk a Q-táblát, például kitöltjük 0 értékekkel, vagy valamilyen a priori becsléssel, ha rendelkezésre áll ilyen.

Ezután a tanulási folyamat a következőképpen zajlik. Tételezzük fel hogy az aktuális időpontban éppen az $s \in \mathcal{S}$ állapotban vagyunk. Ki kell választanunk, hogy milyen akciót lépünk. A Q-learning legegyszerűbb, sztenderd változatában ez a választás úgy történik, hogy tekintjük azon $a \in \mathcal{A}$ akciókat, amelyek maximalizálják a $Q(s, a)$ Q-értéket az $\mathcal{A}$ halmazon, és ezek közül választjuk ki az egyiket. A gyakorlatban azonban sokszor

valamivel komplikáltabban választunk, jellemezően valamiféle felfedezési dinamikát is építve az algoritmusba. Ez történhet például olyan módon, hogy az algoritmus a lépés választása előtt sorsol, és megadott kicsi (és a futás során jellemzően egyre csökkenő) valószínűséggel teljesen véletlenszerűen választ lépést, míg a többi esetben az előbb leírt Q-értékmaximalizáló szabály szerint választ.

Miután az algoritmus meglépte a lépést, információt szerez: egyfelől a szerzett jutalom mennyiségét (r), másfelől a következő állapotot (s'). Ezek alapján fogjuk frissíteni a Q-táblát.

A sztenderd frissítési szabály az, hogy ilyenkor az (s, a) párhoz rendelt Q-értéket frissítjük, mégpedig a következő értékre:

$$Q'(s,a) = (1-\alpha) \cdot Q(s,a) + \alpha \cdot (r + \gamma \cdot \max_a Q(s',a))$$

Itt az $r + \gamma \cdot \max_a Q(s', a)$ a frissített becslés az elérhető maximális diszkontált összjutalomra (az azonnali haszon r, a γ együtthatóval szorzott szumma pedig a jövőbeli összjutalom diszkontálva). Azonban az algoritmus nem erre az újraszámolt értékre frissíti a Q-értéket, hanem egy α és $1 - \alpha$ együtthatókkal vett konvex kombinációra; ezzel kisimítva, konzervatívabbá téve a tanulófolyamatot.

A Q-learning arra épít, hogy ezen update szabály alkalmazásával a Q-értékek konvergálni fognak a tényleges optimális diszkontált összjutalomhoz. A Q-learning algoritmust bevezető [8] cikk igazolta ezt bizonyos megszorításokkal: feltéve hogy véges az akció- és állapottér, hogy minden (s, a) pár végtelenszer meg van látogatva, hogy az α learning rate nullához tart. Azóta több általánosítását is igazolták ennek a tételnek.

A Q-learningnek számos variánsa és továbbfejlesztése létezik. Kiemelendő például a *mély Q-tanulás*, ahol egy neurális hálóra bízzuk azt a feladatot, hogy Q-értékeket becsüljön. Részben ezzel lesz analóg a később tárgyalandó Actor-Critic architektúracsalád is.

1.4.2. Actor-Critic modellek

Az *Actor-Critic modellek* lényege, hogy két különálló modellt alkalmazunk egy meghatározott kombinációban, és ezek együttesétől várjuk hogy megoldják a feladatot. Konkrétabban, az egyik modulnak (*Critic*) az lesz a feladata hogy megbecsülje egy adott akció vagy állapot "értékét", és az *Actor* modulnaklesz az a feladata, hogy (a *Critic*től kapott információk

alapján) kiválassza az akciót, amelyet megtesz a modell. Az Actor és a Critic modult is a szokásos Reinforcement Learning technikával tanítjuk, hogy minél jobban lássák el a saját részfeladatukat, és azt reméljük hogy ennek eredményeképpen a kettőjük által alkotott összetett modell teljesítménye is kielégítő lesz.

