Írta:
PIGLERNÉ LAKNER ROZÁLIA
STARKNÉ WERNER ÁGNES

ÁGENS-TECHNOLÓGIA

Egyetemi tananyag

TYPOTEX

2011

LEKTORÁLTA: Dr. Borgulya István, Pécsi Tudományegyetem Közgazdaságtudományi Kar Gazdaságmódszertani Intézet

TÁMOGATÁS:
Készült a TÁMOP-4.1.2-08/1/A-2009-0008 számú, „Tananyagfejlesztés mérnök informatikus, programtervező informatikus és gazdaságinformatikus képzésekhez” című projekt keretében.

ISBN 978-963-279-527-0

KÉSZÜLT: a Typotex Kiadó gondozásában

FELELŐS VEZETŐ: Votisky Zsuzsa

AZ ELEKTRONIKUS KIADÁST ELŐKÉSZÍTETTE: Csépány Gergely László

KULCSSZAVAK:

mesterséges intelligencia-ágensek, multi-ágens-rendszerek; keresési módszerek; tudásbázisú ágens; tudásreprezentáció; bizonytalan adatok kezelése; szakértői ágens; gépi tanulás; genetikus módszerek; neurális hálózatok; robotika; virtuális valóság; gépi látás; beszédfelismerés.

ÖSSZEFOGLALÁS:

A Veszprémi Egyetemen (jelenleg Pannon Egyetem) 1992-ben indult informatikatanár-képzésben folyamatosan oktatunk mesterséges intelligenciához kapcsolódó ismereteket kötelező tárgyak keretében. Ezeket az órákat mind a nappali, mind a levelező hallgatók szívesen látogatták és ismerkedtek meg olyan területekkel, amelyek informatikai műveltségüket emelték.
Emellett olyan témákról is hallhattak, amelyek az oktatás területén is felhasználhatók.
A mostani képzési formában a hallgatók érdeklődése a téma iránt nem csökkent, ezért úgy gondoljunk, hogy egy célirányosan számukra összeállított, összefoglaló elektronikus jegyzet segítségével a rövidebb képzési idő ellenére is átfogó képet tudunk adni ezen tudományterület sokszínűségéről.

Tartalomjegyzék

1. fejezet

Bevezetés

A Veszprémi Egyetemen (jelenleg Pannon Egyetem) 1992-ben indult informatikatanár képzésben folyamatosan oktatunk mesterséges intelligenciához kapcsolódó ismereteket kötelező tárgyak keretében. Ezeket az órákat mind a nappali, mind a levelező képzésben részt vevő hallgatók szívesen látogatták és ismerkedtek meg olyan területekkel, amelyek informatikai műveltségüket emelték. Emellett olyan témákról is hallhattak, amelyek az oktatás területén is felhasználhatók.

A Bolognai folyamat azonban átalakította a tanárképzés folyamatát, és az informatikatanár MSc képzés ideje alatt kevesebb időt biztosít a mesterséges intelligencia témakörök bemutatására. Ezért szükségessé vált egy rövidebb terjedelmű, átgondoltabb jegyzet elkészítése.

A mesterséges intelligencia kutatások célja olyan rendszerek (számítógépes programok, robotok) létrehozása, amelyek intelligens módon képesek feladatokat megoldani. Az utóbbi években egy új szemléletű, viselkedésalapú megközelítés van terjedőben, amely ezt a feladatot úgy fogalmazza meg, hogy a mesterséges intelligencia kutatások, fejlesztések célja, hogy a feladatmegoldást olyan szereplőkkel, *ágens*ekkel végeztesse el, amelyek az intelligens viselkedés bizonyos vonásaival rendelkeznek. Ezért ezen új felfogás tükrében, az elsősorban informatikatanároknak szánt elektronikus jegyzet az „Ágens technológia” elnevezést kapta.

A mesterséges intelligencia ismeretek oktatásához 2004-ben [19] készült egy egyetemi jegyzet informatikatanárok számára, de az ismeretek folyamatos változása és a korábban említett képzési folyamat megváltozása miatt szükségessé vált egy új elektronikus jegyzet megtervezése, elkészítése.

A mostani képzési formában a hallgatók érdeklődése a téma iránt nem csökkent, ezért úgy gondoljunk, hogy egy célirányosan számukra összeállított, összefoglaló elektronikus jegyzet segítségével a rövidebb képzési idő ellenére is átfogó képet tudunk adni ezen tudományterület sokszínűségéről.

Nagy örömünkre szolgál, hogy ezúttal egy magyar nyelvű, a magyar hallgatók által szabadon hozzáférhető tankönyv készülhetett el a TÁMOP-4.1.2-08/1/A-2009-0008 program támogatásával.

Az egyes fejezetekhez – a terjedelmi korlátok által megengedett mértékben – igyekeztünk kidolgozott példákat és egyszerű animációkat is biztosítani, ami az elméleti anyagrészek megértését segítheti.

Reméljük, hogy elektronikus tankönyvünk nemcsak az informatikatanár szakos érdeklődő hallgatókat segíti majd abban, hogy az ágens technológia érdekes és izgalmas világával megismerkedjenek, hanem más szakos hallgatók is érdeklődéssel olvasnak el egy-egy fejezetet vagy a teljes jegyzetet.

Veszprém, 2011. június 17.

Lakner Rozália, Werner Ágnes
Pannon Egyetem
Műszaki Informatikai Kar

2. fejezet

Ágensek, multi-ágens rendszerek

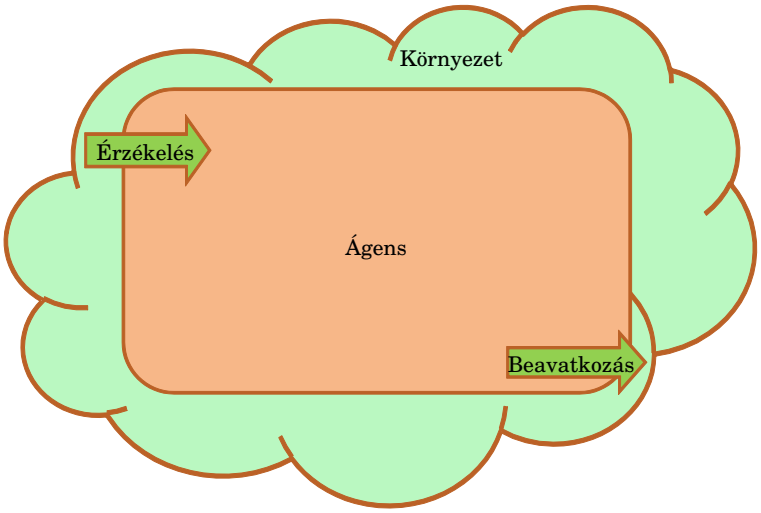
2.1. Alapok

A számítógépek megalkotása óta foglalkoztatja a kutatókat az a probléma, hogy számítógép-programok és rendszerek mint önálló entitások fel tudják-e venni a versenyt az emberrel bizonyos területeken. Az egyszerű számológépektől kiindulva a fejlődés napjainkban elérte azt a szintet, hogy a számítógépek alkalmazásával meglehetősen összetett problémák kezelése, támogatása vált elérhetővé. Néhány példát említve vállalatirányítási rendszerek, ipari folyamatok monitorozó és szabályozó rendszerei, orvosi diagnosztikai rendszerek, tervező rendszerek, robotok segítik az ember mindennapi munkáját. A gépek és az emberek „versengése” azt a tendenciát eredményezi, hogy ezeket a rendszereket egyre inkább felruházzák az emberekre jellemző képességekkel (például tanulás, természetes nyelv megértése, képek, hangok értelmezése), így ezek a rendszerek adott problématerületükön az emberrel egyenrangú szakértőknek, intelligens gépeknek tekinthetők.

Egy intelligens rendszer a problémamegoldás során megszerzett ismeretei alapján oldja meg az adott feladatot és megadja ennek eredményét. Egy ilyen viszonylag független, önálló egységet **ágens**nek tekintünk. Az ágens eredete a latin ago, agere szó, melynek elsődleges jelentése mozgásba hoz, elintéz, cselekszik. A szó alapjelentése szerint ágens lehet mindaz, ami bizonyos fokú önállósággal bír, valamilyen környezet veszi körül, továbbá ezt a környezetet érzékeli és szükség esetén környezetébe beavatkozik. Ebben az értelemben ágensnek tekinthető egy ember, aki érzékszervei (pl. fül, szem, orr) segítségével érzékeli környezetét és különböző testrészei (pl. száj, kéz, láb) segítségével beavatkozik. Ágens lehet egy robot is, amely kamerái, szenzorai segítségével érzékel és motorokkal vezérelt beavatkozói (pl. robot karok) segítségével beavatkozik. Ágens lehet továbbá egy szoftver is, amelynél input és output adatok (kódolt bitsorozatok) formájában történik az érzékelés és a beavatkozás. Ki kell hangsúlyozni azonban, hogy környezet nélkül nem ágens az ágens. Például egy programkód önmagában nem tekinthető ágensnek, megfelelő környezetbe ágyazott működése során válik azzá.

2.2. Ágensek és tulajdonságaik

Az ágens egy környezetébe ágyazott entitás, amely a környezetének aktív részeként azzal kapcsolatban áll, szükség esetén környezetébe beavatkozhat, vagy kommunikálhat más ágensekkel (lásd 2.1. ábra).



