Mesterséges intelligencia alapjai jegyzet

Molnár Antal Albert

2020. június 18.

Előszó

Ez a jegyzet a Debreceni Egyetemen a Kádek Tamás által oktatott, *INBPM0418E* tárgykódú *A mesterséges intelligencia alapjai* tárgyhoz nyújt némi segítséget a vizsgára készülő hallgatóknak.

A leírtakért semmilyen felelősséget nem tudok vállalni, hiszen még jómagam is csak ismerkedem a mesterséges intelligencia világával.

A jegyzet nagyrészt az órákon tárgyalt könyvből[1], valamint Várterész Magda előadásaiból[2] merít.

Tartalomjegyzék

1. fejezet

Témakörök, melyeket mélységben ismerni kell

1.1. Ágens szemlélet

1.1.1. Az ágens fogalma

1. Definíció. Ágens. Egy **ágens** (agent) bármi lehet, amit úgy tekinthetünk, mint ami az **érzékelői** (sensors) segítségével érzékeli a **környezetét** (environment), és **beavatkozói** (actuators) segítségével megváltoztatja azt.

Az emberi ágensnek van szeme, füle és egyéb szervei az érzékelésre, és keze, lába, szája és egyéb testrészei a beavatkozásra.

A robotágens kamerákat és infravörös távolsági keresőket használ érzékelőként, és különféle motorokat beavatkozóként.

A szoftverágens billentyűleütéseket, fájltartalmakat és hálózati adatcsomagokat fogad érzékelőinek bemeneteként, és képernyőn történő kijelzéssel, fájlok írásával, hálózati csomagok küldésével avatkozik be a környezetébe.

Azzal az általános feltételezéssel fogunk élni, hogy minden ágens képes saját akcióinak érzékelésére (de nem mindig látja azok hatását).

2. Definíció. Érzékelés. Az érzékelés (percept) fogalmat használjuk az ágens érzékelő bemeneteinek leírására egy tetszőleges pillanatban.

3. Definíció. Ágens érzékelési sorozata. Egy ágens érzékelési sorozata (percept sequence) az ágens érzékeléseinek teljes története, minden, amit az ágens valaha is érzékelt. Általánosságban, egy adott pillanatban egy ágens cselekvése az addig megfigyelt teljes érzékelési sorozatától függhet. Ha az összes lehetséges érzékelési sorozathoz meg tudjuk határozni az ágens lehetséges cselekvéseit, akkor lényegében mindent elmondtunk az ágensről.

4. Definíció. Ágensfüggvény. Matematikailag megfogalmazva azt mondhatjuk, hogy az ágens viselkedését az ágensfüggvény (agent function) írja le, ami az adott érzékelési sorozatot egy cselekvésre képezi le.

5. Definíció. Ágensprogram. Egy mesterséges ágens belsejében az ágensfüggvényt egy ágensprogram (agent program) valósítja meg.

1. Megjegyzés. Különbség ágensfüggvény és ágensprogram között. Az ágensfüggvény egy absztrakt matematikai leírás, az ágensprogram egy konkrét implementáció, amely az ágens architektúráján működik.

1.1.2. Az ágens jellemzése (teljesítmény, környezet, érzékelők, beavatkozók

6. Definíció. TKBÉ (PEAS) leírás. Egy ágens tervezése során az első lépésnek mindig a feladatkörnyezet lehető legteljesebb meghatározásának kell lennie.

Ágenstípus	**Teljesítménymérték (Performance)**	**Környezet (Environment)**	**Beavatkozók (Actuators)**	**Érzékelők (Sensors)**
Taxisofőr	Biztonságos, gyors, törvényes, kényelmes utazás, maximum haszon	Utak, egyéb forgalom, gyalogosok, ügyfelek	Kormány, gáz, fék, index, kürt, kijelző	Kamerák, hangradas, sebességmérő, GPS, kilométeróra, motorérzékelők, billentyűzet

7. Definíció. Teljesen megfigyelhető környezet.

Ha az ágens szenzorai minden pillanatban hozzáférést nyújtanak a környezet teljes állapotához, akkor azt mondjuk, hogy a környezet teljesen megfigyelhető.