Az Actor szubmodul tanítása az egyszerűbb. Ez a policy gradiens módszereknél megszokott módon zajlik, azzal a sajátossággal, hogy a célfüggvényt a Critic ágens szolgáltatja. Tehát a policy gradiens módszernél szokásos

$$\mathbb{E}_\pi\big[A^\pi(a|s)\nabla_\theta \ln \pi_\theta(a|s)\big]$$

célfüggvényt tekinthetjük, annyi különbséggel hogy itt A^π szerepére a Critic szubmodul becslését használjuk.

A Critic modul tanítása ennél valamivel összetettebb, ehhez sztenderd választás például a **Temporal Differences** (TD, ld. [6] könyv, 6. fej.) algoritmuscsalád valamely tagja, például a legegyszerűbb TD(0) tanítóalgoritmus.

A TD(0) algoritmus a következő frissítési szabályon alapul. Tegyük fel, hogy egy, az állapotokon definiált $V(s)$ értékfüggvényt akarunk optimalizálni. Ezt úgy tesszük, hogy egy rögzített policy szerint haladunk a Markov-láncon és minden lépés után frissítjük az aktuális V-t. Tegyük fel, hogy a t. lépés keretében az s_t állapotból az s_{t+1} állapotba léptünk, és R_t jutalmat kapunk. Ekkor legyen:

$$V_{t+1}(s_t) = V_t(s_t) + \alpha \cdot \big(R_t + \gamma V_t(s_{t+1}) - V_t(s_t)\big),$$

Vázlatosan mindössze ennyiből áll a TD(0) algoritmusa. A Temporal Differences algoritmuscsaládban annak ennél sokkal összetettebb és kifinomultabb algoritmusok is, melyek távolabbra tekintenek vissza a trajektória múltjára, hogy pontosabb módon frissítsék V-t, ám a gyakorlatban gyakran a TD(0) is megfelelően teljesít.

2. fejezet

Optimális kereskedés autoregresszív árfolyamatokon

Ebben a fejezetben összefoglaljuk Deák Sándor és Rásonyi Miklós "An explicit solution for optimal investment problems with autoregressive prices and exponential utility" című cikkét ([2]), amely felveti (és megoldja) azt a problémát, amelynek gépi tanulással történő megoldhatóságát a kutatásunk során vizsgáltuk. A cikk azt vizsgálja, hogy milyen kereskedési stratégiát érdemes követni egy olyan piacon, ahol az árak egy úgynevezett autoregresszív folyamat szerint alakulnak. Habár a standard közgazdaságtani modellekben az árakat gyakran martingáljellegű folyamatokkal (pl. Brown-mozgás) szokás modellezni, számos kutatási eredmény van arra vonatkozóan, hogy a piaci árak tendenciát mutatnak arra, hogy hosszú távon "visszatérjenek az átlaghoz" a hozamaikat nézve. Az autoregresszív folyamatok pedig egy relatíve könnyen kezelhető modelljét adják az ilyen tendenciákat mutató folyamatoknak.

2.1. Problémafelvetés

Tekintsünk egy pénzügyi piacot, amelyen kereskedhetünk egy olyan eszközzel, amelynek ára egy autoregresszív idősor szerint alakul, tehát a

$$X_{t+1} = \alpha X_t + \sigma \cdot \epsilon_{t+1}$$

rekurzió határozza meg, ahol $\alpha \in \mathbb{R}$ és $\sigma > 0$ konstansok, míg az ϵ_t valváltozók független, standard normáleloszlású változók. Gyakran $X_{t+1} - X_t = \beta \cdot X_t + \sigma \cdot \epsilon_{t+1}$ alakban szokás felírni az egyenletet, itt persze $\beta = \alpha - 1$ Ezen kívül persze a "készpénzünk" alakulását is számon tartjuk. A közgazdaságtan eszközárazással foglalkozó területein szokásos módon feltesszük, hogy lényegében végtelen hitel áll rendelkezésünkre kamatok nélkül. Szintén megengedett az hogy negatív mennyiségű részvényt tartsunk (*shortolás*), sőt, tetszőleges valós számnyi részvényt tarthatunk.