2.1. ábra. Ágens és környezete

(Megjegyzés: Az ágens fogalmának könnyebb megértéséhez futtassa az agensszemantikus.exe fájlt.)
Egy ágens legfontosabb tulajdonságai a következők:

- Környezetbe ágyazottan működik, a környezetén kívül állapotai nem értelmezhetők.
- Képes a környezetét észlelni, azaz figyeli a környezet valamely tulajdonságát vagy tulajdonságait, valamint ezek változásait.
- Képes a környezetére hatni, azaz valamilyen cselekvés végrehajtásával a környezetébe beavatkozhat.
- Autonóm, azaz önálló működésre képes.

Egy ideális ágens rendelkezhet további tulajdonságokkal is:

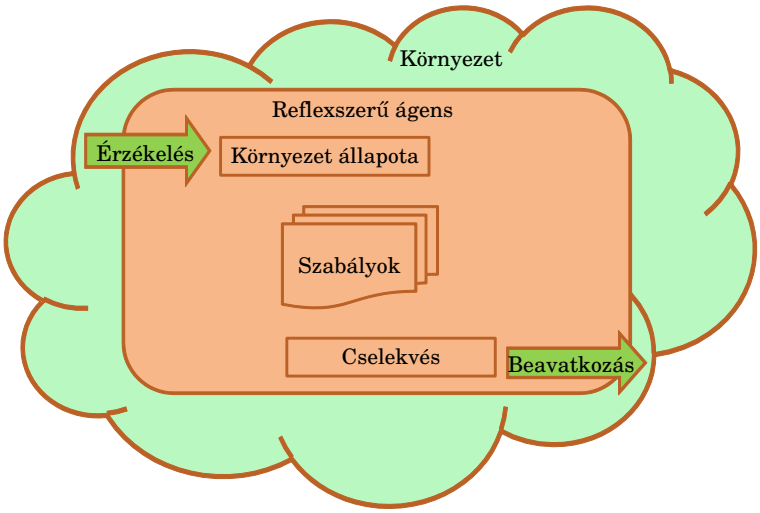
- Képes a többi ágenssel kommunikálni, például helyzetét jelezni, tudását megosztani.
- Célvezérelten működik, céljai elérése érdekében cselekszik, amely cél általában a rendszer globális céljának elérése (pl. mattadás egy sakkjátszmában, padló koszos részének megtisztítása).
- Környezetéről csak részleges információkkal rendelkezik, például csak a közvetlen környezetét érzékeli.

- Saját erőforrásokkal rendelkezik.
- Racionálisan, helyesen cselekszik, azaz nem cselekszik tudatosan céljai ellen és igyekszik a lehető legjobb alternatívát választani.

2.3. Ágensek csoportosítása

Az ágenseket képességeik és viselkedésük alapján a következő csoportokba sorolhatjuk.

2.3.1. Reflexszerű ágensek



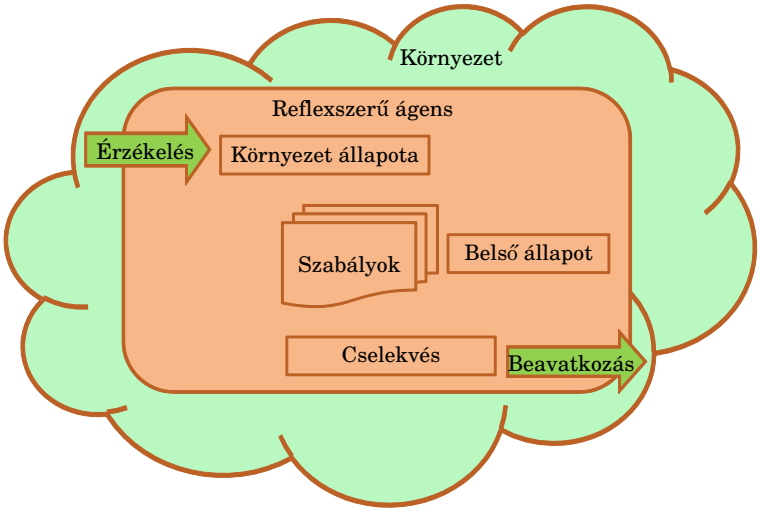
2.2. ábra. Reflexszerű ágens

A **reflexszerű ágens** (lásd 2.2. ábra) feltétel-cselekvés szerkezetű szabályai (szabályokról részletesebben a 6. fejezetben lesz szó) alapján gyors, egyszerű működést lát el (például egy autóvezető ágens szabálya „ha az előző autó fékez, akkor kezdj fékezni”). Az érzékelés ebben az esetben közvetlenül meghatározza a végrehajtandó cselekvést, ezért az ágensnek nem szükséges információt tárolnia a múltbeli viselkedéséről. Működése nagyon egyszerű: észleli a környezet jelenlegi állapotát, keres egy ehhez az állapothoz illeszkedő szabályt, majd végrehajtja a szabály következmény részében szereplő cselekvést. Ilyen működést mutat például egy egyszerű helyesírás-ellenőrző vagy egy adatgyűjtő ágens.

2.3.2. Belső állapottal rendelkező ágensek

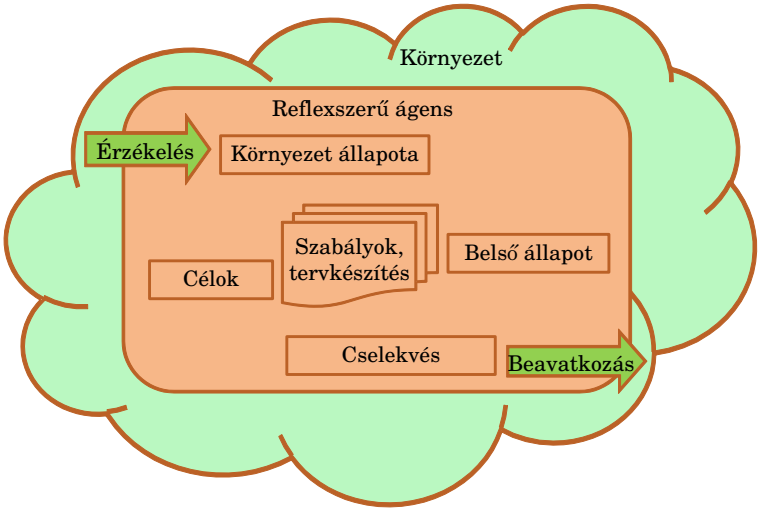
A **belső állapottal rendelkező ágens** (lásd 2.3. ábra) olyan reflexszerű ágensként fogható fel, amely a világ korábbi állapotát belső állapotában tárolja. Így nemcsak az aktuális állapot ismeretében dönthet, hanem a korábban észlelt értékeket is figyelembe veheti. A korábbi állapot nyilvántartása összehasonlítási alapot szolgáltat arra vonatkozóan, hogy mi változott.

Ebben az esetben kétfajta információ figyelembevétele szükséges: egyrészt hogyan változik a világ az ágenstől függetlenül (például előzést végrehajtó autó helyzetének változása az ágenshez képest), másrészt az ágens cselekedetei hogyan befolyásolják a világot (például sávváltás esetén üres hely marad a korábban használt sávban).



2.3. ábra. Belső állapottal rendelkező ágens

2.3.3. Célorientált ágensek



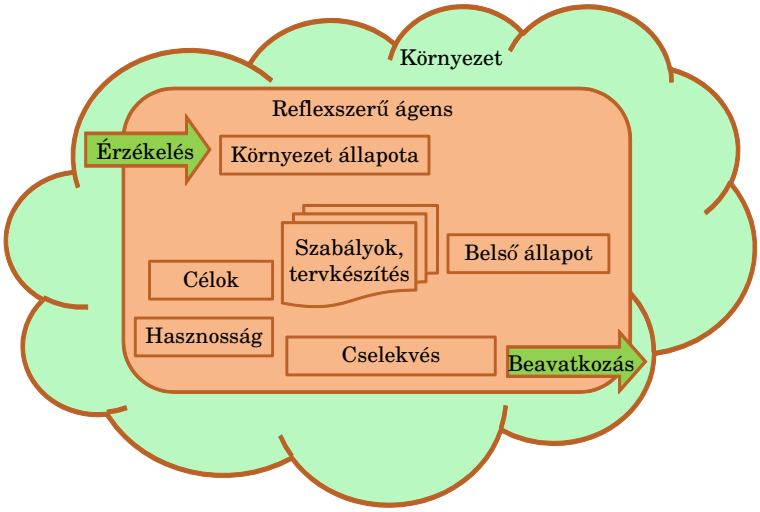
2.4. ábra. Célorientált ágens

A **célorientált ágens** (lásd 2.4. ábra) rendelkezik egy elérendő állapottal, céllal is (például autóvezető ágens meghatározott úticéllal). Az ágens ebben az esetben a cél elérése

érdekében tervet készít cselekedete előtt. Ez a feladatok többségénél bonyolult, többlépéses következtetést igénylő feladat, amely magában foglalja a keresést (lásd 3. fejezet) is.

