

# Bevezetés a gépi tanulásba

DR. BOTZHEIM JÁNOS, GULYÁS LÁSZLÓ és NAGY BALÁZS  
magyar nyelvű előadásai alapján

*Utoljára frissítés: 2024. június 13.*

## Előszó

Ez a jegyzet a 2023/2024/2. tavaszi félévben készült, mely a *Bevezetés a gépi tanulásba* című tárgy **magyar nyelvű** előadásait dolgozza fel. Elsődleges forrásaim a levetített prezentációk voltak, valamint azon magyarázatok, melyek elhangoztak.

A jegyzet a szó szoros értelmében jegyzetnek minősül, azaz a magyarázatok messze nem olyan precízek, mint amilyeneket egy hivatalos egyetemi jegyzet megkövetelne, valamint a szerkezete is inkább vázlatpontokban foglaja össze tömörén a lényeget. Emiatt **semmiképp sem helyettesíti az előadás anyagait**, legfeljebb egy áttekinthetőbb összefoglalást biztosít.

Igyekeztem a legjobb tudásom szerint összeállítani a jegyzetet, ennek ellenére előfordulhatnak benne elgépelések, hibák, stb. Ha találsz illet, kérlek értesíts e-mailben a(z) ap3558@inf.elte.hu címen.

Sikeres felkészülést kívánok!

*Kiss-Bartha Nimród*

## Rövidítések

- **GT** – gépi tanulás  $\iff$  ML – machine learning
- **MI** – mesterséges intelligencia  $\iff$  AI – artificial intelligence
- **RL** – megerősített tanulás (*reinforcement learning*)

## Források

- [1.] Az előadás prezentációi (*Canvason elérhetők*)
- [2.] Valentinusz (Boda Bálint és mások) jegyzete (*angol nyelvű*) [[link](#)]
- [3.] BME-s jegyzet (*magyar nyelvű*) [[pdf](#)]

## Tartalomjegyzék

<b>1. Mesterséges intelligencia vs. gépi tanulás – Bevezető</b>	<b>4</b>
1.1. Mi az a gépi tanulás? . . . . .	4
1.2. Mióta létezik? . . . . .	4
1.3. Mik a hátráltató tényezők? . . . . .	4
1.4. Az MI értelmezése . . . . .	5
1.5. Az MI történelme . . . . .	6

<b>2. Napjaink MI-je – Optimalizáció</b>	<b>7</b>
2.1. Mitől optimalizáció? . . . . .	7
2.2. Intelligens viselkedés . . . . .	7
2.3. Mitől nehéz az optimalizáció? . . . . .	9
2.4. Hogyan próbálkozhatunk? . . . . .	9
<b>3. Felügyelt tanulás</b>	<b>10</b>
3.1. Felügyelet nélküli tanulás – visszatekintés . . . . .	10
3.2. Felügyelt tanulás . . . . .	10
3.3. A tanulás típusai . . . . .	10
3.4. Mi az a felügyelt tanulás? . . . . .	11
3.5. Problémák . . . . .	11
3.6. Döntési fák . . . . .	11
3.7. Regresszió . . . . .	13
<b>4. Neurális hálózatok</b>	<b>14</b>
4.1. Biológiai és mesterséges neuronok . . . . .	14
4.1.1. Biológiai neuronok . . . . .	14
4.1.2. Mesterséges neuronok . . . . .	14
4.2. Egyrétegű perceptronok . . . . .	15
4.2.1. Lineáris szeparabilitás . . . . .	15
4.2.2. Logikai kapuk . . . . .	16
4.3. Perceptronok tanítása . . . . .	17
<b>5. Etikai és jogi kérdések az MI területén</b>	<b>21</b>
5.1. Az MI eredményei és nem kívánt következményei . . . . .	21
5.2. Az MI és a művészeti viszonya – Théâtre d’opéra spatial . . . . .	21
5.3. Felmerülő gondok az adatbázisa kapcsán . . . . .	22
5.4. Korábbi találkozásaink az MI-vel . . . . .	22
<b>6. Felügyelet nélküli tanulás</b>	<b>23</b>
6.1. Fő különbségek a felügyelt tanulástól . . . . .	23
6.2. Klaszterezés . . . . .	23
6.2.1. Agglomeratív klaszterezés . . . . .	24
6.2.2. Prototípus alapú klaszterezés ( $k$ -means clustering) . . . . .	24
6.2.3. DBSCAN . . . . .	27
6.3. Dimenziócsökkentés . . . . .	27
<b>7. Megerősített tanulás</b>	<b>28</b>
7.1. Mi az a megerősített tanulás? . . . . .	28
7.2. Háttere, ágazatai . . . . .	28
7.3. Felépítése . . . . .	29
7.4. Modell alapú RL – Markov döntési folyamat . . . . .	30
7.5. Modellmentes RL – Mély RL algoritmusok . . . . .	32
<b>8. Biológiai alapú MI megoldások</b>	<b>34</b>
8.1. Inspiráció, motiváció . . . . .	34
8.2. Rajointelligence – példák, motivációk . . . . .	34
8.3. Térbeli csoportosítás – spacial clustering . . . . .	34
8.4. Legrövidebb utak . . . . .	35
<b>9. Evolúciós algoritmusok</b>	<b>36</b>
9.1. Motiváció . . . . .	36
9.2. Háttere, közös jellemzők . . . . .	36
9.3. Genetikus algoritmusok . . . . .	36

<b>10. Rajintelligencia</b>	<b>38</b>
10.1. Bevezetés, motiváció . . . . .	38
10.2. Particle swarm optimisation (részecske raj optimalizáció) . . . . .	38
10.3. Simplified Swarm Optimisation . . . . .	39
10.4. Szentjánosbogár algoritmus (firefly algorithm) . . . . .	39
10.5. Gravitációs keresőalgoritmus . . . . .	39
<b>11. Etorobotika</b>	<b>40</b>
11.1. Motiváció . . . . .	40
11.2. Etorobotika . . . . .	40
11.3. Etológiai mintánk . . . . .	40
11.4. Etológiai kutatási folyamat . . . . .	41
11.5. Kísérletek . . . . .	41
<b>12. Multiágens szimuláció és tanulás (Multi-agent simulation and learning)</b>	<b>42</b>
12.1. Motiváció . . . . .	42
12.2. Mikro- és makroviselkedés . . . . .	42
12.3. Ágens alapú szimuláció . . . . .	42
12.4. Multiágens megerősített tanulás . . . . .	42

# 1. Mesterséges intelligencia vs. gépi tanulás – Bevezető

## 1.1. Mi az a gépi tanulás?

- fogalmak tisztázása: gépi tanulás (GT)  $\neq$  mesterséges intelligencia (AI)
- a két fogalom kapcsolódik egymáshoz, de nem ugyanarra vonatkoznak
  - az MI sokkal régebbre nyúlik vissza ('50-es évek)
  - a GT frissebb, az MI-ből alakult ki
  - (mondhatnánk, hogy  $GT \subset AI$ )
  - viszont mégis összekapcsolónak, ugyanis jelenleg a **mély gépi tanulás** (*deep machine learning*) uralja az MI-t

## 1.2. Mióta létezik?

- 1956 óta vannak ilyen célú kutatások
  - Dartmouth konferencia, ahol megjelentek az MI alapítói
  - azt kutaták, hogyan lehet mesterséges intelligenciát csinálni
  - másik szempontjuk, hogy hogyan lehet eladni
  - elnevezés: **mesterséges intelligencia** → marketing húzás (támogatók, szponzorok figyelmét felkeltsék vele)
- csak az elmúlt években robbant be a kutatási terület
  - a kutatások folyamatosak voltak '56 óta
  - csak az elért **eredményekről nem gondoljuk, hogy intelligensek**
  - a támogatás „hullámzó” volt (erről később)
- felmutatható **eredmények**
  - GPS, navigáció  $\leftarrow$  gráfkereső algoritmusok (BFS, DFS)
  - nyelvhelyesség-ellenőrző Wordben
  - postákon OCR (optikai karakterfelismerő) rendszerekkel szortírozzák a csomagokat már a '80-as évek óta (!)
  - szövegkövetkeztető (text prediction)

## 1.3. Mik a hátráltató tényezők?

- mi az az **intelligencia**? → nincs univerzálisan elfogadott definíció
  - olyan tulajdonság, amivel az ember rendelkezik... és mások?
  - egyetlen példából nehéz általánosítani
  - eddig „legjobb” megfogalmazás: egy olyan **gép**, hogy gép + digitalizáció = MI
- ezt a gépet az ember alkotja meg, hogy az *jobbá* váljon az embernél (jobban teljesítsen feladatakat nála)
  - mit jelent, hogy *jobban teljesítsen*?
  - pontosabban? gyorsabban? olcsóbban?
  - ilyen értelemben a varrógép is mesterséges intelligencia... ez kicsit kiábrándító

- a legtöbbször úgy épülnek be az életünkbe ezek az „intelligens” dolgok, hogy fel sem tűnik a létük
  - mintha egy közmű lenne
  - pl. az elektromosság megjelenésekor mindenki démonizálta az új technológiát, 100 évvel később anapság akkor pánikolunk, amikor nincs áram
  - hasonló érzelmek / gondolatok forognak az MI körül is

#### 1.4. Az MI értelmezése

A) Szűk MI (narrow AI):

- jelenlegi korunk
- adott feladatot (feladatcsoportot) jobban csinál, mint az ember
- ekkor a szűk értelmezésben intelligensnek nevezhetők

B) Általános MI (AGI – artificial general intelligence)

- felülmúlja az emberi intelligenciát
- minden feladatot képes elvégezni

C) Egyéb elméleti koncepciók:

- szingularitás: fel sem tudjuk fogni a fejlődés mértékét (irodalmi művekből ered a fogalom)
- szuperintelligencia
- öntudatra ébredés (→ mi az az öntudat?)

A **vita** arról szól, hogy *még nem* vagy *már* eljutottunk-e az AGI-ba? Ennek előírására **teszteket** állítottunk elő.

1. Turing-teszt:

- ha lehet vele beszélgetni egy interfésze keresztül úgy, hogy 30 perc eltelte után nem tudjuk eldönteni, hogy ember-e vagy gép → a gép intelligens
- eredmény: elértek
- ChatGPT átmenne a teszten (szigorú értelemben nem, mert ha megkérdezzük tőle, hogy ő ember-e vagy gép, (egyelőre) gépet fog mondani)

2. Robot hallgató-teszt:

- (mármint egyetemi diákok)
- beíratjuk egy egyetemre; ha lediplomázik, mint egy hallgató → a gép intelligens
- eredmény: egyesek átmentek rajta

3. Employment test

- állást kereső robot; ha felveszik → intelligens
- nehéz tesztelni

4. Egyéb tesztek:

- IKEA-teszt: ha össze tud rakni egy IKEA-s bútor, akkor intelligens
- Kávé-teszt: ha tud készíteni kávét, akkor intelligens
  - a trükk a feladatban megvívó komplexitás

- sok a be nem számítható változó (polc magassága, hol találja a kávét, ismeri-e a ketyogós használatát, stb.)
- nem sikerült elérni semminem / senkinek

Praktikus dolgokban alulmaradnak, az ember jobb bennük.

### 1.5. Az MI történelme

- hullámokban érkeztek / érkeznek a fejlődések
- megoszlanak a nézetek, hány hullám volt / van
- kutatáspromócióra tökéletesen használható a terület ezen természete

1956-ban megszületik az **MI**, mint fogalom



hatalmas **támogatás**, érdeklődő hallgatók



de **nem** jelentek meg hasznosnak, *intelligensnek*  
minősülő eredmények → az *első MI tél*

- 5-10 évvel később: fiatalabb kutatók újabb ötletekkel érkeztek
  - mantra: „az első társaság rosszul csinálta, de mi tudjuk”
  - ugyanúgy elbuktak
- most úljabb felfele ívelés tapasztalható
- a régebbi eredmények sosem tűntek el, csak nem MI-ként gontolunk rájuk (*ld. korában*)
- utolsó hullám eleje:
  - 2000-2001. – felismerték, hogy hétköznapi tudásra, „*józan észt*” igénylő feladatok megoldására, trivialitások felismerésére nem alkalmas
  - gondolat: „*ha meggondoljuk, a csecsemő és a gyerek is ezeket a trivialitásokat tanulja meg, amiig fel nem nő*”
  - projekt: Open Mind Common Sense
    - \* az általános tudásunknak, a *józan észnek* az adatbázisa
    - \* ilyeneket tartalmazott, pl. *az anyám anyja a nagyanyám*
  - minek összegyűjteni ezeket, ha ott az **internet**?



ezt felhasználva kapjuk meg a **gépi tanulást** → elérkeztünk napjainkig

## 2. Napjaink MI-je – Optimalizáció

Az előadás téTELmondata: az MI-kutatás (legtöbb) feladata az **optimalizáció**ról szól.

### 2.1. Mitől optimalizáció?

- példa:  $f^* = \arg \max_{x \in X} f(x)$
- feladat: keressük azt az modellt, aminek a paramétereit (**súlyait**, jele  $w \in W$ ) **a legjobban állítom be**, a legjobban teljesít általában minimumot keresünk
- jó = minimumtól való eltérés kicsi
- $f^* = \arg \min_{w \in W} f(w, x)$  (ahol  $w$  a tanult paraméterek és  $x$  a bemeneti példányok)
- valahol könnyebben látszik, valahol nehezebben látszik az optimalizáció (problémamegoldás) pl. legrövidebb  $A$ -ból  $B$ -be vezető út (lehetséges jelentései: legrövidebb számú lépéssorozat, legrövidebb úthossz, legrövidebb idő)

### 2.2. Intelligens viselkedés

Tipikus feladatok, amiket ha képes megoldani, intelligensként tekintünk rá.