Optimális kereskedési stratégiát szeretnénk tehát keresni, ehhez először definiálnunk kell a megengedett stratégiák halmazát. Vezessük be az $\mathcal{F}_t = \sigma(X_s, 0 \leq s \leq t)$ σ-algebrák sorozatát. Informálisan fogalmazva: ezek a szigma-algebrák kódolják hogy milyen információ áll rendelkezésünkre a t. időpillanatban. Formálisan ez annak felel meg, hogy a $\mathcal{F}_t$-mérhető függvények pontosan azok, amelyek értéke kiszámolható az $(X_0, \ldots, X_t)$ valószínűségi változók értékeinek ismeretében. Értelemszerű módon mi azokat a függvényeket szeretnénk megengedett stratégiaként definiálni, ami minden pillanatban az addig elérhető információk függvényében ad meg egy kereskedési lépést. Ennek megfelelően **kereskedési stratégia** alatt olyan $(\phi_1, \phi_2, \ldots, \phi_t, \ldots) : \Omega \Rightarrow \mathbb{R}$ függvénysorozatokat fogunk érteni, amelyek esetén mindegyik ϕ_t függvény rendre $\mathcal{F}_{t_1}$-mérhető. A ϕ_t függvény jelentése az, hogy a t. pillanatban hány darabot tartunk a részvényből (mint említettük, ez tetszőleges valós szám lehet). A megengedett statégiák halmazát Φ jelöli.

Egy $(\phi_t)_{t \in \mathbb{N}^+}$ stratégiához hozzárendelhetjük a L_t^ϕ vagyonfolyamatot, amelyet a

$$L_{t-1}^\phi = L_{t-1}^\phi + \phi_t(X_t - X_{t-1})(t \geq 1)$$

rekurzió határoz meg, és értelemszerűen a t időpontban - az akkori kereskedés előtti - összvagyont (készpénz és részvény együtt) adja meg, a részvényt az aktuális árfolyamon számolva. Az L_0 természetesen a befektető kezdőtőkéjének felel meg.

Ennek megfelelően a végső vagyont az

$$L_T^\phi = L_0 + \sum_{j=1}^{T} \phi_j(X_j - X_{j-1})$$

egyenlet adja meg.

Tisztázásra szorul még, hogy pontosan mit értünk optimális stratégiák alatt a problémában. Az ezzel foglalkozó irodalomban általában nem a (várható) összvagyon (L) maximalizálása a cél, hanem egy megfelelő $U : \mathbb{R} \to \mathbb{R}$ függvénnyel a $\mathbb{E}\big[U(L)\big]$ mennyiségé.

Ezen U megnevezése **hasznosságfüggvény** (*utility function*), bevezetését pedig több szempont is motiválja.

A hasznosságfüggvény informálisan azt adja meg, hogy "mennyire örülönk" adott mennyiségű pénznek. Az emberek valós életbeli preferenciáit márpedig sokkal pontosabban leírhatók konkáv hasznosságfüggvényként, mint egy identikus $x \mapsto x$ leképezésként: hiszen adott mennyiségű extra pénz kevésbé hasznos számunkra, amennyiben már eleve több vagyonnal rendelkezünk, mint amikor kevesebbel (ez a *diminishing marginal utility* jelenség). A valóság pontosabb modellezésén kívül pedig az is motiválja a hasznosságfüggvények bevezetését, hogy gyakran könnyebben kezelhetővé teszi matematikailag is a megoldandó problémát.

A cikkben az $U(x) = -e^{-x}$ függvényt fogjuk hasznosságfüggvényként alkalmazni.

Ezen előkészületek után mostmár összefoglalhatjuk a megoldandó problémát, az alábbi módon.