2.3.4. Hasznosságorientált ágensek

A célorientált ágensek elérendő céljuknak megfelelően többlépéses tervet készítenek cselekedeteik előtt. Az ágensek számára megadott célok azonban nem mindig egységesek, több cél esetében lehetnek ellentmondóak. Ebben az esetben egy úgynevezett hasznossági függvény megadásával két vagy több állapot összehasonlíthatóvá válik, ezzel az ellentmondás feloldható. A **hasznosságorientált ágens** (lásd 2.5. ábrát) a hasznossági függvényt felhasználva hoz döntéseket, készít tervet. Erre példa egy olajfinomító-vezérlő ágens vagy egy tőzsdei részvény-vásárló ágens.



2.5. ábra. Hasznosságorientált ágens

2.4. Ágens környezetek

Az ágensek működése az ágensek képességén kívül a környezetük tulajdonságaitól is függ. A továbbiakban áttekintjük a környezetek legfontosabb osztályozási szempontjait.

2.4.1. Hozzáférhetőség

Egy környezet *teljesen megfigyelhető*, ha az ágens érzékelő berendezései révén hozzáférhetőség biztosított a környezet teljes állapotához. Ez kényelmes lehet, hiszen a környezet változásának nyomon követéséhez nem kell nyilvántartani semmilyen belső állapotot. Zajos, pontatlan érzékelők esetén vagy amennyiben bizonyos állapotok nem hozzáférhetők, *részlegesen megfigyelhető* környezetről beszélünk.

2.4.2. Meghatározottság

Determinisztikus környezetről beszélünk akkor, ha a környezet következő állapotát egyértelműen meghatározza az előző állapot és az ágens cselekedete (például sakkozó ágens). Ellenkező esetben a környezet *sztochasztikus* (például autóvezetésnél a defektet véletlenszerű események tekintjük). Amennyiben a környezet teljesen megfigyelhető és determinisztikus együttesen, akkor az ágensnek nem kell bizonytalanságot kezelnie.

2.4.3. Epizódszerűség

Az epizód észlelések és cselekvések egy jól elkülöníthető sorozata, amelyben az elemi epizódok cselekedetei nem függnek az előző elemi epizódok cselekedeteitől (például egy hegesztő robot minden hegesztendő alkatrészt az előzőleg elkészített munkadarabtól függetlenül kezelhet). *Epizódszerű* környezet esetén az ágens tapasztalata epizódokra bontható, és cselekvése kizárólag az adott epizódtól függ. *Sorozatszerű* környezet esetén minden egyes döntést a korábbi cselekedetek is befolyásolhatják (például a sakk sorozatszerű környezetnek tekinthető, azonban egy sakkverseny játszmái már epizódszerűek).

2.4.4. Változékonyság

Statikus környezetről beszélünk ha a környezet csak az ágens cselekedete következtében változhat (például sakk). A statikus környezet kezelése egyszerű, mivel az ágensnek nem kell figyelnie a környezet állapotát, miközben gondolkodik. Amennyiben a környezet (időben) folyamatosan változhat, úgy az *dinamikus* (például autóvezetés).

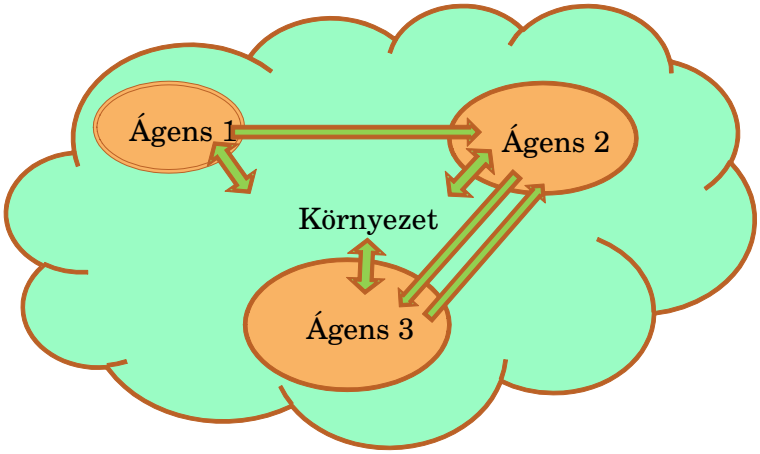
2.4.5. Folytonosság

Diszkrét környezet esetén az észlelések és cselekvések véges halmazzal adhatók meg (például sakkjátszma esetén véges számú lehetséges lépés és véges számú diszkrét állapot van). *Folytonos* állapotú környezet például az autóvezetés, ahol az autók sebessége, gyorsulása folytonos értékkészletű.

2.5. Multi-ágens rendszerek

A több, egymással kölcsönhatásban álló ágensből álló rendszereket **multi-ágens rendszer**eknek vagy **több-ágenses rendszer**eknek nevezzük. Egy multi-ágens rendszer részei a következők:

- Környezet, amely gyakorlatilag egy kiterjedéssel rendelkező tér.
- Ágensek, amelyek a rendszer aktív elemei.
- Objektumok, amelyek a környezetben helyezkednek el. Ezek passzív elemek, amelyeket az ágensek érzékelhetnek, létrehozhatnak, megsemmisíthetnek, módosíthatnak.



2.6. ábra. Multi-ágens rendszer

- Műveletek, amelyek segítségével az ágensek érzékelik és manipulálják az objektumokat.
- Relációk, amelyek objektumok és ágensek kapcsolatát írják le.
- Környezet sajátosságait leíró szabályok, műveletek.

A multi-ágens rendszerben szereplő ágenseknek általában korlátozott tudásuk, érzékelési és beavatkozási hatáskörük van. Jellegzetes képességük a kommunikáció, amely segítségével megoszthatják egymással információikat, a kooperáció, amely az együttműködést biztosítja céljaik elérésében, valamint a koordináció, amely összehangolja a rendszer működését.

2.6. Alkalmazási területek

Az ágens rendszerek néhány jellemző felhasználási területe:

- információs ágensek,
- interfész ágensek,
- asszisztensek,
- ágens-alapú szimulációk,
- szoftvertechnológiai alkalmazások,
- intelligens épületek.

3. fejezet

Keresések

3.1. Alapok

A **mesterséges intelligencia** olyan feladatok számítógépes megoldásával foglalkozik, amelyek megoldása nehéz, az embertől is kellő szakértelmet, kreativitást, intuíciót – azaz intelligenciát igényelnek. Ilyen feladat például a dámajáték vagy a sakk, a bűvös kocka kirakása, a matematikai tételbizonyítás és az orvosi diagnózis.

A mesterséges intelligencia feladatok az alábbi közös vonásokkal jellemezhetők:

- A feladat megoldása nehéz még az ember számára is, azonban ma még általában az ember a jobb.
- Nem rendelkeznek minden részletében tisztázott, fix megoldó mechanizmussal.
- A megoldás elemi tevékenységek sorozataként állítható elő, amely előre nem rögzített, és több lehetséges sorozat közül kell kiválasztani.
- A feladatmegoldás kereséssel történik, amely során szisztematikus próbálkozással választjuk ki a következő „lépést”.
- A probléma tere nagy lehet, ezért az összes lehetőség kipróbálása a kombinatorikus robbanás problémája miatt szisztematikus módon nem lehetséges. A megoldás során irányított keresésre van szükség.
- Emberi szakértelem/ intuíció/ gyakorlati tapasztalat – úgynevezett heurisztikus ismeret – alkalmazásával korlátozható a keresés. (A heurisztikával vezérelt keresés a mesterséges intelligencia rendszerek legjellegzetesebb közös vonása.)
- „Elég kedvező” megoldás elégséges.

A mesterséges intelligencia feladatok megoldása legtöbbször arra az alapfeladatra vezethető vissza, amely során egy irányított gráfban keressük az egyik csúcsból egy másik csúcsba vezető utat. A látszólag különböző megoldási módszerek ellenére a feladatok végrehajtása, megoldása azonos algoritmus sémára vezethető vissza – ez az általános gráfkereső algoritmus, amelyhez különböző vezérlési stratégiák rendelésével kapjuk meg a konkrét gráfkereső technikákat.

3.2. Kereső ágens

A keresési feladat problémakitűzése a következő:

- Adott:
  - a kezdeti állapot(ok),
  - műveletek/akciók halmaza,
  - célállapot vagy célteszt.
- Meghatározandó:
  - egy a kezdeti állapotból egy célállapotba vezető út (akciósorozat).

A kezdeti állapot és a lehetséges műveletek impliciten definiálják a feladat problématerét (állapotterét). A **kereső ágens** az állapottér felépítésével és szisztematikus bejárásával, módszeres próbálkozással oldja meg a feladatot. Ennek a folyamatnak egy elemi lépése az úgynevezett kiterjesztés, amely során a kereső ágens egy adott állapotból műveletek alkalmazásával újabb állapotokat (a kiindulási állapot gyermekeit) állít elő. A problémamegoldás lényeges kérdése, hogy a kiterjeszthető állapotok közül melyiket érdemes választani. Ennek megválaszolása különböző keresési technikákat eredményez, amelyek közül a legfontosabbakat a következő fejezetekben mutatjuk be.

Az állapotteret irányított gráffal szemléltethetjük, amelynek csúcsai (csomópontjai) az állapotokat – beleértve a kezdeti- és célállapotokat –, élei a műveleteket reprezentálják. Amennyiben a keresés során előállított állapotok mindegyikét új állapotként vesszük figyelembe, fa struktúrát kapunk. A továbbiakban keresőfák bejárásával foglalkozunk.