#### A) Tervezés

- adott bizonyos lépések elvégzésének egy sorrendje → cél:
  - hogy a legtöbbet gyártsuk
  - hogy a legnagyobb legyen a haszon
- ha nem triviálisan kicsi a feladat, nem várhatjuk el, hogy az MI jobban teljesítsen

#### B) Kép- / hang- / szövegfelismerés

- pl. ismerje fel, hogy milyen betűt ábrázol a rajz
- megadunk egy **veszteségfüggvényt** / **hibafüggvényt** → cél: az ettől való eltérés minimalizálása

#### C) Tanulás

- kulcs: **fejlődés** ( minden egyes iterációnál váljon jobbá a korábbi iterációhoz képest )
- pl. *darts célbabotása*: ha nagyon magasan van, vigye lejjebb, ha túl balra dob, jobbrább kell tolnom  $\Rightarrow$  **paraméterek finomhangolása**
- találja el a célpontot → vizsgálja meg az eredményt → változtasson a paramétereken (váljon jobbá)
- szintén, a célunk
  - hiba minimalizálása
  - pontosság maximalizálása
- feladat: határozzuk meg, hogy a *hibafüggvény az adott helyen hogyan fog változni*

↓

**gradiens, deriválás**

- gépi tanulás osztályai

(a) Felügyelt tanulás (*supervised learning*)

- tudjuk a jó választ
- az MI eredményét összehasonlíthatjuk vele (jó-e vagy sem, mennyire jó, stb.)
- *előfeltétel*: álljon a rendelkezésünkre sok példa / adat, amik meg vannak címkézve (**címkézett adat**) → ez a súlyállítgatás miatt fontos

(b) Felügyelet nélküli tanulás (*unsupervised learning*)

- címke nélküli adatunk van
- „*majd meglátjuk*” alapon derül ki a képessége
- pl. a '70-es évek óra így működtek a karakterfelismerő rendszerek → **klaszterezéssel** megállapítja, mennyire hasonlítanak
- van, hogy mi adjuk meg, hány csoportba rendezze a bemeneteket (pl. számjegyek esetén 10-et), VAGY rábízzuk, hogy saját maga fedezze fel ezeket a csoportokat  
*lehetséges probléma*: a 9-es és a 6-os számjegyet azonosnak veszi

(c) Megerősítéses tanulás (*reinforcement learning*)

- régi ötletről van szó
- próbálja meghaladni a felügyeletet
- nincs címke, és adat sem igazán → helyette **visszacsatolás** van
- ha a visszacsatolás szerint helyes az eredmény, **jutalomban** részesül (*reward*)
- „*próba-szerencse*” (*trial and error*) alapjú
- példa:  $\alpha_0$  sakkjáték

- *megjegyzés*: a tanulás **lassú folyamat**

- felügyelt és megerősített tan.: sok számítás és futtatás
- felügyeletlen tan.: valamivel kevesebb
- ha meggondoljuk, az emberek is lassan tanulnak

D) Valami új generálása

- a tanulás alkalmazása a szoftver generatív aspektusában (Dall-E, ChatGPT, stb.)
- az alábbi gondolatok álnak mögötte

(a) Önkódoló rendszerek (*autoencoders*)

- az alábbiakra van szükségünk:

\* adott egy bemenet és egy kimenet:  $in, out \in \mathbb{R}^D$

\* kódolós (*encoding*):  $f : \mathbb{R}^D \rightarrow \mathbb{R}^d$ , ahol  $d \ll D$ <sup>1</sup>

\* dekódolás (*decoding*):  $g : \mathbb{R}^d \rightarrow \mathbb{R}^D$

\* generatív kimenet (generative output):  $g(y)$ , ahol  $y \in \mathbb{R}^d$  tetszőleges (random)

---

<sup>1</sup>Jelentése: sokkal kisebb a dimenzionalitása  $d$ -nek  $D$ -hez képest

- az *ötlet mögötte*: van egy óriási dimenzionalitású adatunk → a kódoló  $f$  függvénytel lecsökkenjük a dimenzionalitását → az így kapott eredményt dekódoljuk a  $g$ -vel, hogy a bemenetivel azonos dimenzionalitású eredményt kapjunk
- *cél*: úgy kódoljuk át a bemenetet, hogy az eredeti bemenet „jellegzetességeit” megőrizze
- olyan, *mint* a JPEG tömörítő algoritmus; hogy az ember számára nem látható részleteket eliminálja a képből, ezzel csökkentve a fájlmeretet
- *eredmény*: minél jobbak a leképezőfüggvényeink, annál jobban fog hasonlítani a vég-eredmény az eredeti bemenetre
- pl. arc generálása, javító algoritmusok

(b) *GAN – generative adversarial network*

- lehetséges magyar fordítása: generatív ellenfél hálózat
- példán keresztül szemléletve: adott két MI
  - \* az egyiknek (*A* vagy **generátor**) a feladata: tanulja meg, hogyan kell bankjegyet hamisítani
  - \* a másiknak (*B* vagy **diszkriminátor**) a feladata: tanulja meg felismerni a hamis bankjegyeket
  - \* a két modell interakcióban van egymással
  - \* kezdetben az *A* szörnyen teljesít, így a *B*-nek könnyű dolga van
  - \* ahogy *A* egyre fejlődik, úgy válik *B*-nek a feladata nehezebbé
  - \* ezt a módszert **generátor-diszkriminátor modellnek** is hívják

### 2.3. Mitől nehéz az optimalizáció?

- $f^* = \arg \min_{w \in W} (w, x)$  – jellemzően  $W \subseteq \mathbb{R}^N$ , ahol  $N > 10^6$
- azaz *hatalmas a dimenzionalitás*, amivel dolgoznunk kell
  - nem egyértelmű, melyik súlyt finomhangoljuk
- nem gyakran folytonos / deriválható / monoton → csúnya függvények
  - más szóval „rücskös” függvények (*rugged functions*)
  - nem alkalmazhatók a jól ismert tteleink analízisból → nincs esélyünk megtalálni az egzakt szélsőértékeit
  - ezért **közelítenünk kell felé**

### 2.4. Hogyan próbálkozhatunk?

- a biológiából, a természetből inspirálódunk → **matematikai modellek**
  - agy → **mesterséges neurális hálók** (*artificial neural networks*)
  - evolúció → **evolúciós algoritmusok** (*evolutionary algorithms*)
  - társas rovarok → **raj intelligencia** (*swarm intelligence methods*)
- róluk később bővebben lesz szó

### 3. Felügyelt tanulás

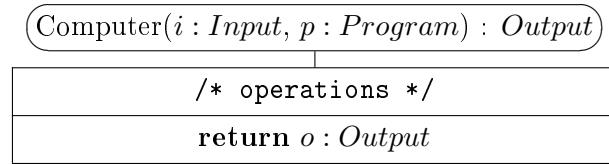
Más fordítások: ellenőrzött tanulás.

#### 3.1. Felügyelet nélküli tanulás – visszatekintés

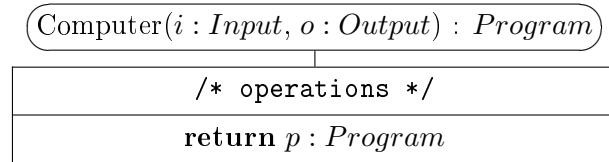
- nagy adathalmaz → megpróbál hasonlóságokat és különbségeket észrevenni → csoportosítás, klaszterezés
- nincs visszacsatolás a környezettől, nincs (számszerű) visszajelzés

#### 3.2. Felügyelt tanulás

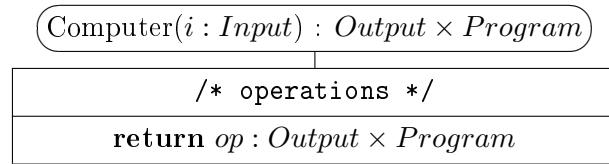
- van egy „tanító”, azaz az **adataink meg vannak címkézve**, ami egyfajta vissza jelzésül szolgál
- pl. **hagyományos számítógépes programozás**



- ehhez képest a **felügyelt tanulás**



- emberibb példa: „*mit csinál a gép*” feladattípus (2. osztályból) → a programra kell rájönni, ahol ismerjük a bemenetet és a kimenetet
- **felügyelet nélküli tanulás**: a visszacsatolás hiányzik (címkeinformáció) → sokkal nehezebbnek tűnik



#### 3.3. A tanulás típusai

- **tanulás**: saját teljesítményünk fejlesztése megfigyelések alapján Machine Learning
- **gépi tanulás**: az MI „részhalma”, a számítógépes rendszerek a megfigyelendő adatokban található mintákból tanul
- tanulási feladatok
  - **klasszifikáció** (osztályozás): ahol egy véges halmaz a kimenet
  - **regressziós** :  $\mathbb{R}$  ahol a kimenet egy numerikus predikció, (numerikus) feladat
- (2. osztályos példa) hogyan jövünk rá a programra / az algoritmusra? → mintát fedezünk fel a bemenet és kimenet között ÉS / VAGY próbálkozunk (és ezt optimalizáljuk)
- a hiba segítségével lehet irányítani a tanítást → modell (függvény)  $\Rightarrow$  távolságfüggvény

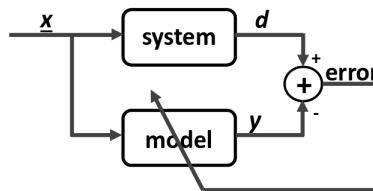
### 3.4. Mi az a felügyelt tanulás?

Adott egy  $N$  darabból álló **tanító adathalmaz** (*training data set*), mely

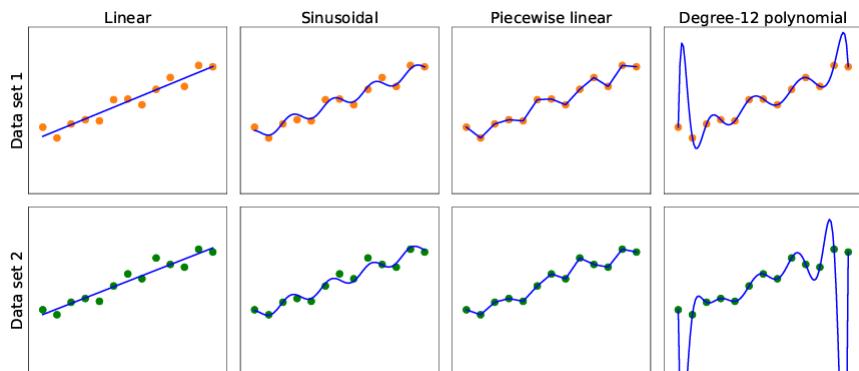
$$(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$$

bemeneti-kimeneti párokból áll, melyeket egy ismeretlen  $y = f(x)$  függvény generált. A feladat, hogy fedezzük fel a  $h$  függvényt (**hipotézist** – modellt), ami az igazi  $f$  függvényt közelíti.

- adottak bement-kimenet párosak:  $x_1^{(p)}, \dots, x_n^{(p)}, d^{(p)}$
- $p$ : a minták száma,  $n$ -dimenziós bemenet, skaláris kimenet
- $x$  vektor → rendszeren és modellen keresztül →  $d$  és  $y$



- a hiba vezérli a tanulást, és a hibát onnan tudjuk, hogy ismerjük a helyes kimenetet
- sokféle tanítási módszer létezik
- szemléletes példa:  $(x, y)$  pontpárokat adunk meg → találunk egy függvényt, ami átmegy ezeken a pontokon



- mi a baj ezzel a megközelítéssel? → túl specifikus az eredmény, más bemenet esetén nem lesz jó az eredményünk
- adathalmaz / tanítóminta – ne csak egy adott adatszettre illeszkedjen rá → **legyen jó az általánosítóképessége**

### 3.5. Problémák

- alulilleszkedés: túl egyszerű a modell, így nem jó az általánosító képessége
- túlilleszkedés: túl specifikus, szintén nem kezeli jól a tesztadatot

### 3.6. Döntési fák

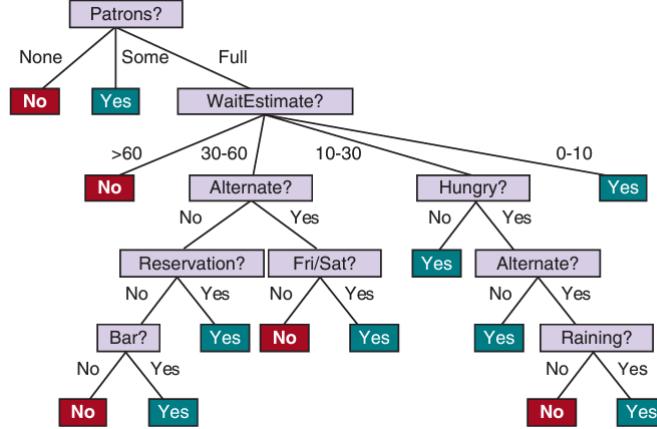
- fák egyes csúcsaiban egyes helyzetek vannak, ahol az élek egyes döntéseket jelentenek
- egy logikai döntési fára gondolhatunk így is:

$$\text{Output} \iff (Path_1 \vee Path_2 \vee \dots)$$

ahol az egyes *Pathok* (fagráfon utak) a gyökérből az egyes levelekbe vezető utakat jelölik (tehát, hogy igaz vagy hamis döntésre jutottunk az egyes lépésekkel követően)

- pl. éttermi várakozási probléma

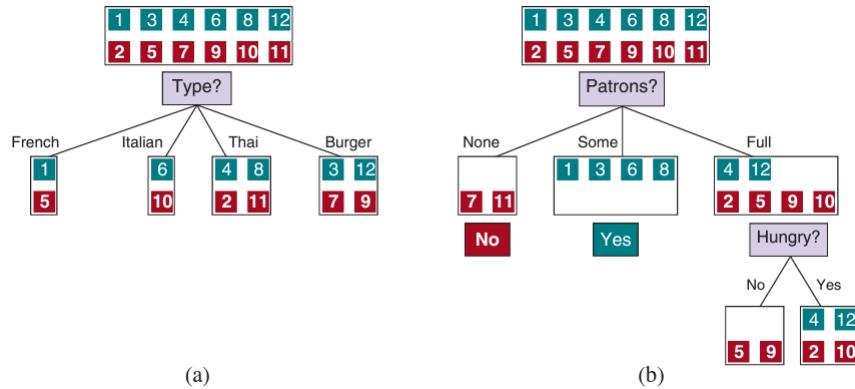
- rengeteg szempont felsorolva, mindenekhez tartozhat néhány eset
- a szerzők felállítottak egy döntési fát erre a szituációra



- mit csináljon, ha vannak információink nemcsak az inputról, hanem az outputról?
- az attribútumok alapján hogyan tudunk felépíteni egy fát?