8. Definíció. Determinisztikus (deterministic) vagy sztochasztikus (stochastic).

Amennyiben a környezet következő állapotát jelenlegi állapota és az ágens által végrehajtott cselekvés teljesen meghatározza, akkor azt mondjuk, hogy a környezet determinisztikus, egyébként sztochasztikus.

9. Definíció. Epizódszerű (episodic) vagy sorozatszerű (sequential) környezet.

Epizódszerű környezetben az ágens tapasztalata elemi "epizódokra" bontható. Minden egyes epizód az ágens észleléseiből és egy cselekvéséből áll. Nagyon fontos, hogy a következő epizód nem függ az előzőben végrehajtott cselekvésektől. Epizódszerű környezetekben az egyes epizódokban az akció kiválasztása csak az aktuális epizódtól függ.

10. Definíció. Statikus (static) vagy dinamikus (dynamic) környezet.

Ha a környezet megváltozhat, amíg az ágens gondolkodik, akkor azt mondjuk, hogy a környezet az ágens számára dinamikus; egyébként statikus.

11. Definíció. Diszkrét (discrete) vagy folytonos (continuous) környezet.

A diszkrét/folytonos felosztás alkalmazható a környezet állapotára, az időkezelés módjára, az ágens észleléseire, valamint cselekvéseire. Például egy diszkrét állapotú környezet, mint amilyen a sakkjáték, véges számú különálló állapottal rendelkezik. A sakkban szintén diszkrét az akciók és cselekvések halmaza. A taxivezetés folytonos állapotú és idejű probléma: a sebesség, a taxi és más járművek helye folytonos értékek egy tartományát járja végig a folytonos időben.

12. Definíció. Egyágenses (single agent) vagy többágenses (multiagent) környezet.

Az egyágenses és többágenses környezetek közötti különbségtétel egyszerűnek tűnhet. Például a keresztrejtvényt megfejtő ágens önmagában nyilvánvalóan egyágenses környezetben van, míg egy sakkozó ágens egy kétágensesben. Vannak azonban kényes kérdések. Először is: leírtuk azt, hogy egy entitás hogyan tekinthető ágensnek, ugyanakkor nem magyaráztuk meg, mely entitások tekintendők ágensnek. Egy A ágensnek (például a taxisofőrnek) egy B objektumot (egy másik járművet) ágensnek kell tekintenie, vagy egyszerűen egy sztochasztikusan viselkedő dolognak, a tengerparti hullámokhoz vagy a szélben szálló falevelekhez hasonlatosan? A választás kulcsa az, hogy vajon B viselkedése legjobban egy A viselkedésétől függő teljesítménymérték maximalizálásával írhatóe le. Például a sakkban a B ellenfél saját teljesítménymértékét próbálja maximalizálni, amely – a sakk szabályainak következtében – A teljesítménymértékét minimalizálja. Így a sakk egy versengő (**competitive**) többágenses környezet. Másrészről, a taxi vezetési környezetben az ütközések elkerülése az összes ágens teljesítménymértékét maximálja, így az részben kooperatív (**cooperative**) többágenses környezet. Emellett részben versengő is, hiszen például csak egy autó tud egy parkolóhelyet elfoglalni. A többágenses környezetekben felmerülő ágenstervezési problémák gyakran egészen mások, mint egyágenses környezetekben. Többágenses környezetekben például a kommunikáció (communication) gyakran racionális viselkedésként bukkan fel; egyes részlegesen megfigyelhető versengő környezetekben a sztochasztikus viselkedés racionális, hiszen így elkerülhetők a megjósolhatóság csapdái.

1.2. Állapottér reprezentáció

1.2.1. Az állapottér fogalma

Az állapottér-reprezentáció az egyik leggyakrabban használt reprezentációs mód ami egy probléma formális megadására szolgál.