A keresőfák bejárására használt **általános keresési algoritmus** lépései a következők:

1. Legyen L a kezdeti állapoto(ka)t tartalmazó lista.
2. Ha L üres, akkor leállás – a keresés sikertelen;
   egyébként legyen n egy csomópont L-ből.
3. Ha n célállapot, akkor leállás – eredmény megadása;
   egyébként
   - n törlése L-ből,
   - n gyermekeinek előállítása,
   - n gyermekeinek hozzáadása L-hez,
   - visszalépés 2-re.

A bemutatott algoritmus szisztematikus módon állítja elő a keresőfát. Az algoritmusban szereplő L lista a kiterjeszthető csúcsokat, a keresés frontját tartalmazza. Ezt *nyílt csúcsok halmazá*nak is nevezik. Az algoritmus alkalmazásának legfontosabb kérdése, hogy a 2. lépésben melyik nyílt csúcsot válasszuk. Ahogy ezt már korábban is jeleztük, ennek megválaszolása a konkrét keresési technikáknál jelenik meg.

A keresés hatékonyságát az alábbi szempontok szerint mérhetjük:

- talál-e megoldást?
- a talált megoldás jó megoldás-e? (például alacsony útköltségű)
- keresési eljárás költsége (idő- és memóriaigény).

Általában megállapítható, hogy a keresés költségének meghatározásakor figyelembe kell vennünk mind a megoldás (út) költségét, mind a keresési eljárás költségét.

3.3. Neminformált/vak keresések

A **neminformált** vagy **vak keresés**ek nem tartalmaznak heurisztikus ismereteket. A kereső ágens képessége, hogy szisztematikus módon bejárja a keresőfát és felismeri, ha célállapotba jut.

3.3.1. Szélességi keresés

Szélességi keresés során a keresőfában mindig a „legmagasabb szinten" lévő csomópontok valamelyikét terjesztjük ki. Az algoritmust az általános keresési algoritmus (lásd 3.2. fejezet) következő módosításával írhatjuk le:

- n az **első** csomópont L-ből (2. lépés)
- n gyermekeinek hozzáadása L **végéhez** (3. lépés)

azaz a nyílt csúcsok halmazának kezelésekor mindig az első elemet választjuk és a gyermekeket a lista végére tesszük.

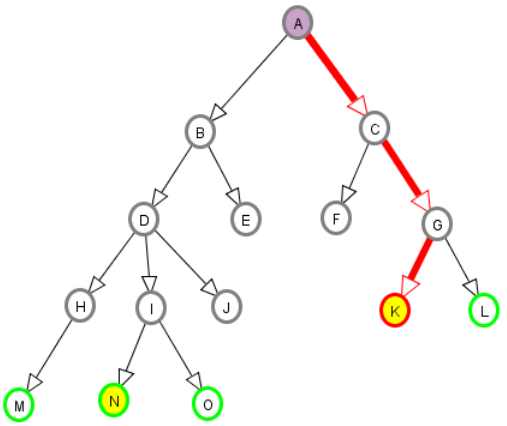
A szélességi keresésre példa a 3.1. ábrán látható, a keresés lépéseit részletesen a `szelessegi.avi` fájl mutatja be (a fájl az AI Space Graph Searching http://aispace.org/search/ honlapon található segédprogrammal készült).

Az algoritmus tulajdonsága, hogy amennyiben létezik megoldás, azt megtalálja (azaz teljes), a legjobb megoldást (a legkevesebb akcióból álló megoldási utak egyikét) találja meg (azaz optimális), memóriaigénye és időigénye az elágazások számában exponenciális (azaz b^d, ahol b az elágazások/akciók száma, d a megoldás mélysége).

3.3.2. Mélységi keresés

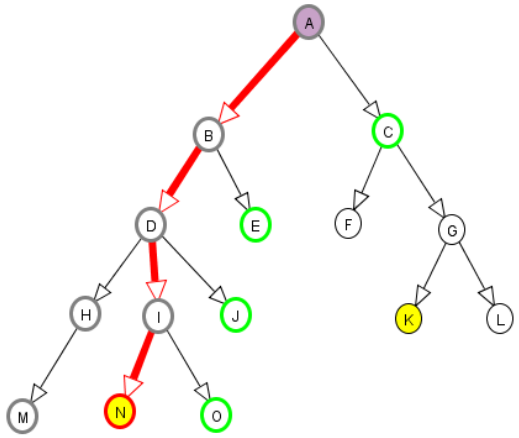
Mélységi keresésnél a keresőfa „legmélyebb szintjén" lévő csúcsok egyikét terjesztjük ki. Az általános keresési algoritmus (lásd 3.2. fejezet) módosítása ebben az esetben a következő:

- n az **első** csomópont L-ből (2. lépés)
- n gyermekeinek hozzáadása L **elejéhez** (3. lépés)



L (nyílt csúcsok)
A
B C
C D E
D E F G
E F G H I J
F G H I J
G H I J
H I J K L
I J K L M
J K L M N O
K L M N O

3.1. ábra. A szélességi keresés lépései



L (nyílt csúcsok)
A
B C
D E C
H I J E C
M I J E C
I J E C
N O J E C

3.2. ábra. A mélységi keresés lépései

azaz a nyílt csúcsok halmazának kezelésekor mindig az első elemet választjuk és a gyermekeket a lista elejére tesszük.

A mélységi keresésre példa a 3.2. ábrán látható, a keresés lépéseit részletesen a `melysegi.avi` fájl mutatja be (a fájl az AI Space Graph Searching http://aispace.org/search/ honlapon található segédprogrammal készült).

Az algoritmus nem teljes (mivel végtelen ág lehetséges), nem optimális, memóriaigénye arányos a megoldás méretével és az akciók/elágazások számával (azaz $b*d$), időigénye az elágazások számában exponenciális (azaz b^d).

3.3.3. Egyenletes keresés

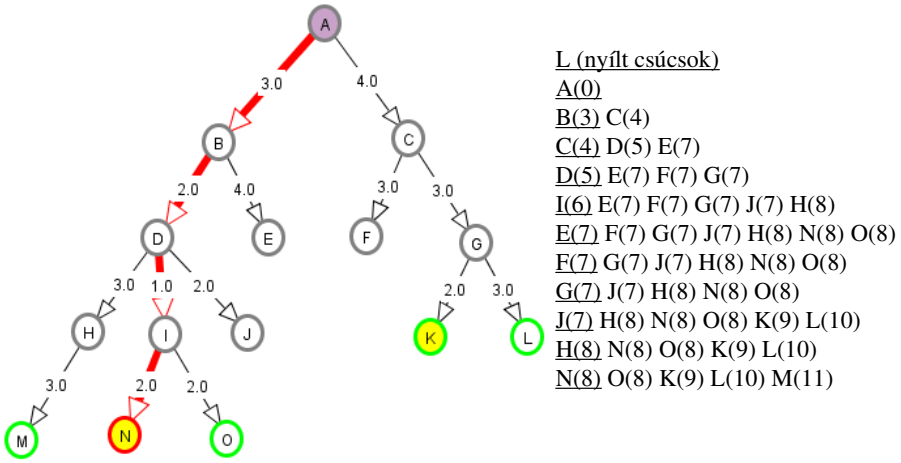
Egyenletes keresés során a csomópontokhoz hozzárendeljük az adott állapotba vezető út költségét, s ennek figyelembe vételével a keresési fában mindig a pillanatnyilag legkisebb költségű csomópontok valamelyikét terjesztjük ki.

Az általános keresési algoritmus módosítása:

- *n* az **első** csomópont *L*-ből (2. lépés)
- *n* gyermekeinek hozzáadása *L*-hez, majd *L* **rendezése a csomópontok növekvő költsége szerint** (3. lépés)

ahol a költség a kezdeti állapotból az adott csomópontba vezető út költsége.

Az egyenletes keresésre példa a 3.3. ábrán látható, a keresés lépéseit részletesen az `egyenletes.avi` fájl mutatja be (a fájl az AI Space Graph Searching http://aispace.org/search/ honlapon található segédprogrammal készült).



3.3. ábra. Az egyenletes keresés lépései

Az algoritmus speciális változata a szélességi keresés (amennyiben az élek költségét egységnyinek tekintjük). Tulajdonságai a szélességi keresés tulajdonságaihoz hasonlóak.

3.4. Informált/heurisztikus keresések

A vak keresési algoritmusok hatékonyságának javítása, a feladatmegoldások számításigényének csökkentése megvalósítható a feladathoz kapcsolódó információ, heurisztika figyelembe vételével.

Heurisztika lehet bármely tanács, amely gyakran hatékony, azonban nem biztos hogy minden esetben az. A heurisztika technikailag megvalósítható egy kiértékelő függvénnyel, amely a probléma egy állapotához rendelt szám, amely legtöbbször a célállapot eléréséig hátralevő út költségét becsli.

3.4.1. Hegymászó keresés

A **hegymászó keresés** során egy csomópont közvetlen leszármazottait vizsgáljuk, és ezek közül mindig a legjobbat választjuk.

Algoritmusa a következő:

1. Legyen *n* a kezdeti állapot.
2. Ha *n* egy célállapot, akkor leállás – eredmény megadása;
3. egyébként
   - *n* valamennyi *n'* leszármazottjának előállítása;
   - legyen *n* a legjobb *n'*;
   - visszalépés 2-re.