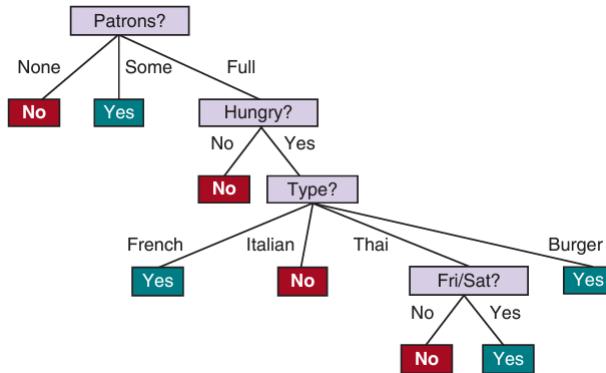
Example	Input Attributes										Output
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	
X <sub>1</sub>	Yes	No	No	Yes	Some	\$\$\$	No	Yes	French	0-10	y <sub>1</sub> = Yes
X <sub>2</sub>	Yes	No	No	Yes	Full	\$	No	No	Thai	30-60	y <sub>2</sub> = No
X <sub>3</sub>	No	Yes	No	No	Some	\$	No	No	Burger	0-10	y <sub>3</sub> = Yes
X <sub>4</sub>	Yes	No	Yes	Yes	Full	\$	Yes	No	Thai	10-30	y <sub>4</sub> = Yes
X <sub>5</sub>	Yes	No	Yes	No	Full	\$\$\$	No	Yes	French	>60	y <sub>5</sub> = No
X <sub>6</sub>	No	Yes	No	Yes	Some	\$\$	Yes	Yes	Italian	0-10	y <sub>6</sub> = Yes
X <sub>7</sub>	No	Yes	No	No	None	\$	Yes	No	Burger	0-10	y <sub>7</sub> = No
X <sub>8</sub>	No	No	No	Yes	Some	\$\$	Yes	Yes	Thai	0-10	y <sub>8</sub> = Yes
X <sub>9</sub>	No	Yes	Yes	No	Full	\$	Yes	No	Burger	>60	y <sub>9</sub> = No
X <sub>10</sub>	Yes	Yes	Yes	Yes	Full	\$\$\$	No	Yes	Italian	10-30	y <sub>10</sub> = No
X <sub>11</sub>	No	No	No	No	None	\$	No	No	Thai	0-10	y <sub>11</sub> = No
X <sub>12</sub>	Yes	Yes	Yes	Yes	Full	\$	No	No	Burger	30-60	y <sub>12</sub> = Yes

- eddig csak kézzel építettük fel, most automatizáljuk a folyamatot
- precedenciát állítunk fel az attribútumok alapján; azaz lesznek fontosabb attribútumok (szempontok), amik előnyt élveznek a többivel szemben



- mikor fontosabb egy attribútum, ha jobban szeparálja az eseteket

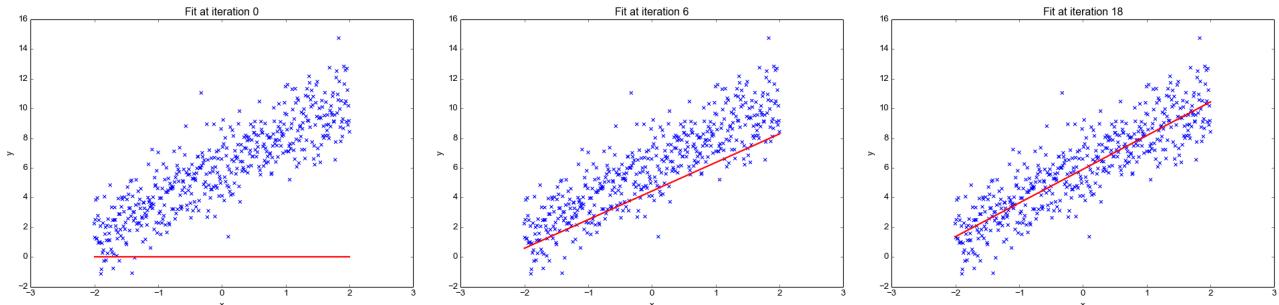
- ezt az algoritmust rekurzíve meghívjuk és így építjük fel a fát → optimális lesz az bemeneti adathalmaz szempontjából, megkapjuk, hogy optimálisan hogy jutunk el a legrövidebb úton a fa gyökeréből a csúcsba



- előnyei
  - könnyű megérteni
  - hatalmas adathalmazra is jól kiterjeszthető (scalability)
  - rugalmasan kezeli a véges és folytonos adatokat
  - klasszifikáció és regresszió egyszerre
- hátrányai
  - szuboptimális pontosság (főleg a mohó keresés miatt)
  - ha a fa mély, egy új előrejelzés nagyon költséges lehet egy új példánál
  - a döntési fák instabilak – egyetlen új példa hozzáadása a teljes fát megváltoztathatja

### 3.7. Regresszió

- feladat: egy lineáris függvényt hogyan illeszthetünk rá egy adathalmazra



## 4. Neurális hálózatok

### 4.1. Biológiai és mesterséges neuronok

#### 4.1.1. Biológiai neuronok

- a biológiából jött a motiváció – ha ott működik, miért ne működne gépeknél is?
- **emberi idegrendszer:** egy nagyon összetett rendszer
  - *feladatai:* emlékezés, gondolkozás, problémamegoldás, döntéshozatal, stb.
  - *neuron:* jeleket kap meg más neuronuktól és ezeket kombinálja
    - \* egy baba az agyában  $10^{11}$  db. neuronnal születik
    - \* egy neuron kb. 1000 neuronhoz van hozzácsatlakoztatva  $\rightarrow 10^{14}$  db. kapcsolat (vagy él, ha gráf osztóból közelítjük meg)
    - \* az emberi memóriakapacitás: 1 és 1000 TB között
  - ezek a jelek a *dendritek* mentén haladnak
  - ha a jel elég erős  $\rightarrow$  kimeneti jel az *axonon* keresztül más neuronokhoz
- felépítése
  - neuronok (idegsejtek) –  $10^{11}$  darab van belőlük
  - átlagosan 100 összeköttetés per idegsejt  $\rightarrow$  100 millió összesen
  - sejttest  $\rightarrow$  információ  $\rightarrow$  axon

#### 4.1.2. Mesterséges neuronok

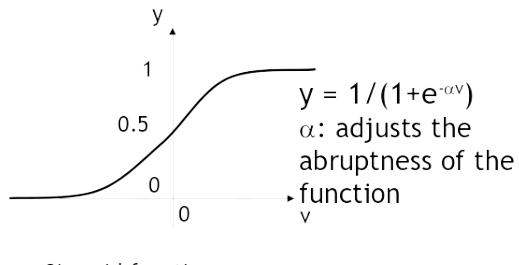
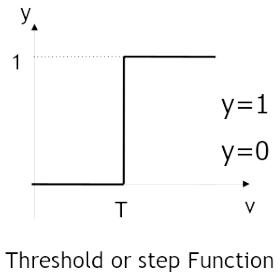
- az emberi agy és a számítógép összehasonlítása

	Számítógép	Emberi agy
<i>Sebesség</i>	4 GHz felett	40-50 Hz
<i>Működési mód</i>	sorosan (lineárisan)	párhuzamosan
<i>Ember felismerése</i>	bonyolult	könnyű <sup>2</sup>

- az idegrendszernek egy egyszerűsített modelljét használjuk fel az MI-ben is
- **mesterséges neuron:** a bemenetek  $n$ -dimenziós *vektorok*  $((x_1, \dots, x_n)$ , ezek a), melyek *súlyokkal* rendelkeznek  $((w_1, \dots, w_n))$ , melyeket egy  $v$  értékben „összeadó csomópontban” (*summing junction*) számszerűsítünk. Ezeket átadjuk egy  $\Phi$  függvénynek, ami visszaalakítja olyan formátumúvá, hogy bemenetként fel tudják használni más neuronok

$$((x_1, w_1), \dots, (x_n, w_n)) =: \mathbf{v} \mapsto \Phi(\mathbf{v}) := (y_1, \dots, y_n)$$

- önmagukban csak egyszerű feladatok megoldására képesek
- viszont ha elég sokat használunk fel belőlük, melyek össze vannak kötve egymással, képesek komplex feladatokat is megoldani
- **$\Phi$  – aktivációs függvény:** a kimenetet két aszimptota között korlátozza le, hogy a neuronok egy ésszerű, dinamikus intervallumon belül legyenek értelmezve  
Tipikusan **lépcsős** (*step function*) vagy **szigmoid függvény** (*sigmoid function*) szokott lenni.



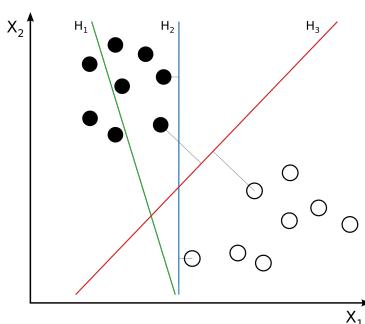
1. ábra. Lépcsős és szigmoid függvény

## 4.2. Egyrétegű perceptronok

- Rosenblatt vezette be 1958-ban, ez volt az első modell felügyelt tanulásra
- a McCulloch–Pitts-modellen alapult
- feladat: lineárisan szétválasztja (szeparálja)  $\Rightarrow$  **osztályozza az adathalmazt**
- előfeltétel: az adat legyen **lineárisan szeparálható**
- ezen limitáció miatt elapadt a lelkesedés iránta, de a '80-as években ismét elkezdtek érdeklődni a kutatók (napjainkban is fontosak)
- ennek továbbfejlesztett változata a *többrétegű perceptronok*, melyek lineárisan nem szeparálható mintákat is képesek csoportosítani

### 4.2.1. Lineáris szeparabilitás

- Matematikai jelentése: szerkesszünk egy hipersíket, amely kettéválasztja az adatokat úgy, hogy egy adat pontosan egy csoportba tartozhat (függvény grafikonja felett vagy alatt)
- Megjegyzés: kétdimenziós adatok esetén egyenest,  $n$ -dimenziós adathalmaz esetén  $(n-1)$ -dimenziós hipersík lesz a szeparátorunk



2. ábra. A  $H_1$  hipersík nem választja szét az adathalmazt, míg a  $H_2$  és  $H_3$  igen.

- Példa: vegyük két bemenetet:  $(x_1, x_2)$ 
  - Megadhatunk-e rá egy szétválasztó egyenest?

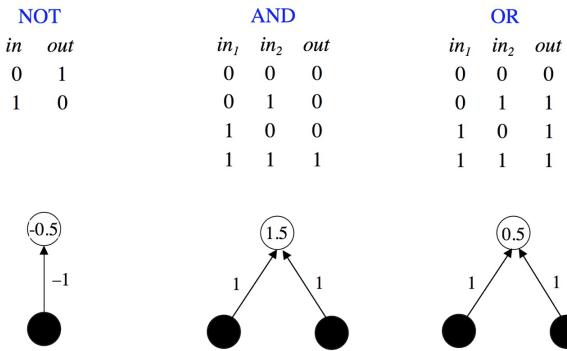
$$-w_0 + x_1 w_1 + x_2 w_2 = 0$$

- A perceptronok paramétereit **súlyoknak** (*weights*) hívjuk:  $w_1, w_2, \dots$
- Mi az egyenlete?

$$x_2 = -\frac{w_1}{w_2}x_1 + \frac{w_0}{w_2}$$

#### 4.2.2. Logikai kapuk

- Tipikusan kétváltozós, McCulloch-Pitts perceptronokkal megoldható feladat a logikai kapuk implementációja
- A **logikai függvények** (*boolean functions*), mint a  $\wedge$  (AND),  $\vee$  (OR) és a  $\neg$  (NOT) jól szeparálhatók lineárisan.<sup>3</sup>
- Mindegyik függvényre vannak bemeneteink:  $in_1$  és  $in_2$ , valamint egy  $out$  kimenet, melyekhez meg kell határozunk a súlyokat és a küszöbértéket



3. ábra. A NOT, AND és OR függvények implementációja

- Vannak azonban olyan függvények, amelyekhez kézzel nem szerkeszthetünk ilyen hálózatot. Ilyen például az XOR



4. ábra. Az XOR-hoz nem tudunk megfelelő súlyokat megadni

- Mindegyik **tanítóminta** (*training pattern*) lineáris egyenlőtlenségeket állít elő a kimenet számára, melyek a hálózat bemeneteiből és a hálózat paramétereiből állnak  $\rightarrow$  ebből kiszámíthatjuk a súlyokat és a küszöbértéket
- A következő egyenlőséget kell kielégítenie:

$$out = \text{sgn}(w_1 \cdot in_1 + w_2 \cdot in_2 - \theta).$$

$in_1$ $in_2$ $out$ 0     0     0 0     1     0 1     0     0 1     1     1	$w_1 0 + w_2 0 - \theta < 0$ $w_1 0 + w_2 1 - \theta < 0$ $w_1 1 + w_2 0 - \theta < 0$ $w_1 1 + w_2 1 - \theta \geq 0$	$\Rightarrow$ $\Rightarrow$ $\theta > 0$ $w_2 < \theta$ $w_1 < \theta$ $w_1 + w_2 \geq \theta$
---	---	---

5. ábra. Az AND-hez tartozó egyenlőtlenségrendszer

---

<sup>3</sup>**Megjegyzés.** Összesen  $2^4 = 16$  darab logikai függvény van, viszont a felsorolt 3-ból ( $\wedge$ ,  $\vee$  és  $\neg$ ) kirakhatjuk az összes többi is.