13. Definíció. Probléma. Egy **probléma** (problem) formális megragadásához az alábbi négy komponensre van szükség:

- **kiinduló állapot**: amiből az ágens kezdi a cselekvéseit,
- **cselekvések halmaza**: ágens rendelkezésére álló lehetséges cselekvések,
- **állapotátmenet-függvény**: visszaadja a rendezett ⟨ cselekvés, utódállapot ⟩ párok halmazát (Egy alternatív megfogalmazás az **operátor**ok egy halmaza, amelyeket egy állapotra alkalmazva lehet az utódállapotokat generálni.) lásd még: *14. Definíció*
- **állapottér**: A kezdeti állapot és az állapotátmenet-függvény együttesen implicit módon definiálják a probléma **állapotteré**t: azon állapotok halmazát, amelyek a kiinduló állapotból elérhetők.

14. Definíció. Operátor alkalmazási előfeltétel teszt. Ahhoz, hogy meggyőződjünk arról, hogy egy adott állapotra egy adott operátor alkalmazható, előbb meg kell vizsgálnunk, hogy az állapotra alkalmazható-e az operátor. Ezt a vizsgálódást operátor alkalmazási előfeltétel tesztnek nevezzük.

15. Definíció. Célteszt. A célteszt ellenőrzi a célfeltételek teljesülését egy adott állapotban.

1.2.2. Az állapottérgráf

16. Definíció. Állapottérgráf. Az állapottér egy gráfot alkot, amelynek csomópontjai az állapotok és a csomópontok közötti élek a cselekvések.

17. Definíció. Állapottér útja. Az **állapottér egy útja** az állapotok egy sorozata, amely állapotokat a cselekvések egy sorozata köt össze.

18. Definíció. Állapottér-reprezentációs gráf bonyolultsága. Egy állapottér-reprezentált probléma megoldásának sikerét jelentősen befolyásolja a reprezentációs gráf bonyolultsága:

- a csúcsok száma,
- az egy csúcsból kiinduló élek száma,
- a hurkok és körök száma és hossza.

Ezért célszerű minden lehetséges egyszerűsítést végrehajtani. Lehetséges egyszerűsítések:

- a csúcsok számának csökkentése — ügyes reprezentációval az állapottér kisebb méretű lehet;
- az egy csúcsból kiinduló élek számának csökkentése — az operátorok értelmezési tartományának alkalmas megválasztásával érhető el;
- a reprezentációs gráf fává alakítása — a hurkokat, illetve köröket „kiegyenesítjük"

1.2.3. Költség és heurisztika fogalmak

19. Definíció. Lépésköltség. Az x állapotból az y állapotba vezető cs cselekvés **lépésköltség**e (step cost) legyen $lk(x, cs, y)$. Tételezzük fel, hogy a lépésköltség nemnegatív.

20. Definíció. Útköltség-függvény. Egy **útköltség-függvény**, egy olyan függvény amely az állapottér minden útjához hozzárendel egy költséget.

21. Definíció. Megoldás. Egy út, amely a kiinduló állapotból egy célállapotba vezet.

22. Definíció. Optimális megoldás. A legkisebb útköltségű megoldás.

1.3. Megoldáskereső algoritmusok

1.3.1. Fakereső algoritmusok

1.3.2. Gráfkereső algoritmusok

1.3.3. Szélességi kereső

1.3.4. Mélységi kereső

1.3.5. Visszalépéses kereső

1.3.6. Egyenletes költségű (optimális kereső)

1.3.7. Legjobbat először kereső

1.3.8. Az A* algoritmus

1.4. Kétszemélyes játékok

1.4.1. A játékok reprezentációja

A továbbiakban a

- teljesen megfigyelhető,
- véges,
- determinisztikus,
- kétszemélyes,
- zérusösszegű

játékokkal foglalkozunk.