Az algoritmus nem tárolja a kereső fát, csak a pillanatnyilag vizsgált csomópontot és ennek gyermekeit – így memóriaigénye minimális. Sikere azonban nagyban függ a bejárt felület alakjától. Hátránya, hogy a matematikában ismert gradiens módszerhez hasonlóan lokális szélsőérték keresésre alkalmas, optimális megoldás előállítása nem garantált.

3.4.2. Előretekintő keresés

Az **előretekintő keresés** elve megtalálni egy célállapotot, amilyen gyorsan csak lehetséges. A kiértékelés alapja ebben az esetben egyedül a céltól való távolság becsült értéke, amely alapján mindig a célhoz legközelebbnek tűnő csomópontot terjeszti ki.

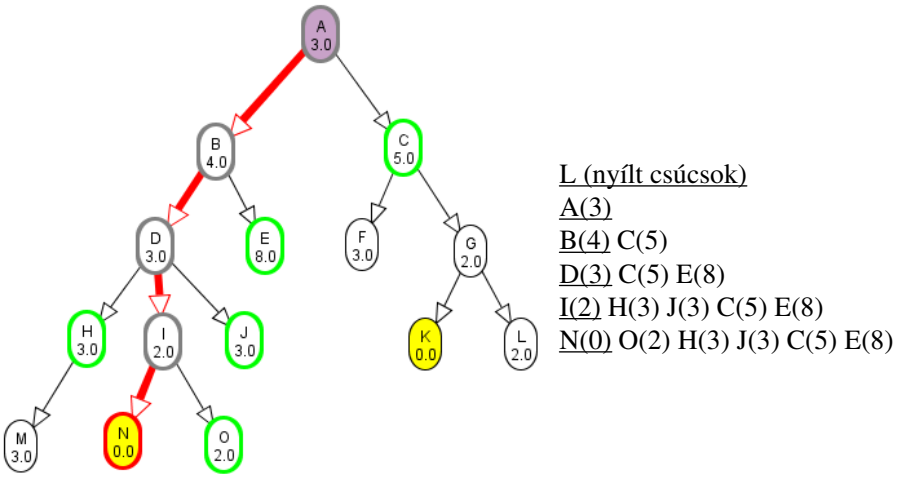
Az előretekintő keresést az általános kereső algoritmus alábbi módosításaként értelmezhetjük:

- *n* az **első** csomópont *L*-ből (2. lépés)
- *n* gyermekeinek hozzáadása *L*-hez, majd *L* **rendezése a csomópontok növekvő költsége szerint** (3. lépés)

ahol a költség az adott csomópontból a célba vezető út becsült költsége.

Az előretekintő keresésre példa a 3.4. ábrán látható, lépéseit részletesen az `eloretekinto.avi` fájl mutatja be (a fájl az AI Space Graph Searching http://aispace.org/search/ honlapon található segédprogrammal készült).

Az algoritmus hatékonysága jelentősen függ a becslő függvénytől, optimális megoldás nem garantált.



3.4. ábra. Az előretekintő keresés lépései

3.4.3. *A* és *A** algoritmus

Az A **algoritmus** az egyenletes keresés és előretekintő keresés előnyös tulajdonságait egyesíti. Az egyenletes keresés a keresés biztonságának megtartását, az előretekintő keresés a kiterjesztések számának csökkentését segíti elő. A kiértékelés alapja a már megtett út (*n* csomópont tényleges távolsága a kezdeti állapottól: *g(n)*) és a még várható út (*n* csomópont becsült távolsága a céltól: *h(n)*) költségösszege (*f(n)* = *g(n)* + *h(n)*).

Az A **algoritmus** az általános kereső algoritmus alábbi módosításával definiálható:

- *n* az **első** csomópont *L*-ből (2. lépés)
- *n* gyermekeinek hozzáadása *L*-hez, majd *L* **rendezése a csomópontok növekvő költsége szerint** (3. lépés)

ahol a költség az adott csomópont *f(n)* értéke, azaz a kifejtésre kerülő csúcs *f(n)* szerint minimális.

Az A^* **algoritmus** olyan A algoritmus, melynek heurisztikus függvénye minden csúcs esetén *alsó becslés*, azaz nem nagyobb, mint a ténylegesen hátralevő út költsége:

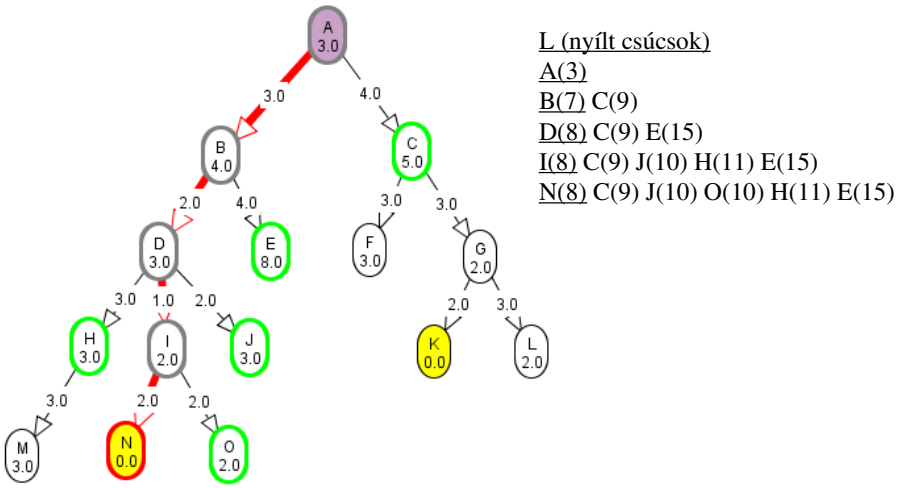
$\forall n : h(n) \leq h^*(n)$.

A heurisztikus függvényre adott alsó becslés kritérium biztosítja, hogy az A^* algoritmus mindig optimális megoldást szolgáltat.

Az A^* algoritmusra példa a 3.5. ábrán látható, a keresés lépéseit részletesen az `acsillag.avi` fájl mutatja be (a fájl az AI Space Graph Searching http://aispace.org/search/ honlapon található segédprogrammal készült).

Amennyiben minden csomópont esetén *h(n)=0*, az egyenletes kereséshez jutunk. Amennyiben emellett az élek költsége egységnyi, a szélességi keresést kapjuk vissza.

(Megjegyzés: A fejezethez tartozó gyakorló feladatok a Keresesek_feladatok.pdf fájlban találhatók.)



3.5. ábra. Az A algoritmus lépései*

4. fejezet

Logikusan gondolkodó ágens

4.1. Alapok

A **logikusan gondolkozó ágens** a világ aktuális állapota ismeretében a világ (általa) még nem ismert tulajdonságairól képes ismeretet előállítani. Ezzel kapcsolatos döntéseit logikai következtetések segítségével végzi.

Ebben a fejezetben bemutatunk egy tényszerű ismereteken alapuló, hatékony következtetést megvalósító tudásreprezentációs technikát, a logikát. Ezen belül megismerkedünk az egyszerű igaz vagy hamis állítások leírására alkalmas nulladrendű logikával vagy más néven ítéletkalkulussal, és bonyolultabb állítások leírására alkalmas elsőrendű logikával vagy más néven predikátumkalkulussal. Mindkét kalkulus esetében bemutatjuk a nyelv szimbólumait (a kifejezéseket, amelyekkel bánni tudunk), s a kifejezésekkel történő építkezés szabályait – azaz a nyelv szintaxisát. Az így felépített mondatokat összekapcsoljuk a világgal, s a mondatoknak jelentést adunk – ezáltal megismerkedünk a nyelv szemantikájával. Végül bemutatjuk azokat a mechanikus eljárásokat, következtetési módszereket, amelyek segítségével az adott szintaxis és szemantika mellett új mondatok, új ismeretek állíthatunk elő.

4.2. Ítéletkalkulus

4.2.1. Szintaxis

Az **ítéletkalkulus jelkészlete** az alábbi elemekből áll:

- *ítéletváltozó*k (logikai változók): például p, q, r, süt a nap, hétfő van, …
- *ítéletkonstans*ok: T, F (igaz és hamis reprezentálására)
- *logikai műveleti jel*ek: $\neg$ (negáció), $\vee$ (vagy), $\wedge$ (és), $\rightarrow$ (implikáció), $\leftrightarrow$ (azonosság)
- *elválasztó jel*ek: például ( ) {}.

A **szintaxis szabályai** a következők:

- *atomi formula* (atom)
  - minden ítéletkonstans atomi formula
  - minden ítéletváltozó atomi formula
- *formula*
  - minden atomi formula egyben formula is
  - ha A és B formulák, akkor $(\neg A), (A \vee B), (A \wedge B), (A \rightarrow B), (A \leftrightarrow B)$ kifejezések is formulák.

A formulaképzés szabálya teljes (azaz segítségével az összes ítéletkalkulusbeli formula előállítható) és definíciójából láthatóan rekurzív.

4.2.1. példa: Tekintsük a következő állításokat, és fogalmazzuk meg ezeket ítéletkalkulusbeli formulákkal:

A_1: Ha a hallgató sokat tanul, akkor jó zárthelyit ír.
A_2: Ha a hallgató jó zárthelyit ír, akkor megajánlott jegyet kap.
A_3: Megajánlott jegy esetén nem kell vizsgázni.
A_4: Ha a hallgató sokat tanul, akkor nem kell vizsgáznia.