Adott egy piac amelyen a fentebb definiált "részvény" és "készpénz" termékekkel lehet kereskedni; karakterizáltuk a megendedett stratégiák halmazát is. Legyen adott egy T pozitív egész szám, az időhorizont. Az időhorizontig összegyűjtött vagyont szeretnénk optimalizálni az U hasznosságfüggvényre tekintettel, tehát szeretnénk az $\mathbb{E}[U(L_T^{\phi})]$ mennyiséget maximalizálni.

Valójában a probléma két variánsát is megvizsgáljuk: az eddig leírton kívül, mintegy benchmarking céljából, meg fogjuk keresni az optimális stratégiát abban az esetben is, amikor csakis a kezdeti árfolyam ismeretében kell döntést hoznunk a teljes időhorizontig tartó kereskedésre vonatkozóan (tehát előre rögzítenünk kell hogy hogyan szeretnénk kereskedni).

2.2. Fő eredmények

A cikk lényegi állításai az alábbi tételben vannak összegyűjtve:

2.1. Tétel. *A fejezetben ismertetett kereskedési problémában az alábbiak teljesülnek az optimális stratégiákkal kapcsolatban:*

1. *Definiáljuk a* $\hat{\varphi}_t^T(z) := \dfrac{\beta z}{\sigma^2} \cdot \theta_t^T$ *stratégiát, ahol* $\theta_t^T := 1 - (T-t) \cdot \beta$. *Ekkor az optimális stratégiát* $\left(\hat{\varphi}_t^T(X_{t-1})\right)$ *adja meg. Abban a variánsban pedig, ahol előre rögzítenünk*

kell a stratégiát, ott $\left(\hat{\varphi}_t^T(X_0)\right)$ *az optimális.*

2. *Ezen stratégiák szerint kereskedve* $\mathbb{E}\big[U(L_T^{\hat{\varphi}})\big]$ *értéke rendre*

$$\mathbb{E}\big[U(L_T)\big] = -\frac{1}{\sqrt{\gamma_\beta(T)}} \cdot e^{-\frac{\beta^2 X_0^2}{2\sigma^2}T}$$

a sztenderd esetben. Az előre rögzített stratégia esetében pedig

$$e^{-\frac{\beta^2 X_0^2}{2\sigma^2}T}$$

ez az érték.

Itt a γ_β *jelölés az alábbi függvényt takarja:*

$$\gamma_\beta(T) = \begin{cases} \left(\beta^{2T} \cdot \Gamma(\frac{1}{\beta^2} + T)\right)/\Gamma(\frac{1}{\beta^2}) & \text{if } \beta \neq 0 \\ 1 & \text{if } \beta = 0 \end{cases}$$

ahol Γ *pedig a szokásos gamma-függvény.*

3. *Stabil autoregresszív folyamat esetén, azaz mikor* $|\alpha| < 1$*, feltéve hogy* $var(X_t) = 1$ *és hogy* $X_0 \sim \mathcal{N}(0,1)$*, a* $\mathbb{E}\big[U(L_T^{\hat{\varphi}})\big]$ *értéke rendre:*

$$-\sqrt{\frac{\beta+2}{2-(T-1)\beta \cdot \gamma_\beta(T)}}$$

a sztenderd esetben, az előre rögzített stratégiákkal pedig

$$-\sqrt{\frac{\beta+2}{2-(T-1)\beta}}$$

4. γ_β*-ra a* $\lim_{T\to\infty} \frac{\gamma_\beta(T)}{h_\beta(T)} = 1$ *aszimptotika teljesül, ahol*

$$h_\beta(T) = \begin{cases} \Gamma\left(\frac{1}{\beta^2}\right)(\beta^2)^{1-\frac{1}{\beta^2}}\sqrt{2\pi\left(T-1+\frac{1}{\beta^2}\right)}\left(\frac{\left(1+(T-1)\beta^2\right)^2}{e}\right)^{T-1+\frac{1}{\beta^2}}, & \text{if } \beta \neq 0, \\ 1, & \text{if } \beta = 0. \end{cases}$$