- Láthatjuk, hogy végtelen sok megoldásunk lehet. Hasonló igaz ez a NOT-ra és az OR-ra is.
- Ha a fenti egyenlőtlenséget elvégezzük az XOR-ra  $\rightarrow$  a 2. és a 3. ellent mond a 4. egyenlőtlenséggel, így nincs megoldása a feladatnak (hátránya a perceptronoknak)

$$\begin{array}{c|cc|c}
 & in_1 & in_2 & out \\
 \hline
 & 0 & 0 & 0 \\
 & 0 & 1 & 1 \\
 & 1 & 0 & 1 \\
 & 1 & 1 & 0
 \end{array} \Rightarrow \boxed{\begin{array}{l} w_1 0 + w_2 0 - \theta < 0 \\ w_1 0 + w_2 1 - \theta \geq 0 \\ w_1 1 + w_2 0 - \theta \geq 0 \\ w_1 1 + w_2 1 - \theta < 0 \end{array}} \Rightarrow \boxed{\begin{array}{l} \theta > 0 \\ w_2 \geq \theta \\ w_1 \geq \theta \\ w_1 + w_2 < \theta \end{array}}$$

6. ábra. Az AND-hez tartozó egyenlőtlenségrendszer

- komplexebb hálózatokra lenne szükségünk, vagyis olyanokra, melyek több kisebb hálózatot kombinálnak; vagy egy másik aktivációs függvényt kellene használnunk
- akárhogy is, bonyolultabbá válik a küszöbérték és a súlyok meghatározása papíron

### 4.3. Perceptronok tanítása

- angolul: **training of perceptrons** (itt: *tanítás*, esetleg *kiképzés*)

A perceptronok tanításának algoritmusai.

#### I.) Inicializáció

- állítsuk be a kezdeti értékeit a súlyoknak:  $w_1, w_2, \dots, w_m$
- állítsunk be egy  $\theta$  **küszöbértéket** egy random számra  $a(z) \theta \in \left[-\frac{1}{2}, \frac{1}{2}\right]$  intervallumból
- állítsuk be a  $\eta$  **tanulási rátát** egy pozitív számra úgy, hogy  $\eta < 1$

#### II.) Aktiváció

- számítsuk ki a  $p$ -edik iteráció **tényleges kimenetét**

$$y = \Phi \left( \sum_{i=1}^m x_i \cdot w_i \right)$$

- aktivációs függvény: **küszöbérték / lépésfüggvény** (vagy előjelfüggvény)

$$y(p) = step \left[ \left( \sum_{i=1}^m x_i \cdot w_i \right) - \theta \right]$$

#### III.) Súlyok tanítása:

a perceptronok súlyainak frissítése

- új súlyok:

$$w_i(p+1) = w_i(p) + \Delta w_i(p)$$

- súlykorrekció:

$$\Delta w_i(p) = \eta \cdot x_i(p) \cdot e(p)$$

- hiba (*error*):

$$e(p) = d(p) - y(p)$$

#### IV.) Iteráció:

inkrementáljuk egyesével a  $p$ -t, térjünk vissza a II. lépéshöz és ismételjük a folyamatot a konvergenciáig

Példa az algoritmusra: perceptronok tanítása a logikai „és” műveletére.

Az epochnak 4 lehetséges bemeneti mintája van:

$$00, 01, 10, 11.$$

Kezdeti súlyok:

$$w_1 := 0,3 \text{ és } w_2 := -0,1.$$

Küszöbérték:  $\theta := 0,2$ . Tanulási ráta:  $\eta := 0,1$ .

Epoch	Ite-ráció	Bemenetek		Kezdeti súlyok		Elvárt kimenet	Tényleges kimenetek	Hiba	Végleges súlyok	
	$p$	$x_1(p)$	$x_2(p)$	$w_1(p)$	$w_2(p)$	$d(p)$	$y(p)$	$e(p)$	$w_1(p+1)$	$w_2(p+1)$
1	1	0	0	0,3	-0,1					
	2	0	1							
	3	1	0							
	4	1	1							

7. ábra. 1. lépés – Initializáció

Tényleges kimenetek:

$$y(p) = y(1) = \text{step} \left[ \left( \sum_{i=1}^m x_i \cdot w_i \right) - \theta \right] = \text{step} [(0 \cdot (0,3) + 0 \cdot (-0,1)) - 0,2] = \text{step} (-0,2) = 0$$

Epoch	Ite-ráció	Bemenetek		Kezdeti súlyok		Elvárt kimenet	Tényleges kimenetek	Hiba	Végleges súlyok	
	$p$	$x_1(p)$	$x_2(p)$	$w_1(p)$	$w_2(p)$	$d(p)$	$y(p)$	$e(p)$	$w_1(p+1)$	$w_2(p+1)$
1	1	0	0	0,3	-0,1		0			
	2	0	1							
	3	1	0							
	4	1	1							

8. ábra. 2. lépés – Aktiváció

Az „és” műveletre a 00 bemenetekre 0 lesz az eredmény.

Hiba:

$$e(p) = e(1) = d(1) - y(1) = 0 - 0 = 0.$$

Epoch	Ite-ráció	Bemenetek		Kezdeti súlyok		Elvárt kimenet	Tényleges kimenetek	Hiba	Végleges súlyok	
	$p$	$x_1(p)$	$x_2(p)$	$w_1(p)$	$w_2(p)$	$d(p)$	$y(p)$	$e(p)$	$w_1(p+1)$	$w_2(p+1)$
1	1	0	0	0,3	-0,1	0	0	0		
	2	0	1							
	3	1	0							
	4	1	1							

9. ábra. 3. lépés – Súlyok tanítása (1/2)

**Megjegyzés.** Az „epoch” kifejezés szó szerint *kort*, *korszakot* jelent. Szinonimaként a GT témakörében még generációnak (*generation*) is nevezik, leginkább az evolúciós algoritmusok esetében. Arra utal, hogy „hányadik generációs fejlettségben” van az adathalmaz. Nem vagyok benne biztos, hogyan szokták lefordítani a magyar szakirodalomban, így inkább az angol megfelelőjét választottam.

Az új súlyok kiszámítása.

$$\begin{aligned}\Delta w_1(p) &= \Delta w_1(1) = \eta \cdot x_1(1) \cdot e(1) = 0, 1 \cdot 0 \cdot 0 = 0 \\ \Delta w_1(p+1) &= w_1(1) + \Delta w_1(1) = 0, 3 + 0 = 0, 3 \\ \Delta w_2(p) &= \Delta w_2(1) = \eta \cdot x_2(1) \cdot e(1) = 0, 1 \cdot 0 \cdot 0 = 0 \\ \Delta w_2(p+1) &= w_2(1) + \Delta w_2(1) = (-0, 1) + 0 = -0, 1\end{aligned}$$

Epoch	Iteráció	Bemenetek		Kezdeti súlyok		Elvárt kimenet	Tényleges kimenetek	Hiba	Végeleges súlyok	
	$p$	$x_1(p)$	$x_2(p)$	$w_1(p)$	$w_2(p)$	$d(p)$	$y(p)$	$e(p)$	$w_1(p+1)$	$w_2(p+1)$
1	1	0	0	0, 3	-0, 1	0	0	0	<b>0, 3</b>	-0, 1
	2	0	1							
	3	1	0							
	4	1	1							

10. ábra. 3. lépés – Súlyok tanítása (2/2)

Innentől  $p := p + 1 = 2$ . Ismételjük a 2-3-4. lépésekét, ameddig el nem kezd konvergálni.

Epoch	Iteráció	Bemenetek		Kezdeti súlyok		Elvárt kimenet	Tényleges kimenetek	Hiba	Végeleges súlyok	
	$p$	$x_1(p)$	$x_2(p)$	$w_1(p)$	$w_2(p)$	$d(p)$	$y(p)$	$e(p)$	$w_1(p+1)$	$w_2(p+1)$
1	1	0	0	0, 3	-0, 1	0	0	0	0, 3	-0, 1
	2	0	1	<b>0, 3</b>	<b>-0, 1</b>					
	3	1	0							
	4	1	1							

11. ábra. 4. lépés – Iteráció

Epoch	Iteráció	Bemenetek		Kezdeti súlyok		Elvárt kimenet	Tényleges kimenetek	Hiba	Végeleges súlyok	
	$p$	$x_1(p)$	$x_2(p)$	$w_1(p)$	$w_2(p)$	$d(p)$	$y(p)$	$e(p)$	$w_1(p+1)$	$w_2(p+1)$
1	1	0	0	0, 3	-0, 1	0	0	0	0, 3	-0, 1
	2	0	1	0, 3	-0, 1	0	0	0	0, 3	-0, 1
	3	1	0	0, 3	-0, 1	0	1	-1	0, 2	-0, 1
	4	1	1	0, 2	-0, 1	1	0	1	0, 3	0, 0

12. ábra. 1. epoch végeredménye

Epoch	Itéráció	Bemenetek		Kezdeti súlyok		Elvárt kimenet	Tényleges kimenetek	Hiba	Végeleges súlyok	
	$p$	$x_1(p)$	$x_2(p)$	$w_1(p)$	$w_2(p)$	$d(p)$	$y(p)$	$e(p)$	$w_1(p+1)$	$w_2(p+1)$
2	5	0	0	0,3	0,0	0	0	0	0,3	0,0
	6	0	1	0,3	0,0	0	0	0	0,3	0,0
	7	1	0	0,3	0,0	0	1	-1	0,2	0,0
	8	1	1	0,2	0,0	1	1	0	0,2	0,0

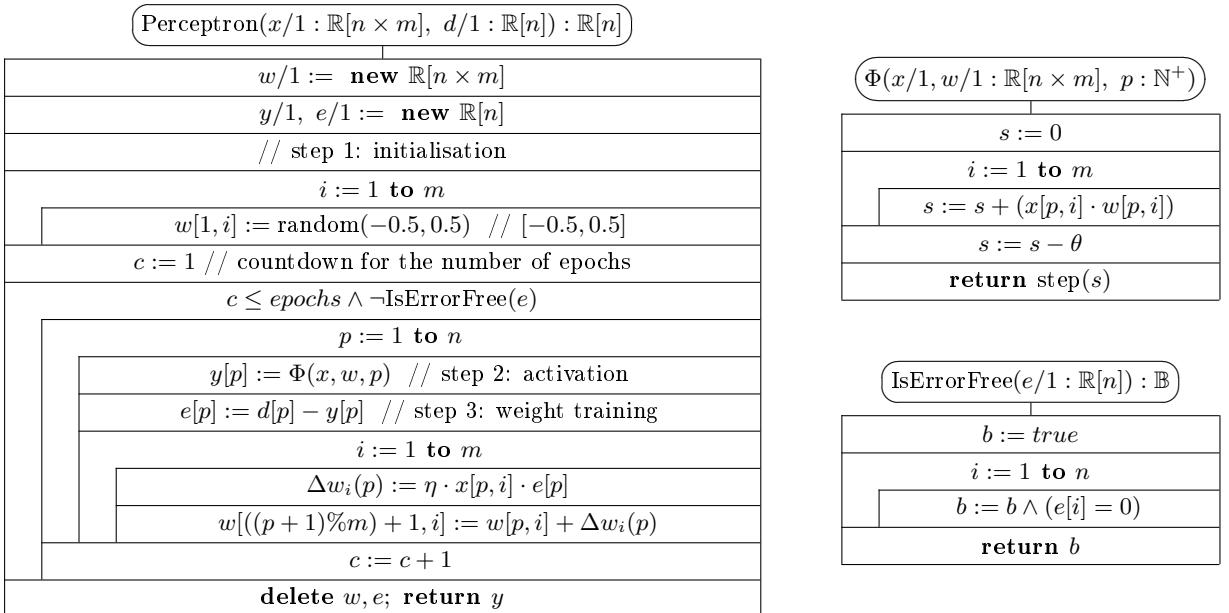
13. ábra. 2. epoch végeredménye

Epoch	Itéráció	Bemenetek		Kezdeti súlyok		Elvárt kimenet	Tényleges kimenetek	Hiba	Végeleges súlyok	
	$p$	$x_1(p)$	$x_2(p)$	$w_1(p)$	$w_2(p)$	$d(p)$	$y(p)$	$e(p)$	$w_1(p+1)$	$w_2(p+1)$
5	17	0	0	0,1	0,1	0	0	0	0,1	0,1
	18	0	1	0,1	0,1	0	0	0	0,1	0,1
	19	1	0	0,1	0,1	0	0	0	0,1	0,1
	20	1	1	0,1	0,1	1	1	0	0,1	0,1

14. ábra. 5. epoch végeredménye

- Ilyen feladat szerepelhet a vizsgán!
- Bebizonyították, hogy ezzel az algoritmussal bármely kétosztályosan szeparálható feladatra adható egy szétválasztó hipersík (ami két osztály esetén egy egyenes).
- A konvergencia gyorsasága sok mindenről függ: a sorrendtől, a küszöbértékektől (egyes  $\theta$  értékekre jobban konvergál), stb.
- A back-propagation algoritmussal több perceptronról összerakhatnak hálózatba

A következő struktogramm csak vázlatosan szemlélteti az algoritmust. Feltételezzük, hogy az egyes paraméterek, mint az  $epochs : \mathbb{N}^+$ ,  $\theta : [-0.5, 0.5]$  és  $\eta : \mathbb{R}$  változók globálisak. A pontos implementációs részleteket (globális változók, helyes inicializáció, függvényparaméter-átadás érték, cím vagy referencia szerint, stb.) az Olvasóra bízzuk.