Atomok (atomi formulák):
p: a hallgató sokat tanul
q: a hallgató jó zárthelyit ír
r: a hallgató megajánlott jegyet kap
s: a hallgató vizsgázik

Formulák:
$F_1 : p \rightarrow q$
$F_2 : q \rightarrow r$
$F_3 : \neg(s \vee r)$
$F_4 : p \rightarrow \neg s$

A mondatokat leíró fenti formulahalmaz az eredeti állítások szerkezetét tükrözi.□

4.2.2. Szemantika

A formulaképzés szabályaival előállított logikai formula egy szabályos szimbólumsorozat, amelynek igazságértéke ad jelentést. Ezt a **szemantika szabályai** szerint, az alábbi lépésekben határozhatjuk meg:

1. A formula *interpretáció*ja, amely minden ítéletváltozóhoz igaz (T) vagy hamis (F) érték rendelését jelenti minden lehetséges módon.
2. Az interpretált formula *kiértékelés*e a műveleti jelek szemantikája (igazságtáblák, lásd 4.1. táblázat) alapján.

4.1. táblázat. A műveleti jelek szemantikája

p	q	$\neg p$	$p \vee q$	$p \wedge q$	$p \rightarrow q$	$p \leftrightarrow q$
T	T	F	T	T	T	T
T	F	F	T	F	F	F
F	T	T	T	F	T	F
F	F	T	F	F	T	T

A formulákat értékük alapján **formulaosztályok**ba sorolhatjuk:

- Egy formula *kielégíthető*, ha van olyan interpretációja, amelyben igaz az értéke.
- Egy formula *érvényes (tautológia)*, ha minden interpretációban igaz.
- Egy formula *kielégíthetetlen*, ha minden interpretációban hamis.

A formulaosztályok között az alábbi kapcsolatok értelmezhetők:

- ha *x* formula érvényes, akkor $\neg x$ kielégíthetetlen (és fordítva)
- ha *x* formula érvényes, akkor *x* kielégíthető (fordítva nem igaz).

Két formulát ekvivalensnek tekintünk, ha minden interpretációban ugyanaz a logikai értékük.

Néhány nevezetes ekvivalencia, logikai törvény:

A $\leftrightarrow$ B = (A $\rightarrow$ B) $\wedge$ (B $\rightarrow$ A)
A $\rightarrow$ B = $\neg$ A $\vee$ B
A $\vee$ B = B $\vee$ A
A $\wedge$ B = B $\wedge$ A
(A $\vee$ B) $\vee$ C = A $\vee$ (B $\vee$ C)
(A $\wedge$ B) $\wedge$ C = A $\wedge$ (B $\wedge$ C)
(A $\vee$ B) $\wedge$ C = (A $\vee$ B) $\wedge$ (A $\vee$ C)
(A $\wedge$ B) $\vee$ C = (A $\wedge$ B) $\vee$ (A $\wedge$ C)
(A $\vee$ F) = A
(A $\wedge$ T) = A
(A $\vee$ T) = T
(A $\wedge$ F) = F
(A $\vee$ $\neg$ A) = T
(A $\wedge$ $\neg$ A) = F
$\neg$ ($\neg$ A) = A
$\neg$ (A $\vee$ B) = $\neg$ A $\wedge$ $\neg$ B
$\neg$ (A $\wedge$ B) = $\neg$ A $\vee$ $\neg$ B
A $\wedge$ (A $\vee$ B) = A
A $\vee$ (A $\wedge$ B) = A
A $\vee$ A = A
A $\wedge$ A = A

4.2.3. Következtetés

A **következtető ágens** fő képessége, hogy ismeretei és adott szabályai segítségével további ismereteket állít elő (azaz következtet).

A következtetés mechanizmusának megértéséhez ismerkedjünk meg a **logikai következmény** fogalmával: Egy A formulának logikai következménye egy W formula akkor és csakis akkor, ha W igaz minden olyan interpretációban, amelyben A igaz.
Jelölése: $A \vDash W$ (A-nak logikai következménye W)

Példa: Vizsgáljuk meg a logikai következmény definíciója segítségével, hogy az $a \wedge (a \rightarrow b)$ formulának logikai következménye-e b (azaz $a \wedge (a \rightarrow b) \vDash b$)!

a	b	$a \wedge (a \rightarrow b)$
T	**T**	**T**
T	F	F
F	T	F
F	F	F

Az igazságtáblából látható, hogy $a \wedge (a \rightarrow b)$ formula egy esetben igaz, amikor *a* és *b* egyaránt igaz. Ebben az esetben b igaz, ezért a logikai következmény definíciója szerint beláttuk, hogy b logikai következménye $a \wedge (a \rightarrow b)$ formulának.□

Mesterséges intelligencia feladatoknál a logikai következmény fogalmát a következőképpen használhatjuk:

$A_1, A_2, \ldots, A_n \vDash W$.

Azaz bizonyos formulákról ($A_1, A_2, \ldots, A_n$) tudjuk, hogy igazak. A logikai következmény definíciója szerint, ha W ezek logikai következménye, akkor W is igaz (tehát W-re következtethetünk). Kérdés az, hogy ezt az összes eset végignézése nélkül hogyan lehet eldönteni.

Ennek megválaszolására a logikai következmény fogalmát értelmezhetjük az érvényesség és a kielégíthetetlenség fogalmával a következőképpen:

A logikai következmény fogalmának értelmezése az **érvényesség** fogalmával:
$A_1, A_2, \ldots, A_n \vDash W$ iff $(A_1 \wedge A_2 \wedge \ldots \wedge A_n) \rightarrow W$ érvényes.
Azaz W formula akkor és csakis akkor logikai következménye $A_1, A_2, \ldots, A_n$ formuláknak, amennyiben $(A_1 \wedge A_2 \wedge \ldots \wedge A_n) \rightarrow W$ formula érvényes.

A logikai következmény fogalmának értelmezése a **kielégíthetetlenség** fogalmával:
$A_1, A_2, \ldots, A_n \vDash W$ iff $A_1 \wedge A_2 \wedge \ldots \wedge A_n \wedge \neg W$ kielégíthetetlen.
Azaz W formula akkor és csakis akkor logikai következménye $A_1, A_2, \ldots, A_n$ formuláknak, amennyiben $A_1 \wedge A_2 \wedge \ldots \wedge A_n \wedge \neg W$ formula kielégíthetetlen.

Az $(A_1 \wedge A_2 \wedge \ldots \wedge A_n) \rightarrow W$ formulát gyakran *tétel*nek, $A_1 \wedge A_2 \wedge \ldots \wedge A_n$-t a *tétel axiómái*nak, feltételeinek, W-t *következmény*nek, konklúziónak nevezzük.

A továbbiakban a tételbizonyítás módszereit mutatjuk be.

Tételbizonyítás igazságtáblával

Az igazságtáblával történő tételbizonyítás esetén a logikai következmény fogalmát, vagy annak értelmezését az érvényesség illetve a kielégíthetetlenség fogalmával használjuk. A tételbizonyítás első lépésében az igazságtábla szemantikája alapján a formulákat minden interpretációban kiértékeljük, majd a megfelelő definíciók szerint a kapott eredményt értelmezzük.

Példa: Tekintsük a következő formulákat:

$F_1 : p \rightarrow q$

$F_2 : \neg q$

$F_3 : \neg p$

Kérdés: F_1 és F_2 formulák logikai következménye-e F_3 formula ($F_1 \wedge F_2 \models F_3$)?

A példához kapcsolódó igazságtábla a következő:

p	q	F_1	F_2	$F_1 \wedge F_2$	F_3	$F_1 \wedge F_2 \rightarrow F_3$	$F_1 \wedge F_2 \wedge \neg F_3$
T	T	T	F	F	F	**T**	**F**
T	F	F	T	F	F	**T**	**F**
F	T	T	F	F	T	**T**	**F**
F	F	T	T	**T**	**T**	**T**	**F**

Lehetőségek a tételbizonyításra:

- Beláthatjuk, hogy minden olyan interpretációban, amelyben F_1 és F_2 igaz, igaz F_3 is (logikai következmény definíciója alapján). Az igazságtáblából látható, hogy $F_1 \wedge F_2$ egy esetben igaz, s ekkor igaz F_3 is.

- Bebizonyíthatjuk, hogy $F_1 \wedge F_2 \rightarrow F_3$ érvényes (logikai következmény értelmezése az érvényesség fogalmával). Az igazságtábla 7. oszlopában látható, hogy a formula minden interpretációban igaz, azaz érvényes.

- Beláthatjuk, hogy $F_1 \wedge F_2 \wedge \neg F_3$ kielégíthetetlen (logikai következmény értelmezése a kielégíthetetlenség fogalmával). Az igazságtábla utolsó oszlopában látható, hogy a formula minden interpretációban hamis, azaz kielégíthetetlen.□

Az igazságtábla módszerrel történő tételbizonyítás hátránya, hogy idő- és memóriaigénye az ítéletváltozók számában exponenciális (az igazságtábla sorainak száma 2^n, ahol n az ítéletváltozók száma).

Tételbizonyítás formális levezetéssel

A formális levezetéssel történő tételbizonyítás lényege, hogy egy axiómahalmazból (egyszerű, érvényes formulákból) kiindulva levezetési (következtetési) szabályok alkalmazásával próbáljuk előállítani az igazolandó állítást (tételt).