23. Definíció. Játék reprezentációja. Egy játék reprezentációja megadható a

$$\langle \mathcal{B}, b_0, \mathcal{J}, \mathcal{V}, \hat{v}, \mathcal{L} \rangle$$

rendezett hatossal, ahol:

- $\mathcal{B}$: a játékállások halmaza,
- b_0: a kezdőállás, ahol $b_0 \in \mathcal{B}$,
- $\mathcal{J}$: a játékosok halmaza, ahol $\operatorname{card} \mathcal{J} = 2$,
- $\mathcal{V}$: a végállások halmaza, ahol $\mathcal{V} \subseteq \mathcal{B}$,
- $\hat{v}$: egy $\mathcal{V} \to \{-1, 0, 1\}$ függvény,
- $\mathcal{L}$: a lépések halmaza.

24. Definíció. Nyertes. Végállásban a $\hat{v}$ függvény határozza meg, hogy melyik játékos nyert:

$$\hat{v}(b) = \begin{cases} 1 & \text{ha } b \text{ állásban a következő játékos nyer} \\ 1 & \text{ha } b \text{ állásban a következő játékos veszít .} \\ -1 & \text{egyébként (döntetlen esetén)} \end{cases}$$

25. Definíció. Lépés. Minden $l \in \mathcal{L}$ egy $\mathcal{B} \to \mathcal{B}$ parciális függvény.

26. Definíció. Játék állapottere. Legyen $\langle \mathcal{B}, b_0, \mathcal{J}, \mathcal{V}, \hat{v}, \mathcal{L} \rangle$ egy játék reprezentációja. Ekkor a **játék állapottere** $\langle \mathcal{A}, a_0, \mathcal{C}, \mathcal{O} \rangle$ definiálható a következőképpen:

- $\mathcal{A} = \mathcal{B} \times \mathcal{J}$;
- $a_0 = \langle b_0, j_0 \rangle$, ahol $j_0 \in \mathcal{J}$ a kezdőjátékos;
- $\mathcal{C} = \{\langle b, j \rangle : \langle b, j \rangle \in A \wedge b \in \mathcal{V}\}$
- $\mathcal{O} = \{o_l : l \in \mathcal{L}\}$, ahol $o_l : \mathcal{A} \to \mathcal{A}$, úgy hogy
  - $\mathrm{dom}(o_l) = \{\langle b, j \rangle : b \in \mathrm{dom}(l)\}$
  - $o_l(\langle b, j \rangle) = \langle l(b), j' \rangle$
  - $j' \in (\mathcal{J} \setminus \{j\})$

27. Definíció. Közvetlen elérhetőség. Az $a \in \mathcal{A}$ állapotból az $a' \in \mathcal{A}$ állapot **közvetlenül elérhető**, ha van olyan $o_l \in \mathcal{O}$, amely esetén

$$a \in \mathrm{dom}(o_l) \text{ és } o_l(a) = a'$$

és ezt $a \Rightarrow a'$ alakban jelöljük.

28. Definíció. Elérhetőség. *Aza* $\in \mathcal{A}$ állapot az $a' \in \mathcal{A}$ állapot **elérhető** ($a \Rightarrow^* a'$), ha

- $a = a'$, vagy
- van olyan $a_1, a_2, \ldots, a_k$ állapotsorozat, hogy $a_1 = a, a_k = a'$ továbbá

$$a_i \Rightarrow a_{i+1} \text{ minden } i \in \{1, \ldots, k-1\} \text{ esetén.}$$

1.4.2. A játékfa

29. Definíció. Játékfa. Legyen $\langle \mathcal{B}, b_0, \mathcal{J}, \mathcal{V}, \hat{v}, \mathcal{L} \rangle$ egy játék reprezentációja és $j_0 \in \mathcal{J}$ a kezdőjátékos. Ekkor a játékfa olyan fa, melynek csúcsaihoz a játék állapotait rendeljük:

- a fa gyökere a $\langle a_0, j_0 \rangle$ állapottal címkézett csúcs,
- a fa levélelemei olyan $\langle b, j \rangle$ állapottal címkézett csúcsok, ahol $b \in \mathcal{V}$
- a fa $\langle b, j \rangle$ állapottal címkézett nem levélcsúcsának gyermekeit olyan $\langle b', j' \rangle$ címkéjű csúcsok alkotják, ahol $\langle b, j \rangle \Rightarrow \langle b', j' \rangle$

2. Megjegyzés. A játékfa a játék állapotterének gráfját fává egyenesíti ki, egy-egy állapot a fában több csúcs címkéjeként is szerepelhet.