## 5. Etikai és jogi kérdések az MI területén

### 5.1. Az MI eredményei és nem kívánt következményei

- az MI létrejöttének elsődleges motivációja az emberek minden napjainak könnyebbé, kényelmesebbé tétele
  - technológiai áttörések (pl. navigációs és GPS, stb.)
  - monoton feladatok automatizációja → hatékonyseg növelése
  - kutatásfejlesztésben áttörések ennek köszönhetően
  - (meg csomó példát lehet mondani, hamar össze tud szedni ilyeneket az ember)
- Center for Humane Technology
  - alapítói: Tristan Harris, Aza Raskin
  - mindenketten az MI etikai kérdéseivel, következményeivel foglalkoznak (*ethicist*)
  - **az MI-dilemma** (*the AI dilemma*): a tudományág és a technológia fejlődésének mértéke beláthatatlan mértéket ölt, így a friss innovációk olyan következményeket vonhatnak maguk után, melyekre nincs felkészülve a társadalunk
    - \* az emberi kép- és hanggenerálás hamar megérkezett, az eredményük minősége gyakran megkülönböztethetetlen a valós nyersanyagtól → a hang és kép alapú bizonyítékok elavulttá, használhatatlanná váltak az igazságszolgáltatásban
    - \* széles körben elérhetővé vált a generatív MI, emiatt felhasználhatjuk saját szolgáltatásaink (pl. rúter meghekkelésére írunk programot), intézményeink kijátszására (pl. autoriter rezsimek manipulációja)
    - \* jogilag teljesen érintetlen, korlátozások nélküli területek, esetek
  - videók az MI-dilemmáról
    - \* <https://www.youtube.com/watch?v=xoVJKj81cNQ&t=203s>
    - \* [https://www.youtube.com/watch?v=cB0\\_-qKba14&t=1163s](https://www.youtube.com/watch?v=cB0_-qKba14&t=1163s)
- egyre többen használják fel képzőművészeti célokra az eszközt
  - MI-generálta zenei videó: <https://www.youtube.com/watch?v=uG8vItscFKc>

### 5.2. Az MI és a művészet viszonya – Théâtre d'opéra spatial

- 2022, Colorado State Fair: éves szépművészeti verseny
- Jason Michael Allen a fotomanipulációs kategóriában nevezett egy MI-generálta képpel
- a Midjourney platformával készült; az első MI-generálta kép, ami **díjat nyert**
- hatalmas közfelháborodás a sajtóban
- felmerül egy csomó kérdés
  - Művészettel számít-e az MI által generált anyag?
  - A szerzői jog kit illet meg? Az utasítás (*prompt*) kiadóját vagy a platformot szolgáltató céget?
  - El fogja venni az emberek munkáját?
  - Eszközöként vagy veszélyes vetélytársként tekintsünk rá? (→ az érem két oldala)



15. ábra. Jason Michael Allen – Théâtre d'opéra spatial

### 5.3. Felmerülő gondok az adatbázisa kapcsán

- generatív mechanizmus: **generátor-diszkriminátor** modell → ha ugyanazzal az adathalmazzal dolgoznak, könnyen **beszennyezhetik egymás adathalmazát** → torz eredményt kaphatunk
- a GPT (*generative pre-trained transformer*) a teljes internetet felhasználja, de **nem jelöli meg a forrásait**, ami alapján generálta a tartalmat (pl. szöveget) → plagizálás, másolás?
  - a jegyzet írása idején (2023/2024/2. félév) egyes GPT modellek elkezdtek erre odafigyelni, pl. a Bing Copilot megjelöli a forrásait
  - kapcsolódó probléma: internetre kiengedve hamar rasszistává vált a ChatGPT → Ki a felelős érte? A cég? A termék? Kit lehet felelősségre vonni?

### 5.4. Korábbi találkozásaink az MI-vel

- a technológia 3 szabálya
  1. Amikor feltalálunk egy új technológiát, az új felelősségek osztályát fogja felfedni.  
(*When you invent a new technology, you uncover a new class of responsibilities.*)
  2. Ha a technológia hatalmat ruház át, az versenyt indít.  
(*If the tech confers power, it starts a race.*)
  3. Ha nem koordináljuk, a verseny tragédiába fulladhat.  
(*If you do not coordinate, the race ends in tragedy.*)
- első MI, amivel az ember találkozott: **sülik a weblapokon, social media** (adatgyűjtés, amit ellenünk használtunk fel) → profilozás megindult, csoportra szabott reklámok

↓

csomó új negatív káros hatás jelent meg, ami ránk is hatássak van  
(doomscrolling, polarizáció, stb.)

↓

adatalapú érdeklődés-fenntartás (*data-based engagement maximisation*)

- második találkozás az MI-vel:

- 2017-ig **generatív MI**; megszabott területeken lehetett hozzáérni (robotika, beszédfelismerés és -szintézis, zene- és képgenerálás, stb.)
- 2017-ben **transzformer modellek** megjelennek
- 2017 után; nyelvfeldolgozás, **generatív nagy multimodális nyelvi modellek** (*generative large language multimodal models*, GLLMM)

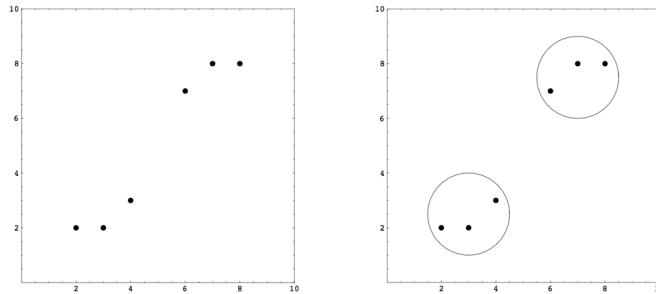
## 6. Felügyelet nélküli tanulás

### 6.1. Fő különbségek a felügyelt tanulástól

- **nincsenek címkék** (néha még az adat osztályai és jellemzői sem ismertek)
- **tipikus feladatok**: csoportosítások, hasonlóságok-különbözőségek felismerése (klaszterezés)
- **stratégiaiák**: **klaszterezés** (*clustering*) és **dimenziócsökkentés** (*dimensionality reduction*)

### 6.2. Klaszterezés

- **klasszifikáció** (felügyelt tanulás): esetén különböző típusú (címkekű) adathalmazunk van és szabályt állít fel, ami az adathoz hozzárendeli az osztályt
- **klaszterezés** (felügyelet nélküli tanulás): hasonló elemek milyen közel vannak egymáshoz, az adatok szerkezetét azonosítja
- **klaszter**: adatok egy csoportja, melyre teljesül, hogy
  - az azonos klaszterbe tartozó pontok jobban hasonlítanak egymásra, közelebb vannak egymáshoz
  - az eltérő klaszterekbe tartozók jelentősen eltérnek egymástól



16. ábra. Klaszterek

- **előnyei**: nincs szükségünk címkékre; néha nem is ismerjük az osztályokat vagy azok jellemzőit; csökkenthetjük vele az adatok számát
- **alkalmazásai**: kép- és jelfeldolgozás, adatelemzés, piacelemzés, tartalommenedzsment
- **elvárások, előfeltételek**
  - $n$  darab  $p$ -dimenziós adat ( $x_i \in X_1 \times \dots \times X_p$  (ahol  $i \in [1..n]$ ))
  - $k$  darab **nemüres klasztert** (*részalmazt* vagy *partíciót*) határozzunk meg, melyeknek uniója megadja az alaphalmazt és páronként diszjunktak

$$\begin{aligned} \forall i \in [1..k], C_i &\subseteq X : C_i \neq \emptyset \\ \forall i, j \in [1..k], i &\neq j : C_i \cap C_j = \emptyset \\ \bigcup_{i=1}^k C_i &= X \end{aligned}$$

- **típusai**: agglomeratív klaszterezés, prototípus alapú klaszterezés (pl.  $k$ -közepű klaszterezés), DBSCAN
- nincs univerzális megoldás, gyakran az adathalmaztól függ a megfelelő stratégia választása

### 6.2.1. Agglomeratív klaszterezés

- angolul: SAHN = *sequential agglomerative hierarchical non-overlapping clustering*
- algoritmus
  - $n$  darab adatot  $n$  különböző klaszterbe soroljuk
  - ezután iteratívan összevonogatjuk a két „legközelebb állókat” (ez lehet *minimum*, *maximum*, *átlag* távolság, stb.)
  - addig folytatjuk, ameddig egy klasztert nem kapunk
- előnyei
  - nem egy partíciót, hanem partíciók sorozatát hozzuk létre
  - nem kell előre meghatároznunk a klaszterek, partíciók számát
- hátránya: a legtöbb ilyen módszer
- az algoritmus matematikailag precíz leírása

1. lépés: adott  $\{x_1, \dots, x_n\} \subseteq R^p$  és  $d : R^p \rightarrow \mathbb{R}$  egy távolságmetrika
2. lépés: kezdetben:  $k_0 := n$ ,  $\Gamma_0 := \{C_{0,1}, \dots, C_{0,k_0}\} = \{\{x_1\}, \dots, \{x_n\}\}$
3. lépés:  $\forall i \in [0..(n-1)], \exists (a, b) \in \mathbb{N} \times \mathbb{N} : \min d(C_{i,a}, C_{i,b})$

$$\Gamma_{i+1} := \left( \Gamma_i \setminus \{C_{i,a}, C_{i,b}\} \right) \cup \{C_{i,a} \cup C_{i,b}\}$$

4. lépés: kimenet:  $\Gamma_0, \dots, \Gamma_{n-1}$

### 6.2.2. Prototípus alapú klaszterezés (*k-means clustering*)

- nem az összes adat tartozhat klaszterekbe
- helyette, mindegyik  $C_i$  klaszterhez hozzárendelünk egy  $v_i$  pontot az adattérben (ezt jellemzően a **klaszter középpontjának** hívjuk)
- az egyes adatpontok abba a klaszterbe tartoznak, melynek a középpontjához áll a legközelebb:

$$\|x_k - v_i\| = \min_{j=1}^K \|x_k - v_j\| \implies x_k \in C_i.$$

- a klaszterezés során egy **partíciómátrix** jön létre

$$U = \begin{pmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & \ddots & \vdots \\ u_{K1} & \cdots & u_{Kn} \end{pmatrix} \quad u_{ik} = \begin{cases} 1 & (x_k \in C_i) \\ 0 & (x_k \notin C_i) \end{cases}$$

- további megállapítások a partíciómátrix koordinátáiról
  - az adathalmaz összes pontja eleme valamely  $i$ -edik klaszternek:  $\sum_{i=1}^K u_{ik} = 1$
  - nincsen üres klaszter:  $\sum_{k=1}^n u_{ik} > 0$
- a prototípus alapú klaszterezések egyik leggyakoribb algoritmusa:

***k*-közpű klaszterezés (*k-means clustering*)**

- a neve onnan jön, hogy megmondjuk, hány  $K \in \mathbb{N}^+$  klaszterbe csoportosítsa a pontokat
- fontos, hogy jól határozzuk meg az optimális klaszterszámot
- hasonlóan iteratív folyamat, melyhez felhasználunk egy ún. **objektív függvényt**

$$J_{KM}(U, V) = \sum_{i=1}^K \sum_{x_k \in C_i} \|x_k - v_i\|^2 = \sum_{i=1}^K \sum_{k=1}^n u_{ik} \cdot \|x_k - v_i\|^2$$

↓

ezt a függvényt kell minimalizálnunk

- a súlypontok / középpontok kiszámítása:

$$v_i = \frac{1}{|C_i|} \cdot \sum_{x_k \in C_i} x_k = \frac{\sum_{k=1}^n u_{ik} \cdot x_k}{\sum_{k=1}^n u_{ik}}$$

- előnyök

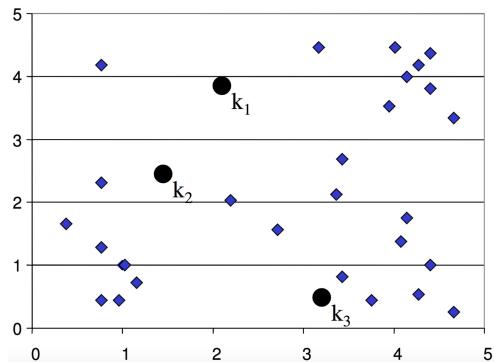
- egyszerű és könnyen implementálható
- intuitív objektív függvény
- relatíve hatékony:  $\mathcal{O}(t \cdot k \cdot n)$ 
  - $n$ : pontok száma
  - $k$ : klaszterek száma
  - $t$ : iterációk száma
  - jellemzően  $k, t \ll n$

- hátrányok

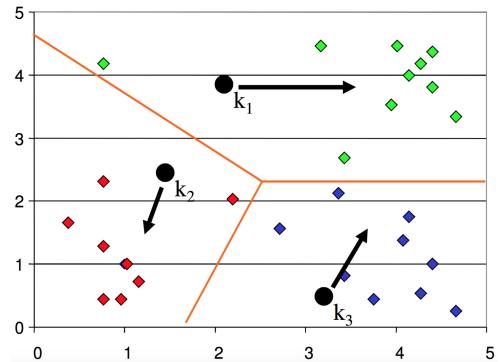
- alkalmazható, ha definiáljuk az átlagot
- muszáj meghatározni a  $K$  klaszterek számát
- vannak más távolságtechnikák, átlagszámítások
- sok esetben **lokális optimumban marad** (pl. nulla adatot tartalmazó klaszter)
- zajos és kirívó adatok kezelésére, valamint nem-konvex adatok felfedezésére sem alkalmas

- algoritmus

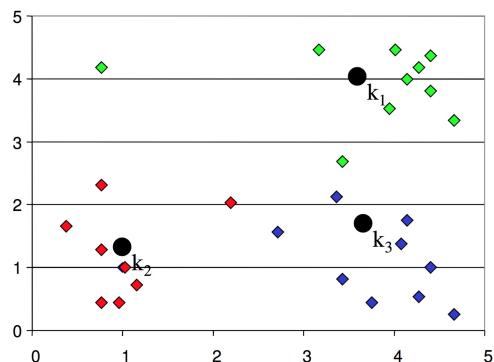
- léc: Határozzuk meg a klaszterek ( $K$ ) számát. Legyen az adathalmaz  $\{x_1, \dots, x_n\} \subseteq R^p$ . Legyen továbbá  $t_{max}$  a maximum iterációk száma,  $\|\cdot\|_v$  a klaszter középpontjától való távolság (norma),  $\varepsilon$  pedig a küszöbérték / tolerancia.
- léc: Inicializáljuk a klaszterek középpontjait random értékekkel.  $V^{(0)} \subseteq R^p$
- léc: Döntsük el az összes ( $N$ ) pontról, melyik osztályba tartozik (a legközelebbi középpontot hozzárendeljük).  $U^{(t)}(V^{(t-1)})$
- léc: Újraszámoljuk a  $K$  klaszter középpontjait úgy, hogy feltételezzük, hogy a pontok a megfelelő partícióhoz tartoznak.  
Ha  $\|V^{(t)} - V^{(t-1)}\|_v \leq \varepsilon \Rightarrow$  véget ér az eljárás.
- léc: A 3-4. lépést addig ismételjük, ameddig egyik pontnak sem változik meg a partícióba való besorolása.  
A végeredmény az  $U$  partíciómátrix és a  $V$  középpontok lesznek.