A cikk a következőképpen kommentálja a tételt. Egyfelől, a tétel megmutatja hogy szignifikánsan több profitra tehetünk szert egy "interaktív" stratégiával, mint egy előre rögzítettel. Másfelől, ami lényegesebb, azt is bemutatja hogy szignifikánsan függ az elérhető profit a β paramétertől is. A β (vagy ezzel ekvivalensen, az α paraméterre) a cikk többször is "memória"-paraméterként hivatkozik. Ennek az értelme persze az, hogy ez a paraméter kontrollálja hogy egy adott időpontbeli ár (mint valószínűségi változó) mennyire van közel β értékéhez.

3. fejezet

Gépi tanulás alkalmazása kereskedési feladatokon: kutatási eredmények

Ebben a fejezetben kerülnek ismertetésre a saját kutatásunk eredményei. Mint a bevezetőben említettük, a kutatás során gépi tanulásra alkalmas modelleket (speciálisabban neurális hálókat) kíséreltünk meg betanítani az autoregresszív árfolyamatokon való optimális kereskedésre.

A kutatás során Python nyelven írt kódokkal dolgoztunk. A tanulókörnyezet kialakításához a *Gymnasium* frameworköt használtuk, míg a modellek felépítéséhez és a tanítási pipeline-hoz a *stable-baselines3* csomagot.

A kutatási folyamat hosszú ideig kísérletező stádiumban volt. Számos tanítóalgoritmust és architektúrát kipróbáltunk. Mint korábban már említettük, végül a PPO algoritmusra esett a választásunk, de megemlítendők ezek mellett a SAC és A2C algoritmusok is. Azonban figyelembe véve a teljesítményt és az erőforráshatékonyságot is, empirikusan egyértelműen a PPO tűnt a legjobb választásnak.

Azonban úgy tapasztaltuk, hogy még a PPO sem képes kielégítően megoldani az eredeti feladatot, legalábbis az általunk adekvátnak vélt futási idő keretein belül. Ezért némiképpen átalakítottuk a feladatot, a következő formára: ha a jelenlegi ár x_t, akkor $c \cdot x_t$ részvényt vásárolunk, ahol c a tanulóalgoritmus outputja. Tehát egy ilyen módon paraméterezett stratégiában kell a paraméter-együtthatót megtalálni a tanulóalgoritmusnak. Az új feladat alakját persze az motiválta, hogy az optimális megoldás képletében is a jelenlegi ár szerepel egy adott együtthatóval szorozva. Bár ezen átalakítás tulajdonképpen nem jelenti a feladat

különösebb relaxációját, mégis úgy tapasztaltuk, hogy a teljesítmény drámaian megugrott az új feladaton.

A kielégítő teljesítmény elérésében a jutalmazás kalibrálásának is jelentős szerepe volt. Nyilvánvalóan a megoldandó feladat alakjának naivan az a jutalmazási séma felelne meg, hogy minden egyes lefuttatott T időegységig tartó epizód után a kapott jutalom mértéke a $\mathbb{E}\big[U(L_T)\big]$ végső hasznosságnak felel meg. Ez azonban több okból is problémás a reinforcement learning perspektívából.

Egyfelől, nehézséget jelent a jutalmazás *ritkasága*, hogy csak minden egyes epizód végén adódik jutalom, egyszerűen az információ szűkössége okán, különösképp magasabb T horizontok esetén. Másfelől az $U(x) = -e^{-x}$ hasznosságfüggvény sem a legszerencsésebb választás a jutalomfüggvény szerepére. Ezt főleg a függvény $-\infty$ felőli exponenciális csökkenése okozza. Egy reinforcement learning algoritmus számára jellemzően nehézséget jelent az hogyha a jutalmak értékkészlete egy túl széles tartományt ölel fel. De az exponenciális jutalmak még ennél is bonyolultabbá teszik a helyzetet, hiszen esetükben nagyságrendi különbségek vannak gradiensek között például nagy negatív x-ek és pozitív x-ek között. Ez pedig ahhoz vezet, hogy nagyon nehéz egy algoritmust úgy bekalibrálni (pl. a step size paramétert, és hasonló metaparamétereket), hogy hatékonyan kezelje mindkétféle esetet.