1.4.3. Nyerő stratégia

30. Definíció. Stratégia. Játékterv, ami minden olyan állásban, amikor ő következik, megmondja a játékosnak, hogy mit lépjen.

31. Definíció. Nyerő stratégia. Játékterv, ami minden olyan állásban, amikor ő következik, megmondja a játékosnak, hogy mit lépjen.

1.5. Lépésajánló algoritmusok

1.5.1. MinMax módszer

1.5.2. NegaMax módszer

1.5.3. Alfa-béta nyesés

1.6. Élkonzisztencia algoritmusok

32. Definíció. Kényszerek terjesztése. A **kényszerek terjesztése** során egy tekintett változó értékére vonatkozó megszorítás következményeit a vele kényszerek útján kapcsolatban álló változók értékeire vonatkozóan is kiterjesztjük, ezen változók értékeire újabb megszorításokat alkalmazva.

33. Definíció. Élkonzisztencia algoritmusok. Az **élkonzisztencia algoritmusok** feladata, a kényszerek terjesztésének hatékony megvalósítása. Ezen algoritmusok egyaránt alkalmazhatóak a keresés megkezdése előtt a probléma méretét csökkentő előfeldolgozó lépésként, vagy akár a keresés közben is. Ezek közül mi az előbbi lehetőséget vizsgáljuk meg.

1.6.1. AC1

Az AC-1 (gyakorlatban nem használt) algoritmus egyszerű naiv megközelítés segítségével mutatja be a kényszerek terjesztésének ötletét.

34. Definíció. Felülvizsgálat. Az $R_{x,y}$ **kényszer felülvizsgálat**a során a $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$ bináris kényszereket tartalmazó véges kényszerkielégítési problémát a $\langle \mathcal{V}, \mathcal{D}', \mathcal{C}' \rangle$ problémával helyettesítjük, ahol

$$\mathcal{D}' = \begin{cases} \{a : a \in \mathcal{D}(x) \wedge b \in D(y) \wedge \langle a, b \rangle \in R_{x,y}\} & \text{ha } z = x \\ \mathcal{D}(z) & \text{egyébként.} \end{cases}.$$

Algorithm 1: AC-1

```
1 Function AC1(⟨V, D, C⟩):
2     repeat
3         D' ← D
4         forall R_{x,y} ∈ C do
5             D ← revise(R_{x,y}, ⟨V, D, C⟩)
6         end forall
7     until D = D'
8     return ⟨V, D, C⟩
```

1.6.2. AC3

Az AC-3 algoritmus (Mackworth) egy már a gyakorlatban is használható algoritmus, amely egy sorban tartja nyilván azokat a kényszereket, amelyeket felül kell vizsgálni.

Ha $R_{x,y}$ felülvizsgálat során az x változó tartománya megváltozik, akkor minden olyan kényszer ismét bekerül a sorba, amely $R_{y,x}$ alakú.

Az algoritmus akkor fejeződik be, mikor elfogytak a sorból a kényszerek.

Az algoritmus időbonyolultsága $\mathcal{O}(n^2 \cdot d^3)$.

Algorithm 2: AC-3

```
1  Function AC3(⟨V, D, C⟩):
2      while W ≠ ∅ do
3          remove an R_{x,y} constraint from W
4          D' ← D
5          D ← revise(R_{x,y}, ⟨V, D, C⟩)
6          if D' ≠ D then
7              if D(x) = ∅ then
8                  return failure
9              else
10                 forall R_{u,w} ∈ {R_{u,w} : R_{u,w} ∈ C ∧ w = x} do
11                     W ← W ∪ {R_{u,w}}
12                 end forall
13             end if
14         end if
15     end while
16     return ⟨V, D, C⟩
```