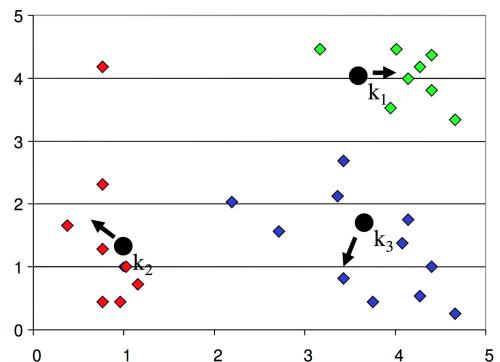
1. lépés: Inicializáció



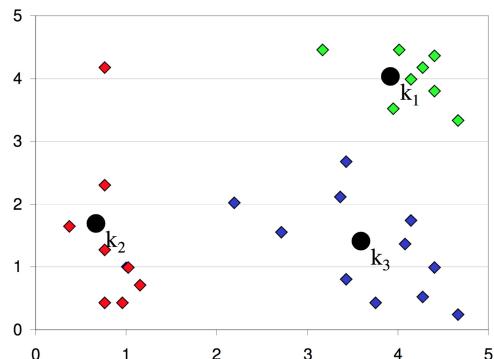
2. lépés: Pontok hozzárendelése klaszterekhez



3. lépés: Középpontok újraigazítása



2. lépés ismét



3-4. lépés (feltéve, hogy elértek a kívánt pontosságot)

### 6.2.3. DBSCAN

- angolul: *Density-Based Spatial Clustering of Applications with Noise*
- (nem volt róla részletesen szó)

### 6.3. Dimenziócsökkentés

- módszer: **főkomponens alapú analízis** (*principal component analysis*, PCA) – legfontosabb dimenziók megragadása
- statisztikus módszerrel az  **$n$ -dimenziós teret  $m$ -dimenziósra csökkentjük** ( $m < n$ )
  - azoktól szabadulunk meg, melyek a bemeneti tulajdonságokra nézve kis varianciájúak
- előnyei
  - kevesebb számítási idő, kevesebb adatot kell tárolni
  - redundanciát csökkenti
  - könnyű ábrázolni
- (nem volt róla szó részletesen az előadáson)

## 7. Megerősített tanulás

Angolul: *reinforcement learning* (RL)

### 7.1. Mi az a megerősített tanulás?

- ez áll a legközelebb ahhoz, ahogyan az emberek vagy állatok tanulnak
  - a kutya betanítása: ha ügyes volt, jutalomfalatot kap
  - csecsemők, babák tanulása a játszáson keresztül
- számítógépen: megközelítés arra, hogy interakciókból tanuljon a gép egy speciális **jutalomfüggvény** segítségével (*reward function*)

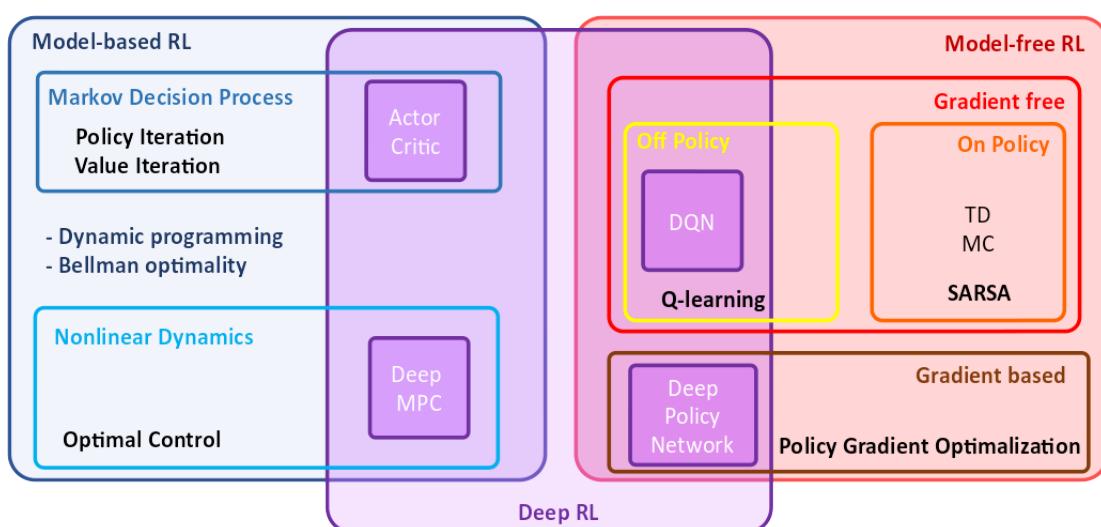
↓

**helyzethez akciókat rendel** → maximalizálja az érte járó jutalmat

- megoldható: próba-szerencse alapú kereséssel vagy késleltetett jutalmazással (állapotról állapotra lépegetünk)
- maga a fogalom 3 különböző dolgot fed le
  1. jelentheti magát a szóban forgó **problémát**, feladatot
  2. jelentheti a **megoldások osztályát** (amik jól megoldják a problémát)
  3. magát a **tudományterületet** (a feladatot és a megoldást tanulmányozza)

### 7.2. Háttere, ágazatai

- háttere, segítő tudományágak
  - az állati viselkedés pszichológiája (etológia)<sup>4</sup>, próba-szerencse módszerek
  - *optimális vezérlés* (*optimal control*), szenzorok, stb.
  - **játékelmélet** (általában diszkrét és folytonos)<sup>5</sup>
- ágazatai



17. ábra. A megerősített tanulás fajtái

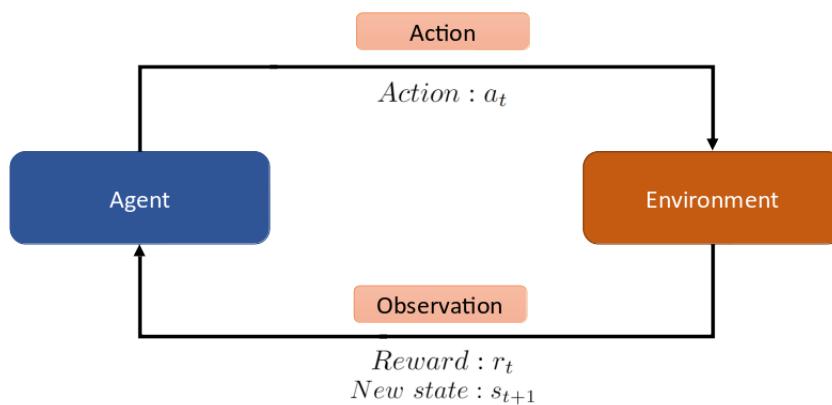
<sup>4</sup>Távolabbrá nyúlik vissza ugyan, de Pavlov kutyái és az ún. *feltételes reflex* is ideköthető.

<sup>5</sup>A sakk ideköthető példaként.

- Modell alapú RL: Markov döntési folyamat, dinamikus programozás, nemlineáris dinamikák
- Modellmentes RL: gradienscsökkentéses metódusok
  - *off-policy*: két külön térképen dolgozunk
  - *on-policy*: menet közben átirja a térképet

### 7.3. Felépítése

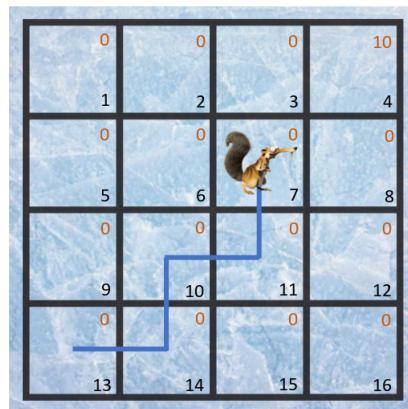
- **ágens (agent)**: akciókat hajt végre a *környezeten* → *Action* :  $a_t$ 
  - **akció (action)**: egy mozgás, amit az ágens kiválthat a környezetében<sup>6</sup>
  - **akciótér (action space)**: a lehetséges akciók halmaza →  $A = \{a_1, \dots, a_n\}$
- **környezet (environment)**: belőle megfigyelések (*observations*) áramolnak vissza az ágensbe, új állapotot és potenciálisan jutalmat eredményezve ezzel → *Reward* :  $r_t$ , *New state* :  $s_{t+1}$ 
  - **jutalom (reward)**: az a visszajelzés, ami az ágens akciójának eredményességét számszerűsíti



18. ábra. A megerősített tanulás komponensei

- a térképen az *állapotok*, *akciók* és *jutalmak* együttese határozzák meg az ágens **trajektóriáját** (útvonalát); pl. az ábrán az ágens trajektóriája:

$$(S_{13} \rightarrow A_{right} \rightarrow R(+0)) \rightarrow (S_{14} \rightarrow A_{up} \rightarrow R(+0)) \rightarrow (S_{10} \rightarrow A_{right} \rightarrow R(+0)) \rightarrow \dots$$



19. ábra. Térkép

<sup>6</sup>Gondolhatunk egy videojáték játékosára és a lehetséges „műveleteire” (fel, le, ugrás, stb.)

## 7.4. Modell alapú RL – Markov döntési folyamat

**Markov döntési folyamat** (*Markov decision process*, MDP): Az MDP egy *diszkrét-idejű sztochasztikus*<sup>a</sup> vezérlési folyamat.

Egy matematikai keretrendszernyújt a döntéshozások modellezéséhez olyan helyzetekben, ahol a kimenetelek részben véletlenszerűek és részben a döntéshozó irányítása alatt állnak.

<sup>a</sup>A sztochasztikus jelentése egy példán keresztül: ha jobbra megyek, nem garantált, hogy jobbra is megyek, de nagy esély van rá.

Az MDP **Markov-tulajdonságú**, ami annyit tesz, hogy annak a valószínűsége, hogy egy meghatározott állapotba váltsunk, kizártlag a jelenlegi állapottól és az eltelt időtől függ és független az idáig megtett állapotváltozások sorozatától.

- a feladat definíciója

- **Állapotok:**  $S := \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$
- **Modell:**  $T(s, a, s') \sim \Pr(s' | s, a)$ , ahol  $s'$  a rákövetkező állapotot jelöli<sup>7</sup>
- **Akciók:**  $A(s), A := \{\text{up, down, left, right}\}$
- **Jutalom:**  $R(s), R(s, a), R(s, a, s')$



[up, down, left, right]

20. ábra. A feladat térképe. A szürke mező falat jelent, amibe beleütközünk

- **stratégia** (angolul *policy*): az a függvényt, ami megoldja a problémát (annak definícióját),  $\pi$ -vel jelöljük. Az optimális stratégiát  $\pi^*$ -gal jelöljük
- a feladat megoldása
  - **Stratégia** (*policy*):  $\pi : S \rightarrow A, \pi(s) = a$
- a modell lehet determinisztikus vagy nemdeterminisztikus (az összegnek 1-nek kell lennie)
  - determinisztikus példa:

$$\sum \Pr = 1 \iff \begin{cases} \Pr(s_5 | s_1, a_{\text{up}}) = 1 \\ \Pr(s_2 | s_1, a_{\text{up}}) = 0 \end{cases}$$

<sup>7</sup>Ez fogalmazza meg a Markov-tulajdonságot.

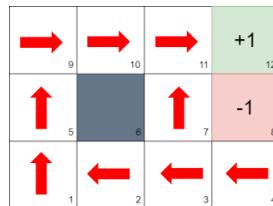
- nemdeterminisztikus vagy szochasztikus példa:

$$\sum \Pr = 1 \iff \begin{cases} \Pr(s_7 | s_1, a_{\text{up}}) = 0,8 \\ \Pr(s_4 | s_1, a_{\text{up}}) = 0,1 \\ \Pr(s_2 | s_1, a_{\text{up}}) = 0,1 \\ \Pr(s_1 | s_1, a_{\text{up}}) = 0 \end{cases}$$

pl. 80%-ban optimálisan viselkedek, de a maradékban meg ráhagyom a döntést a random faktorra → a szabadban **elengedjük felfedezi** a gépet

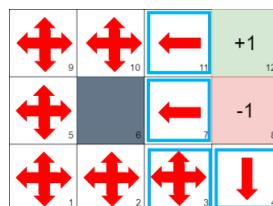
- példa

- A következő világnak az **optimális stratégiája** (*optimal policy*) az alábbi térképen látható, ahol az alapértelmezett jutalom  $R(s) = -0,4$ . minden lépés megtételével egy kicsivel csökken a jutalom, ezzel ösztönözzük, hogy ne maradjon egy helyben.