Az utóbbi probléma, tehát U exponencialitásának kezeléséhez a $\sigma(x) = \dfrac{1}{1+e^x}$ szigmoid-függvény segítségével normalizáltuk a jutalmakat. A jutalmak ritkaságát megoldandó pedig köztes jutalmakat vezettünk be, melyet az előző időpillanathoz képesti profittal tettünk arányossá. Ki kellett azonban a jutalmazási sémában hangsúlyozni azt, hogy valójában a végső vagyonra optimalizálunk, ezért a végső vagyont egy alkalmas konstanssal felskáláztuk. Több nagyságrendbeli konstanssal kísérletezve végül a 10-szeres súly tűnt adekvátnak. Hasonló módon kísérletezve állítottuk be a többi paraméter értékét. A használt architektúra a korábbi fejezetekből ismerős Actor-Critic architektúra volt, mindkettő 3 darab 512 méretű belső neuronréteget tartalmazott (*rejtett réteg*, az input és output rétegeken kívüli egyéb rétegek). A *learning rate* beállításakor a (relatíve magas értéknek számító) 0.01 értéket találtuk optimálisnak.

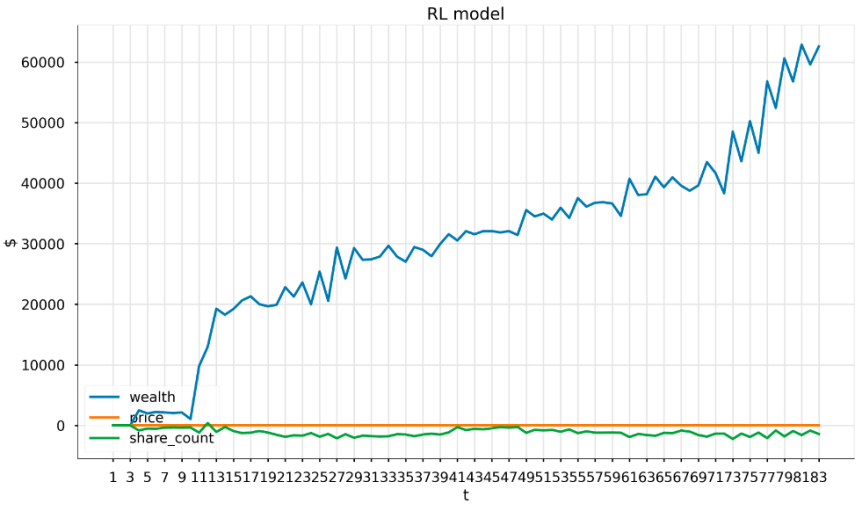
Sajnos a végső beállításokat kevéssel a szakdolgozat határideje előtt véglegesítettem, így nem jutott idő beható kísérletezésre. Az alábbi táblázat tartalmazza az eredményeket. Ezek $\alpha = -0.8$ érték mellett adódtak. Ennél magasabb α értékekre (amik az optimális stratégiát leíró tételben látottak szerint rosszabb eredményt tesznek lehetővé), elvetettem az eredményeket, mert ott jellemzően nem produkált jó teljesítményt (legalábbis logaritmikus

hasznosság tekintetében, a vagyont tekintve jobb teljesítményt mutatott, erről lejjebb).