1.6.3. AC4

Az AC-4 algoritmus (Mohr és Henderson) két előre elkészített adatszerkezettel dolgozik:

- minden $R_{x,y}$ kényszer és $v_x \in \mathcal{D}(x)$ értékhez egy számlálót rendelünk:

$$C_{x,v_x,y} = \text{card}\{v_y : v_y \in \mathcal{D}(y) \wedge \langle v_x, v_y \rangle \in R_{x,y}\},$$

- az x változó minden $v_x \in \mathcal{D}(x)$ értékéhez egy halmazt rendelünk:

$$S_{x,v_x} = \{\langle y, v_y \rangle : v_y \in \mathcal{D}(y) \wedge \langle v_y, v_x \rangle \in R_{y,x}\},$$

  amely megmutatja, hogy mely y változók mely $v_y \in \mathcal{D}(y)$ értékeihez számoltuk hozzá v_x-et.

Jelölje $\mathcal{D}_{-\langle x,v_x \rangle}$ azt a leképezést, amely $\mathcal{D}$ leképezéstől annyiban különbözik, hogy $v_x \notin \mathcal{D}(x)$, vagyis x nem veheti fel v_x értéket:

$$\mathcal{D}_{-\langle x,v_x \rangle}(z) = \begin{cases} \mathcal{D}(x) \setminus \{v_x\} & \text{ha } z = x, \\ \mathcal{D}(z) & \text{egyébként.} \end{cases}$$

Algorithm 3: AC-4 inicializálás

```
1  Function AC4-initialize(⟨V, D, C⟩):
2      Compute S
3      Compute C
4      forall ⟨x, v_x⟩ ∈ {⟨x, v_x⟩ : C_{x,v_x,y} = 0} do
5          D ← D_{−⟨x,v_x⟩}
6      end forall
7      return S, C, D
```

Algorithm 4: AC-4

```
1  Function AC4(⟨V, D, C⟩):
2      S, C, D ← AC4-initialize(⟨V, D, C⟩)
3      W ← {⟨x, v_x⟩ : C_{x,v_x,y} = 0}
4      while W ≠ ∅ do
5          remove an ⟨x, v_x⟩ pair from W
6          forall ⟨y, v_y⟩ ∈ S_{x,v_x} do
7              C_{y,v_y,x} ← C_{y,v_y,x} − 1
8              if C_{y,v_y,x} = 0 and v_y ∈ D_y then
9                  D ← D_{−⟨y,v_y⟩}
10                 insert the ⟨y, v_y⟩ pair into W
11             end if
12         end forall
13     end while
14     return S, C, D
```

Az **AC-4 algoritmus** egy váltótó-érték párokat tartalmazó munkahalmaz segítségével végzi el feladatát. Azon értékek, amelyek által C számlálója 0-ra

csökken törölni kell a változó tartományából. A munkahalmazba a törtölt értékek kerülnek, mert a tőlük függő értékek számlálóit csökkenteni kell. Ezek gyors felderítésében segít S.

3. Megjegyzés. Az AC-4 algoritmus időbonyolultsága: $\mathcal{O}(n^2 \cdot d^2)$. Ennek ellenére valódi problémák esetében sok esetben az AC-3 algoritmus teljesít jobban (Richard J. Wallace).

1.6.4. Visszalépéses kereső

2. fejezet

Témakörök, melyekre rálátással kell rendelkezni

2.1. Következtetések ítéletlogikában

2.1.1. Rezolúciókalkulus ítéletlogikában

2.2. Döntési fák

2.2.1. Az ID3 algoritmus

2.3. Valószínűségi következtetés

2.3.1. Bayes hálok, Bayes tétel

2.3.2. Feltételes valószínűség számítása

2.4. Neurális hálók, kitekintés

2.4.1. Nuerális hálók és mélytanulás

Irodalomjegyzék

[1] **Russell** Stuart J, **Norvig** Peter: Mesterséges Intelligencia modern megközelítésben. Panem Kft., 2005.

[2] Várterész Magda: A mesterséges intelligencia alapjai: Az előadások mellé vetített anyag (2011).