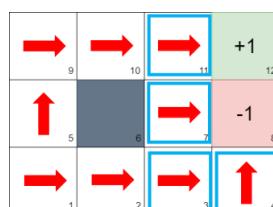
21. ábra. A feladat optimális stratégiája

- Jelöljük ki a 3-as, 4-es, 7-es, 11-es mezőket és ezekhez rendeljünk eltérő jutalmakat.
- Ha  $R(s) = +2$ , akkor az ágenst **felfedezésre** (**adatgyűjtésre**) **ösztönözzük**. Egy adott állapotban olyan akciót fog választani, amelyet még nem próbált ki. Emiatt nem feltétlenül azonnal fogja elérni a végső állapotot.



22. ábra.  $R(s) = +2$

- Azonban ha  $R(s) = -2$ , akkor a **jutalom növelése érdekében** olyan akciókat fog választani, amelyeket korábban már kipróbált. Viszont így is szuboptimális stratégiát kapunk.



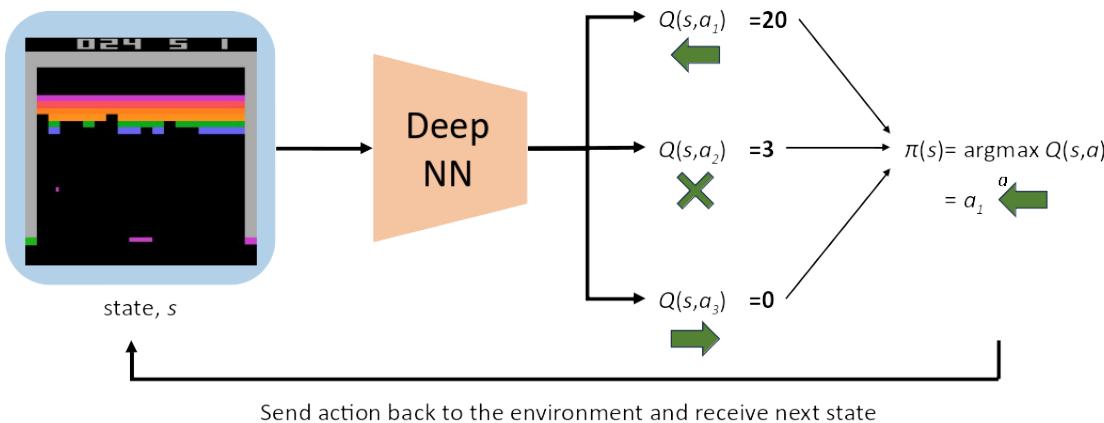
23. ábra.  $R(s) = -2$

- Akárhogy is, egy egyensúlyt kell kialakítanunk a két szempont között.

- **teljes jutalom** (*total reward*):  $R_t = \sum_{i=t}^{\infty} r_i \rightarrow$  ekkora azonnali jutalmat kapunk, hogyha elérünk egy adott állapotot
- **diszkontált teljes jutalom** (*discounted total reward*):  $R_t = \sum_{i=t}^{\infty} \gamma^i r_i \quad (\gamma \in [0, 1])$   
( $\gamma$ : diszkont faktor)  $\rightarrow R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots$
- **$Q$ -függvény**: a jövőbeli várható értékét ragadja meg az ágensnek az  $s_t$  állapotban úgy, hogy  $a_t$  akcióra számít  
$$Q(s_t, a_t) = \mathbb{E}[R_t | s_t, a_t]$$
- ezt felhasználva megkaphatjuk az **optimális stratégiát** (*optimal policy,  $\pi^*$* )  
$$\pi^*(s) = \arg \max_a Q(s, a)$$

## 7.5. Modellmentes RL – Mély RL algoritmusok

- két megközelítéssel oldhatjuk meg a feladatot
1. **Értéktanuló módszer**: keressük meg a  $Q(s, a)$ -t  $\rightarrow a = \arg \max_a Q(s, a)$ 
    - példa: **mély  $Q$ -hálózatok** (*deep  $Q$  networks, DQN*)
    - egy neurális hálót használunk a  $Q$ -függvény megtanulásához, amit majd felhasználunk az optimális stratégia kikövetkeztetéséhez



24. ábra. Érték tanulása az *Atari* videójáték példáján keresztül

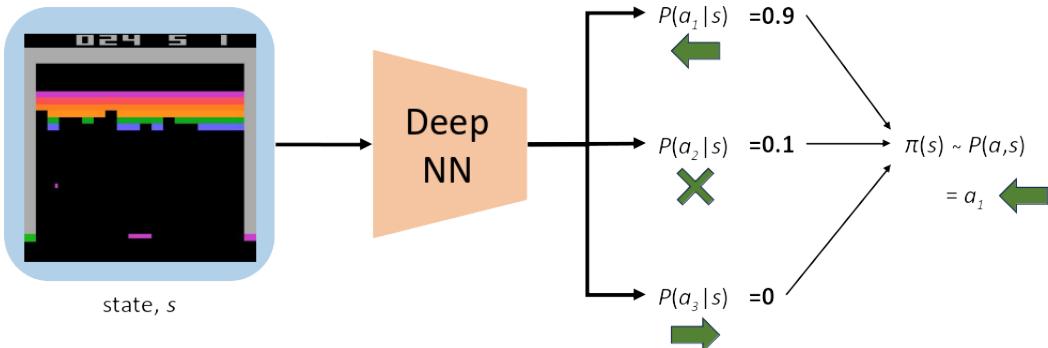
- előnyei
  - előnyös, ha az akciótér diszkrét és kevés elemből áll
- hátrányai
  - nem képes kezelni a folytonos akciótereket
  - a stratégiát determinisztikusan számítjuk ki a  $Q$ -függvényből a jutalom maximalizálásával, így szochasztkus stratégiákat sem képes megtanulni

↓

a felmerülő problémákat az RL tanító algoritmusok egy új osztályával küszöbölnétek ki **gradienscsökkentéses módszerekkel** (*policy gradient methods*)

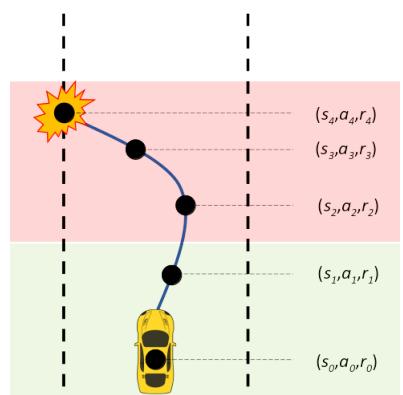
2. Stratégia-tanuló módszer: keressük meg a  $\pi(s)$ -t  $\rightarrow$  a minta legyen  $a \sim \pi(s)$

- gradienscsökkentés (*policy gradient*): közvetlenül optimalizáljuk a  $\pi(s)$  stratégiát



25. ábra. Érték tanulása az *Atari* videojáték példáján keresztül

- kapunk egy eloszlásfüggvényt, amit paraméterezhetünk a szokásás  $\mu$ -vel és  $\sigma^2$ -tel (plusz még normalizált is)  $\rightarrow$  a diszkrét esetet könnyedén átvihetjük folytonosra
  - mivel a jelen példa diszkrét, ezért  $\pi(s) \sim P(a, s) \leftarrow \sum_{a_i \in A} P(a_i | s) = 1$
  - diszkrét akciótér jelentése: „melyik irányba kell mennem?”
  - folytonos akciótér jelentése: „milyen gyorsan kell mennem?”
- előnyös megoldás, ugyanis könnyen írhatunk hozzá **szimulációt**
- tipikus példa: **önvezető autó**
  - algoritmus
    - Inicializáljuk az ágenst.
    - Addig futtatjuk a stratégiát, amíg nem terminál.
    - Rögzítsük a az összes állapotot, akciót, jutalmat.
    - Csökkentsük azon akciók valószínűségét, melyek alacsony jutalmat eredményeztek.
    - Növeljük azon akciók valószínűségét, melyek magas jutalmat eredményeztek.
  - A 4-5. lépésben a veszteséget így számíthatjuk ki:  $loss = -\log P(a_t | s_t) \cdot R_t$ .
  - Gradienscsökkentés:  $w' = w - \nabla loss = w + \nabla \log P(a_t | s_t) \cdot R_t$
  - (f) Ismétlés.



26. ábra. Önvezető autó gradienscsökkentéssel

## 8. Biológiai alapú MI megoldások

### 8.1. Inspiráció, motiváció

- agy, evolúció(természetes kiválasztódás), rajointelligence
- swarm intelligence
- egyedül elég buta, viszont együttesen a problémamegodlási képességek erős
- mérnöki szempontból izgalmasak, mely kedvező tulajdonságai vannak
- nagy létszám esetén sorozatgyártott, egyszerűen összeszerelhető, olcsók, tök egyformák
- nincs főnökük *per se* (a királynő valójában csak egy elcseszett szülőgép, különösebben leszarja a boj életét)
- másik tul: nem kommunikálnak (emberi értékelben), nincs értekezlet, számonkérés, ÖNSZER-VEZŐDŐ módon működik →
- kellemesen függ a viselkedés a dolgozók számától → graceful degradation (méltóságteljesen csökken a teljesítmény); azaz ha meghal a boj fele, akkor fele olyan hatékonyak lesznek, nem fognak ilyen hülyeségekkel foglalkozni, mint hogy gyászoljanak, stb → alkalmazkodnak a környezethez; robosztusság

### 8.2. Rajointelligence – példák, motivációk

- Louis Rosenberg – Unanimous AI
- a méhek az esetek 80%-ában optimális helyet választanak rajon belül
- táncikálnak / kommunikálnak a levegőben : sok dimenzió szerint is optimalizálhatnak (úgy tűnik)
- etológusok adtak egy szigorú algoritmust, ami leírja a mozgásukat és az infósok bebizonyították, h megoldja a feladatot
- minél izgatottabb a méh / jobban jelzi, h menjenek oda a többiek, annál izgágábban táncolnak → gyűjt követőket, elviszi őket az új hejre → toboroznak még méheket
- minél többször megy egy hejre vki, annál kevésbé lelke (ráun xdd)
- mechanizmus a végén: ha bizonyos %-nál többen jönnek egy helyre → sípolással leállítja a többit és ...

### 8.3. Térbeli csoportosítás – spacial clustering

- hangyáknál cemetary organisation – egy helyre, kupacba rendezi az objektumokat (morzsa, golyó, pete, bigyó)
- erre is van egy egyszerűen leírható algoritmus, módszer
- két dimenziós rács, a hangyák véletlenszerűen közlekednek
- ahogy mennek, érzékelik, h mi van a környezetükben
- valamennyi memóriával rendelkeznek, h megnézzek, melyikből volt több idáig (piros, szörge)
- ha outlier találnak, az t áthelyezik a megelelő helyre
- lokális megfigyelés alapján optimalizáció
- egyszerű, de nagyon jól működik

#### **8.4. Legrövidebb utak**

- állítás: hangyák tudnának legrövidebb utat keresni két pont között
- nagyjából egyenes vonalban, de EGYMÁST KÖVETVE közlekednek
- közelisége a legrövidebb utak problémájának
- sztochasztikusan legrövidebb (nagyobb a valószínűsége, h rövidebb lesz,mintsem, h hosszabb)
- a) találunk kaját b) ha találtunk, leghatékonyabb aknázzuk ki (legközelebbi) c) legrövidebb úton szállítsul el
- megfigyelések: ha van választási lehetőség pathból, egyirányban a rövidebbet választják; akadályt raktak a legrövidebb útba aszimmetrikusan → a rövidebb úton kerülik ki
- erre is adtak matematikai modellt az etológusok, majd bebizonították a matekosok, h valóban megoldja a legrövidebb utak problémáját
- megoldás: feromonok (vmi, ami a fajtársakra hat – itt, élelem megtalálását segíti) → nem egymással, hanem az illatnymokkal kommunikálnak
- kétféle feromon: 1) ha keresi az élelmiszerforrást; 2) megvan a forrás és vissza akarja juttani a fészekbe
- a hazafele tartó olyat áraszt ki, amelyik mutatja, honnan hozta a kaját
- ILLÉKONY ANYAGOKRÜÓL VAN SZÓ
- illékony: terjed, eloszlik, de egyre gyengébb is lesz
- arra mennek, ahol a legerősebbnek érzékelik, de ezt sztochaszikusan (tehát nagyobb valószínűségű lesz az erősebb, de sosem biztos esemény) → viszont ameddig járkálnak az útvonalon, addig erős lesz a feromon (lasabban illik el); szóval a rövidebb útvonalat könnyebb megerősíteni, mint a hosszabbat;
- modellezhető 2d rácson
- erre alkottak absztrakt optimalizációs módszert (gráfokra értelmezett) → sok probléma megoldható vele
- marco dorigo

## 9. Evolúciós algoritmusok

### 9.1. Motiváció

- ha már MI, érdemes a TI-t (terézeszetes intelligencia) is tanulmányozni
- utánozzuk le az evolúciót
- fontos jelenség a tanulás → másoljuk le ezt
- további fontos fogalmak: versenyzés, reprodukció, raj(zás), kommunikáció – motivációként szolgálhatnak; ennek mesterséges létrehozása

### 9.2. Háttere, közös jellemzők

- bizonyos tanulásoknál a kiértékelések után tudunk változást létrehozni
- szemben az evolúciónál, a változások értékelődnek ki; melyből a jók élnek túl (evolúció = változások eredménye)
- ki él túl? aki a legjobban adaptálódik a környezethez
- keresési vagy tanulási problémák → megfalmazhatók optimalizációs feladatokként
- leegyszerűsíthetjük az optimalizációra a problémát
- megközelítések: determinisztikus (hegymászó algoritmus, analízis alapú deriválgatók faszságok); szochasztikus (véletlenkeresés, klasszikus szimulált lehűtés, evolúciós algoritmusok)
- klasszikus optimalizáció lokális optimalizációt hajt végre → találjuk meg ebből a globális optimumot
- egyedekbe kódoljuk be az optim. feladatot → ezek változnak az idő függvényében → egyre jobb megoldást adnak az adatoptimalizációs feladatra
- '50-es, '60-as évekre nyúlik vissza a gondolat
- ágai, iskolái: evolúciósprogramozás, ..., vmi nszk-s bigyó
- fogalmak: gén, allén, egyed, genotípus, fenotípus, poluláció,
- szétszórjuk az egyedeket (lehet sokdimenziós vektor) véletlenszerűen; bizonyos változások segítségével generációról generációra fejlődik az operáció → a globális optimum felé kezd sűrűödni a populáció