T	$log(U_T)$	L_T	$opt(U_T)$
10	-75.5	107.8	33
20	-334	22986	67
40	3.7	30637	140
80	-34	37323	294

3.1. táblázat. Eredmények

A táblázatból leolvasható hogy a hasznosság metrika tekintetében a modell teljesítménye nagymértékben elmarad az elvárttól. Ezzel szemben az átlagos vagyont tekintve stabilan növekedő tendenciát tapasztalunk. Ha empirikusan tekintünk tipikus trajektóriákat a modellek futásából, akkor ugyanezt tapasztaljuk: a modell kvázi-monoton módon gyűjti a profitot a futás során. A két metrika közti különbségnek az a magyarázata hogy az utility (és így annak logaritmusa) "pesszimista", azaz egy-egy kiugró negatív trajektória nagyon lehúzza ezt a metrikát. Ez azonban azt jelzi, hogy a modell teljesítménye még további stabilizálásra szorul.



3.1. ábra. Egy tipikus trajektória

4. fejezet

Összefoglalás, kitekintés

[SUMMARY]

A kutatás eddigi anyaga pusztán egy hosszabb kutatási ív elejét képezi. Mint láttuk, a modell jelenlegi teljesítménye nem kielégítő az egyik fontos metrika szerint, így az első cél ez lesz a további kutatás során. Erre több útvonal is kínálkozik.

A legegyszerűbb potenciális megoldás az, hogy jóval hosszabb tanításnak vetjük alá a modelleket. Erre a kutatás eddigi stádiumában a gyorsan iteráló kísérletezés miatt kevesebb lehetőség volt, de ígéretesnek tűnik a teljesítmény skálázódása.

Ígéretes az is, hogy újfajta architektúrákat vessünk be a feladat megoldására, többek között az *attention*-alapú modellek kipróbálása lehet tanulságos.

A jelenlegi feladat kielégítő megoldása után szeretnénk parametrizálatlan ("relaxálatlan") formában is megoldani a feladatot. Könnyen elképzelhető, hogy ez is megoldható pusztán hosszabb tanítással vagy újfajta architektúrák bevezetésével. Ha ezek önmagukban nem vezetnek eredményre, akkor próbálkozhatunk azzal is, hogy különféle *feature extraction* technikákkal segítünk a modellnek a nyers adatok feldolgozásában.

Ha sikerült teljes általánosságában megoldani a feladatot, akkor is számos érdekes kérdés merül fel a probléma horizontján. Ilyen például az, hogy egy kellően nagy kapacitású modell esetén megfigyelhetünk-e tudástranszfert, tehát például egy olyan modell, ami a tanítása során csak bizonyos paraméterű autoregresszív folyamatokkal találkozott, vajon tudja-e kezelni a másféle paramétereket. Ha igen, vajon meddig terjedhet egy ilyen modell flexibilitása? Készíthető-e olyan modell ami agnosztikus az árfolyamat típusára, mindenképpen képes profitábilisan kereskedni rajta, ha a folyamat esetén erre lehetőség van. Ha ez

nem is valósítható meg, a jelen kutatással analóg kérdések számos más árfolyamat-típussal kapcsolatban feltehetők, és ezek hasonlóan érdekes kutatás tárgyát képezhetik..

Irodalomjegyzék

[1] Stephen Boyd. Convex optimization. https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf, 2004.

[2] Sándor Deák and Miklós Rásonyi. An explicit solution for optimal investment problems with autoregressive prices and exponential utility. *arXiv:1501.01506*, 2015.

[3] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. https://sham.seas.harvard.edu/sites/projects.iq.harvard.edu/files/kakade/files/aoarl-2002.pdf, 2002.

[4] John Schulman, Sergey Levine, Phillip Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *arXiv:1502.05477*, 2015.

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

[6] Richard G. Sutton and Andrew G. Barto. Reinforcement learning, 2018.

[7] Zsigmond Tarcsay. Funkcionálanalízis. https://oszkdk.oszk.hu/storage/00/02/62/53/dd/1/Funkcion_lanal_zis_ISBN_szammal.pdf, 2018.

[8] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.