A **levezetési szabály**ok formulákból új formulák előállítását végzik. A legismertebb, gyakran alkalmazott levezetési szabály a **modus ponens**, amely A és $A \to B$ formulákból B formulát állítja elő, azaz:

$$\frac{\begin{array}{l}\mathrm{A}\\ \mathrm{A} \to \mathrm{B}\end{array}}{\mathrm{B}}$$

Egy levezetési szabálytól elvárható, hogy az előállított formula a kiindulási formula (formulahalmaz) logikai következménye legyen. Ebben az esetben a levezetési szabály *helyes*. Igazságtábla módszerrel könnyen belátható, hogy a modus ponens helyes (azaz B logikai következménye A, $A \to B$ formuláknak).

A levezetési szabály másik lényeges jellemzője, hogy képes-e minden logikai következmény formula előállítására, azaz *teljes*-e. A modus ponens nem teljes, azonban az egyszerű használhatósága miatt sokszor alkalmazzák mesterséges intelligencia rendszerekben, pl. szabály alapú szakértői rendszerek (lásd 7. fejezet) következtetési mechanizmusában.

Egy másik levezetési szabály a **rezolúció**, amelynek legegyszerűbb formája a következő:

$$\frac{\begin{array}{l}\mathrm{A} \to \mathrm{B}\\ \mathrm{C} \to \mathrm{A}\end{array}}{\mathrm{C} \to \mathrm{B}} \qquad \text{vagy} \qquad \frac{\begin{array}{l}\neg\,\mathrm{A} \vee \mathrm{B}\\ \neg\,\mathrm{C} \vee \mathrm{A}\end{array}}{\neg\,\mathrm{C} \vee \mathrm{B}}$$

Azaz a rezolúció az $A \to B$ és a $C \to A$ formulákból a $C \to B$ formulát (vagy ezzel ekvivalens módon a $\neg A \vee B$ és a $\neg C \vee A$ formulákból a $\neg C \vee B$ formulát) állítja elő.

A rezolúciós levezetési szabályt a következőképpen általánosíthatjuk:

$$\frac{\begin{array}{l}a_1 \wedge \ldots \wedge a_m \to b_1 \vee \ldots \vee b_k\\ c_1 \wedge \ldots \wedge c_n \to d_1 \vee \ldots \vee d_l \qquad\qquad ahol\ \ d_j = a_i\end{array}}{a_1 \wedge \ldots \wedge \not{a_i} \wedge \ldots \wedge a_m \wedge c_1 \wedge \ldots \wedge c_n \to b_1 \vee \ldots \vee b_k \vee d_1 \vee \ldots \vee \not{d_j} \vee \ldots \vee d_l}$$

vagy

$$\frac{\begin{array}{l}\neg a_1 \vee \ldots \vee \neg a_m \vee b_1 \vee \ldots \vee b_k\\ \neg c_1 \vee \ldots \vee \neg c_n \vee d_1 \vee \ldots \vee d_l \qquad\qquad ahol\ \ d_j = a_i\end{array}}{\neg a_1 \vee \ldots \vee \neg \not{a_i} \vee \ldots \vee \neg a_m \vee \neg c_1 \vee \ldots \vee \neg c_n \vee b_1 \vee \ldots \vee b_k \vee d_1 \vee \ldots \vee \not{d_j} \vee \ldots \vee d_l}$$

Ez utóbbi levezetési szabály speciális formulákat, úgynevezett *konjunktív normálformák*at tartalmaz. A konjunktív normálforma egy olyan formula, amely ítéletváltozók vagy negált ítéletváltozók (úgynevezett literálok) diszjunkcióinak (vagy kapcsolatainak) konjukcióiból (és kapcsolataiból) épül fel. A literálok diszjunkcióját klóznak nevezzük. A logikai törvények, ekvivalenciák alkalmazásával minden formulát átalakíthatunk konjunktív normálformává.

A rezolúciós levezetési szabály helyes és teljes, azaz logikai következményt állít elő, s egyúttal minden logikai következmény előállítására képes.

Tételbizonyítás rezolúcióval

Rezolúcióval történő tételbizonyítás esetén a logikai következmény kielégíthetetlenséggel történő megfogalmazását használjuk fel, azaz a kiindulási formulákból ($A_1, A_2, \ldots, A_n \vDash W$)

és a következmény/célállítás (W) negáltjából konjunkcióval képzett formula ($A_1 \wedge A_2 \wedge \ldots \wedge A_n \wedge \neg W$)kielégíthetetlenségét látjuk be.

A rezolúciós eljárás maga egy **cáfoló eljárás** (ellentmondással történő bizonyítás), amely első lépéseként az előbbi módon képzett konjunktív normálforma klózairól feltételezzük, hogy igazak. A rezolúciós levezetési szabállyal új formulákat állítunk elő. Amennyiben ellentmondáshoz jutunk, a klózhalmaz kielégíthetetlenségét igazoltuk.

4.2.3. példa: Tekintsük a 4.2.1. példában szereplő F_1, F_2, F_3, F_4 formulákat, s igazoljuk rezolúcióval, hogy F_1, F_2, F_3 formulák logikai következménye F_4 (azaz $F_1 \wedge F_2 \wedge F_3 \wedge \neg F_4$ kielégíthetetlen)!

A formulák klózhalmaza a következő:

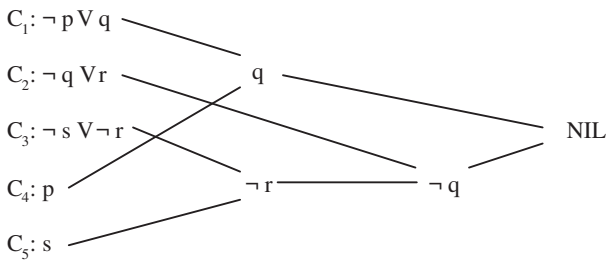
$C_1 : \neg p \vee q$
$C_2 : \neg q \vee r$
$C_3 : \neg s \vee \neg r$
$C_4 : p$
$C_5 : s$

Lássuk be, hogy $C_1 \ldots C_5$ klózhalmaz kielégíthetetlen!

Indirekt bizonyítás (tegyük fel, hogy minden klóz igaz):
C_4 igaz, ha $p = T$
C_5 igaz, ha $s = T$
C_1 igaz, ha ***q* = *T*** (mivel $\neg p = F$, $C4$-ből)
C_3 igaz, ha $r = F$ (mivel $\neg s = F$, $C5$-ből)
C_2 igaz, ha ***q* = *F*** (mivel $r = F$, C_3-ból és C_5-ből) ELLENTMONDÁS!

A bizonyítás lépéseit szemléletesen a 4.1. ábrán levő rezolúciós gráf (más néven cáfolati gráf) mutatja, ahol az ellentmondást az üres klóz (NIL) jelöli. Az új klózok előállítása a rezolúciós levezetési szabály alkalmazásával történt a következőképpen:

4.1. ábra. A bizonyítási eljárás szemléltetése



$$\frac{\neg p \vee q \quad p}{q} \qquad \frac{\neg s \vee \neg r \quad s}{\neg r} \qquad \frac{\neg q \vee r \quad \neg r}{\neg q} \qquad \frac{p \quad \neg p}{NIL}$$

□

A korábban megfogalmazottak alapján az $A_1, A_2, \ldots, A_n \models W$ tételbizonyítás feladata visszavezethető $A_1 \wedge A_2 \wedge \ldots \wedge A_n \wedge \neg W$ formula kielégíthetetlenségének igazolására, amelynek megoldásához a rezolúciós eljárást használtuk.

A **rezolúciós eljárás lépései**:

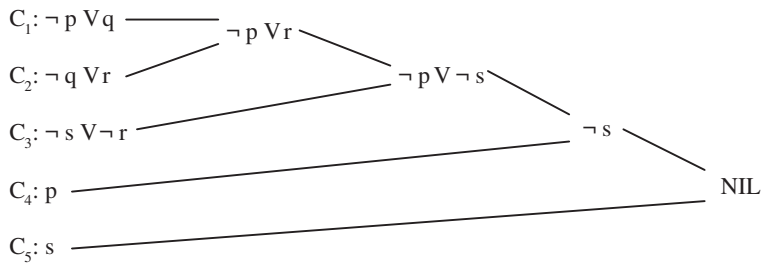
1. Célállítás (W) tagadása, az axiómákhoz való hozzáadása.
2. Az $A_1 \wedge A_2 \wedge \ldots \wedge A_n \wedge \neg W$ formula klóz formára hozása (kiindulási klózhalmaz).
3. Az ellentmondás (üres klóz, NIL) előállításáig:
   - a klózhalmazból két rezolválható klóz választása,
   - a kiválasztott klózok rezolvensének képzése,
   - a rezolvens klóz hozzáadása a klózhalmazhoz.

A rezolválható klózok komplemens literálpárt (egy ítéletváltozót és ennek negáltját) tartalmaznak, a rezolvens klóz pedig a komplemens literálpár elhagyása után maradó részek diszjunkcióval összekapcsolva.

A rezolúciós eljárás tulajdonságai:

- algoritmusa nemdeterminisztikus (a rezolválható klózok választása nem egyértelmű – a 4.2. ábrán látható a 4.2.3. példában szereplő feladat egy másik megoldása),
- helyes (logikai következményt állít elő),
- teljes (minden logikai következmény belátható rezolúcióval).