### 9.3. Genetikus algoritmusok

- 3 operátor. keresztezés, mutáció, szelekció
- minden egyedhez hozzárendelhetünk egy fitnessz értéket (jobb egyedeknek nagyobb szelekciós valószínűség)
- egyed: megoldása az adott problémára
- fitnesszérték: mennyire jó az adott egyed
- szekelcios módszerek: rulett-kerék alapú szelekció
- keresztezés: lehet bináris, intédzser, stb a kódolása → két szülőegyedet kiválasztunk, véletlenszerűen levágjuk őket, majd a két szülő két gyerekkel kapunk a szeletek kereszteződéséből (konkatenáció)
- mutáció: (pl. véletlenül megváltoztatunk egy bitet)

- példa:  $f(x) := x^2$  ( $x \in \{0, 1, \dots, 30, 31\}\right) =: I$ , így  $\max_{x \in I} f(x) \rightarrow$  5-biten eltárolhatunk 32 számot (genotípus), fenotípusa ennek a bináris számnak a decimális alakja; populáció legyen 4.
- actual count: kiválasztások száma (ha 0, akkor az adott egyed kihalt)
- bizonyítható, hogy a határérték az optimum felé tart
- alternatív változatai:  $n$ -pontos keresztezés, uniformis keresztezés (maszksztringet definiálunk az érmefeldobásra, ha fej, cerélünk, ha írás, nem)
- ezt az ötletet viszi tovább: genetikus programok, genetikus programozás
- fákon ( $\rightarrow$  valójában programok, kifejezésfák) hajtják végre a műveleteket
- egy fa leír egy képletet (pl. fonya, yaaay)
- a fákon (ahogy korábban stringeken) hajtjuk végre az evolúciós algoritmust
- tekinthetjük úgy, hogy a fa a program és ezt optimalizáljuk (see: reverse Polish notation, prefix notation)
- függvény-node-ok : amik nem lehetnek levelek (tehát nemlevél csomópontok)  $\{+, -, :, /, \%, IF\}$
- a többi meg szám
- mutáció: kiválasztunk egy függvénycsomópontot és rögzítjük váletlenszerű részfára cseréljük, vagy a levelet cseréli lege ygelőre definiált levélsúcsból
- keresztezés: kicséréli a részfákat
- a példa felügyelt tanulásos minta

## 10. Rajintelligencia

Ez a másik fajta evolúciós MI. Önállónak tekinthetjük akár, de van köze az előbbihez.

### 10.1. Bevezetés, motiváció

- biológia... mint az előző előadáson
- korábban szinte kizárolag matematikai megközelítéssel kutatták az MI-t
- ez eltolódott a biológia általi inspiráció felé
- kevésbé esett róla szó: kollektív intelligencia, egyedek közti kommunikáció
- stigmechia
- rajointelligencia
  - keresés és MI feladat megfogalmazható optimalizáció formájában
  - optimalizációs kritérium (felügyelet nélküli)
  - rendszerek együttműködése optimalizációt hajt végre basically
  - példák: madarak, halak, hangyák, méhek, stb.

### 10.2. Particle swarm optimisation (részecske raj optimalizáció)

- '90-es évek közepében jelent meg, azóta meg rendkívül dinamikusan fejlődő terület
- ágens, részecske; folytonos vektorként képzeljük el őket; valamilyen műveletet végeznek (pl erpülnék)
- pbesz: minden részecskére vontakozik, personal best, általa talált legjobb helyszín a repülés során
- gbest: globális legjobb
- ezek vezérlik az egyes részecskék repülését
- példa:  $k$ -adik iterációban a részecske eddigi legjobb és a raj eddigi legjobbjá

$$(gbest)^k$$

$$s^k \quad (pbest)^k$$

- tegyük fel, h aott van a  $pbest^k$  értéke
- $v^k$  a sebességvektora a részecskének
- a pbest és a gbest az, amik **kitérítik a vektor irányát**
- távolság a pbest és az  $s$  között:  $d^{pbest^k}$
- ugyanez a gbest irányában is
- eredőjét számíthatjuk a kettőnek  $\rightarrow v^{k+1}$  megkaojuk az új sebességet, vektort
- nagyon hatékony, egész egyszerű
- meghatározzuk a  $\Delta v^k$ -t, súlyozzuk ezeket (akár véletlenszerűséget is bevihetünk a súlyokba)
- $w_1 = c_1 \cdot rand()$  és  $w_2 = c_2 \cdot rand()$   $\rightarrow$  random generátor
- pozíciót és sebességet összeadhatunk a fizikában? ... végülis igen, mert diszkrét lépések vannak
- yaaay, kapunk stukiiit... csak szövegeset

- véletlenszerűen létrehozzuk az ágenseket / részecskéket / egyedeket → iterációk elteltével elkezdenek sűrűsödni egy adott pontban
- szokták versenyeztetni az ilyen függvényeket, benchmarkok

### 10.3. Simplified Swarm Optimisation

- egyed = vektorok;  $x_{ij}^{t+1}$ :  $x$   $i$ -edik azonosítójának  $j$ -edik koordinátája a  $(t + 1)$ -edik iterációban
- négy eset, más-más valószínűséggel választjuk, melyiket választjuk ( $\rho \in [0, 1]$ )
- ez is egyszerű, hatékony

### 10.4. Szentjánosbogár algoritmus (firefly algorithm)

- #define firefly
- mire használják a világításukat? → párzás, figyelmeztetési mechanizmus, áldozatok becserkészése
- fényintenzitást fogjuk használni a zalgorithmusban
- arányos lesz ez az attraktivitással, ami meg a távolsággal lesz fordítottan arányos
- $I_0$  : konstans érték, be van szorozva a két paraméter közti távolsággal
- mozgás: random mozgág + vonzóbb dolgok felé mozgás; egy  $\alpha, \beta$  paramétert kapunk
- nem érdekesek az edgecase-ek

### 10.5. Gravitációs keresőalgoritmus

- minden egyednek van tömege; minél nagyobb a tömege, úgy vonzást számolhatunk, sebesség, stb., fizika
- számos ilyen algoritmus
- nem kell 100%-osan követni a valóságot, elég, ha csak őegy modellt állítunk fel
- $G(t)$
- erőhatás, Newton törtévnnye, gyorsulás mértéke
- a  $t$  az egy, mert diszkrét idő, tehát összeadhatunk sebességet és koordinátát
- az összes ilyen algoritmus STRUKTÚRÁJA
- paraméterezés, kezdeti populáció, fitnessz kiértékelés, fő működési mechanizmus, megolások frissítése, majd vége

## 11. Etorobotika

Etorobotika (*ethorobotics*) = etológia (viselkedéstan) + robotika

### 11.1. Motiváció

- több AI működik egy szerkezetben, robotban (beszédszintézis, feltérképezés, mozgás)
- cél: lényegében egy transzformer építése
- Vector (Anki szüleménye) – 4 magnyi processzor, mi, objektumdetekció, hangfelismerés, navigáció
- hogyan tervezünk olyan szociális robotokat, melyek napi szinten kapcsolatban vannak az emberrel
- viselkedéselemek pótlása
- érzelmeket hogyan pakoljuk bele (kezdetben macskaszerű volt)
- arckifejezésekhez érzelmeket azonosítunk
- mi a célja: robot integrálása emberi környezetbe, viszont nem várjatjuk el, h a környezet rendelkezzen programozói előképzettségekkel → robot viselkedése : etológia + robotika

### 11.2. Etorobotika

- motorikus funkciók (felemelni vmit) → viselkedésminták kialakítása (kockát felvesz, elvisz vhoval)
- viselkedés (játék: megkeres kockát, el kell vinni az emberhez) → érzelmek
- idegesség – gyors, heves mozdulatok
- fókuszáltság – lassabb, koncentráltabb

### 11.3. Etológiai mintánk

- KUTYÁK: jól integrálódtak, jól ismert viselkedésminták, hosszú évezredek tapasztalata
- kutatási és fejlesztési folyamat

etológiai kísérlet: ember–állat interakciója



etológiai viselkedésmodell



matematikai modell



robotvezérlés



etorobotikai kísérlet

- lényegében egy flowchart (gagyi struktogram) írja le a viselkedését
- egyszerű példa: napraforgó robot
- explicit megfigyelések alapul → hogyan tehetjük tanulhatóvá

#### 11.4. Etológiai kutatási folyamat

- megfigyelések, rögzítjük
- nézzük, meddig csinálják az adott viselkedést
- teszt: Ainsworth-teszt: kötődés vizsgálata a kisgyerek és gondviselője között → etológusok ki-terjesztették ember-kutyára → robot-ember viszonyrendszerben visszamérhető-e?
  1. hozzászoktatás
  2. idegen megjelenik
  3. gazdi kimegy, első szeparáció
  4. első visszatérés, reunió
  5. kutya egyedül, második elkülönülés
  6. szeparáció folytatása az idegennel
  7. második visszatérés a gazdahoz
- definiálunk ey pontosabb mérőrendszert... ezeket rootok mérésére hogyan?
- amúgy ezt az előadó csinálta (wow)
- nemcsak, h mit szeretnénk mérni az állaon
- mit tudunk belerakni a robotba? top-down (tudjuk, h mit csináljon a robot, ezt kell megvalósítani), bottom-up (szenzorok, motorok → na, együtt mire képesek?)
- borzongások völgye (uncanny valley), emberszabádú robotoknál jelent meg
- csak azért tegyünk be valamit, ha van funkciója (a roboton a realisztikus bőr nem az, mert az embernek a legnagyobb érzékszerve, de a robot nem használja)
- feature matching

#### 11.5. Kísérletek

- Biscee (kék): felszolgáló robot, kommunikáljon a vendégekkel
- Ethon (piros)
- differenciált ???-vel rendelkeztek
- mecanumbot, raspberry pi, jobban modellezhetők az állati mozgások, a számításigényes műveleteket kirendelhetjük egy szervernek (amin a nagy nyelvi modell van pl.)
- külső megfigyelő rendszer: MoCap (CGI, motion capture system, filmekben használják)
- mozgásnak leképezése, matematikai modellje
- robohoz rögzített koordináta rendszer jobb választás volt, mint egy abszolút koordrend
- etogrammok kitöltése, sokkal pontosabbak egy robot esetében (hatékonyabb is, mint amit az etológusok végeznek)

## **12. Multiágens szimuláció és tanulás (Multi-agent simulation and learning)**

Inkább érdekességeként szerepel az előadásban.

### **12.1. Motiváció**

- gyakori képzet, hogy „egy nagy valami egy nagy dolgot végez el”
- ezzel szemben ez egy többszereplős rendszer (ahogy a világ is az általában)
- „két mesterséges intelligencia beszélget...”
- néha ügynöknek is nevezik, ám ez félreérthető
- ágens: olyan szereplő, ami saját maga rendelkezik a cselekedeteiről (autonómia)
- láttunk hasonlót: foraging ants (hangyák)
- kooperáció vagy versenyhelyzet
- szorosan kapcsolódnak játékelméleti gondolatok

### **12.2. Mikro- és makroviselkedés**

- többszereplős rendszereknél eme két szintet szoktuk vizsgálni: mikro (egyes szereplők) és makro (a teljes rendszerszintű működés)
- felfedezhetünk érdekes, előre be nem látható következményeket a rendszer szintjeiben
- ilyenek pl. az emergens tulajdonságok (megjelenő, előbújó)
- individuálisan egyik egyed sem rendelkezik vele (pl. dugóval), viszont egy nagy rendszerben már megjelenhet
- két féle megközelítés:
- ágens-alapú szimuláció (analízis): adott pár mikroszabály az egyedek szintjén, melyek makro szinten generálnak makroviselkedést → egy eszköz vminek a megmagyarázásához
- multiágens (megerősítéses) tanulás (konstrukció): kooperálni, rajzani tanulnak meg

### **12.3. Ágens alapú szimuláció**

- klasszikus példa: detroitban szegregáltan élnek → miért van? ... nehezen ellenőrizhető dolgokat sorakoztatunk fel; eme tesztelés nélküli véleményeket vizsgálhatjuk az alábbi módon  
alkotunk egy szabályrendszert, szimulációt, modellt és nézzük meg, hogy a jelenség emergál-e
- funky közgazdasz: mátrixot rajzolt fel, egy négyzetrács egy lajhelyt jelent, kék és piros "családok" (vagy üres) laknak benne. időközönként megyizsgálják, hogy hány velük azonos színű lakik a szomszédukban. ez lesz a toleranciája, ha ezt megüti, akkor elköltözik
- továbbra is mindenki marad a helyükön, ha 70%-a → továbbra is előbújnak eme szegregációk
- megmutatja, hogy a szegregáció egy rendkívül erős emergens jelenség
- hasonlót vizsgáltak már vírusterjedést, birkanyájakat

### **12.4. Multiágens megerősített tanulás**

- mi van, ha több genst szeretnék egyszerre tanítani?
- cél, hogy közösen oldjanak meg egy problémát

- a) vannak benne specialisták (designer, projekvezető, tesztelő, fejlesztő)
- b) mindenki egyforma és közösen küzdenek a probléma megoldásáért, nincs hierarchia, csupán egymást érzékelve dolgoznak (extra: lehessen számtól független, magyarán fele annyi egyeddel is kezdhessünk valamire)
- vesszük ezeket az üres agyú robotokat, megfogalmazunk nekik egy (egyszerű) feladatot (általában egymáshoz képest mozgási feladatok – álljanak libasorba)
- közlekedési lámpák vezérlése, calen szinkronizációja