4.2. ábra. A bizonyítási eljárás szemléltetése – másik megoldás



4.3. Predikátumkalkulus

Az ítéletkalkulus kifejező ereje felépítéséből adódóan nem teszi lehetővé a világ objektumainak, azok tulajdonságainak és kapcsolatainak leírását. A predikátumkalkulus formulái az ítéletkalkulushoz hasonlóan igaz vagy hamis állításokat reprezentálnak, azonban gazdagabb szintaxisa révén alkalmas az előbb megfogalmazott problémák kezelésére.

4.3.1. Szintaxis

A **predikátumkalkulus jelkészlet**ében az ítéletkalkulus jelkészletén túl további elemek is megtalálhatók, ezek összességében a következők:

- *ítéletváltozó*k (logikai változók): például p, q, r, süt a nap, hétfő van, …
- *ítéletkonstans*ok: T, F (igaz és hamis reprezentálására)
- *logikai műveleti jel*ek: $\neg$ (negáció), $\vee$ (vagy), $\wedge$ (és), $\rightarrow$ (implikáció), $\leftrightarrow$ (azonosság)
- *elválasztó jel*ek: például ( ) {}
- *objektumváltozó*k (változók): például x, y, z, …
- *objektumkonstans*ok (konstansok): például a, b, c, …
- *függvényszimbólum*ok: például f, g, h, …
- *predikátumszimbólum*ok: például P, Q, R, …
- *kvantor*ok: $\exists$, $\forall$.

A **szintaxis szabály**ai a következők:

- *term*
  - minden objektumkonstans term
  - minden objektumváltozó term
  - ha f egy n-argumentumú függvényszimbólum és $t_1, \ldots, t_n$ termek, akkor $f(t_1, \ldots, t_n)$ is term
- *atomi formula* (atom)
  - minden ítéletkonstans atomi formula
  - minden ítéletváltozó atomi formula
  - ha P egy n-argumentumú predikátumszimbólum és t_1, …, t_n termek, akkor $P(t_1, \ldots, t_n)$ atomi formula
- *formula*
  - minden atomi formula egyben formula is
  - ha A és B formulák, akkor $(\neg A), (A \vee B), (A \wedge B), (A \rightarrow B), (A \leftrightarrow B)$ kifejezések is formulák
  - ha A egy formula és x egy változó, akkor a $\forall x A$, $\exists x A$ kifejezések is formulák.

A formulaképzés szabályaival úgynevezett jól formált formulák állíthatók elő rekurzív módon.

4.3.1. példa: Tekintsük a következő állításokat és fogalmazzuk meg ezeket predikátumkalkulusbeli formulákkal:

Állítások:
A_1: Van olyan hallgató, aki minden előadáson figyel.
A_2: Unalmas előadáson egyetlen hallgató sem figyel.
A_3: Egyetlen előadás sem unalmas.
(Később feladatunk lesz annak megválaszolása, hogy A_1 és A_2 állításokból következik-e A_3?)

Predikátumok:
$H(x)$: x egy hallgató
$E(y)$: y egy előadás
$U(z)$: z unalmas
$F(x, y)$: x figyel y-on

Formulák:
$F_1 : \exists x H(x) \wedge \forall y[E(y) \rightarrow F(x, y)]$
$F_2 : \forall x \forall z[E(x) \wedge U(x) \wedge H(z)] \rightarrow \neg F(z, x)$
$F_3 : \forall x \neg[E(x) \wedge U(x)]$
□

4.3.2. Szemantika

A jól formált formulának a **szemantika szabályai** szerint adhatunk jelentést a következő lépésekben:

1. a formula *interpretáció*ja
   - az értelmezés alaphalmazának, univerzumának (U nemüres halmaz) megválasztása
   - hozzárendelések:
     - minden konstans szimbólumnak egy elem megfeleltetése U-ból
     - minden *n*-argumentumú függvényszimbólumhoz egy $U^n \rightarrow U$ leképezés rendelése
     - minden *n*-argumentumú predikátumszimbólumnak egy $U^n \rightarrow \{T, F\}$ leképezés megfeleltetése
2. a formula *kiértékelés*e
   - ha A, B formulák igazságértéke ismert, akkor $\neg A, (A \vee B)$, $(A \wedge B), (A \rightarrow B), (A \leftrightarrow B)$ formulák igazságértékének meghatározása az igazságtáblák szemantikája (lásd 4.1. táblázat) alapján

- $\forall x A$ igazságértéke *T*, ha *A* formula értéke minden $x \in U$ esetén *T*, egyébként *F*
- $\exists x A$ igazságértéke *T*, ha *A* formula értéke legalább egy $x \in U$ esetén *T*, egyébként *F*.

A formulákat értékük alapján az ítéletkalkulusnál megismert módon osztályokba sorolhatjuk, így beszélhetünk a minden interpretációban igaz azaz *érvényes*, a minden interpretációban hamis azaz *kielégíthetetlen*, és az igaz interpretációval rendelkező azaz *kielégíthető* formulákról. A problémát predikátumkalkulus esetén a formula összes interpretációban való kiértékelése jelenti, amely a szemantikai szabályainak megfogalmazásából láthatóan sokkal összetettebb és időigényesebb feladat, mint ítéletkalkulus esetében.

4.3.3. Következtetés

Az ítéletkalkulusnál megismert logikai következmény fogalma (valamint megfogalmazása az érvényesség és a kielégíthetetlenség fogalmával) predikátumkalkulus esetén is alkalmazható.

4.3.3. példa: Tekintsük a következő formulákat:
$F_1 : \forall x\{P(x) \rightarrow Q(x)\}$
$F_2 : P(a)$
$F_3 : Q(a)$
ahol *x* objektumváltozó, *a* objektumkonstans, *P* és *Q* predikátumszimbólumok.

Kérdés: F_1 és F_2 formulákból következik-e F_3?

A logikai következmény definíciója alapján a következőképpen gondolkodhatunk:
F_1 igaz minden *x*-re, speciálisan *a*-ra is.
F_2 igaz.
Az implikáció igazságtáblája alapján F_1 és F_2 igazsága esetén F_3 is igaz, tehát F_3 logikai következmény.□

Az ítéletkalkulusnál megismert cáfoló módszert is alkalmazhatjuk (a kielégíthetetlenség fogalmát felhasználva). Ekkor az előbbi példa esetén $F_1 \wedge F_2 \wedge \neg F_3$ formula kielégíthetetlenségét kell igazolnunk. A problémát ebben az esetben és predikátumkalkulusbeli formulák esetében általában az okozhatja, hogy az igazolandó formula kvantorokat, objektumváltozókat, függvényeket tartalmazhat, amelyek kezelésére további technikák (pl. változók standardizálása, Skolemizálás, prenex formára hozás, egyesítés/unifikáció) alkalmazása szükséges. Ezekről az érdeklődő olvasó a [17] és [11] irodalmakban talál részletes leírást.

A 4.3.3. példa folytatása: A $\forall x(P(x) \rightarrow Q(x)) \wedge P(a) \wedge \neg Q(a)$ formula kielégíthetetlenségének igazolásához a formula klózhalmazából indulunk ki, amely a következő:
$C_1 : \neg P(x) \vee Q(x)$
$C_2 : P(a)$
$C_3 : \neg Q(a)$
(Megjegyzés: C_1 klózban minden objektumváltozót univerzális kvantor köt, de ezt nem jelöljük.)

A C_1, C_2, C_3 klózhalmaz kielégíthetetlenségének igazolása indirekt bizonyítással történik:

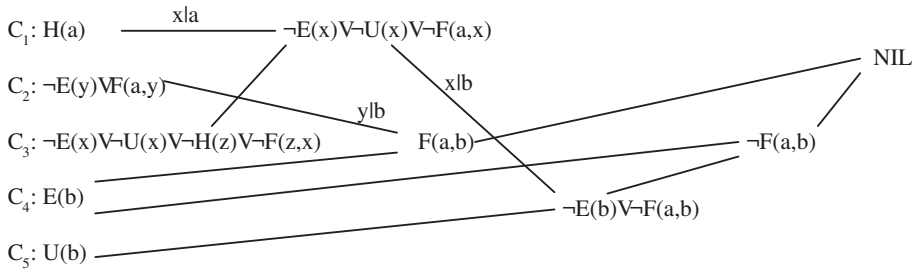
Tegyük fel, hogy minden klóz igaz:
C_2 igaz, ha $P(a) = T$
C_3 igaz, ha $Q(a) = F$
C_1 igaz, ha minden x esetén vagy $P(x) = F$ vagy $Q(x) = T$ (ez ellentmond C_2 és C_3 igazságáról tett megállapításainknak.)□

A 4.3.1. példa folytatása: A formulákból kialakított klózhalmaz a következő:

$C_1 : H(a)$ (F_1-ből, ahol *a* Skolemizálással kapott konstans)
$C_2 : \neg E(y) \vee F(a, y)$ (F_1-ből, ahol *a* Skolemizálással kapott konstans)
$C_3 : \neg E(x) \vee \neg U(x) \vee \neg H(z) \vee \neg F(y, z)$ (F_2-ből)
$C_4 : E(b)$ ($\neg F_3$-ból, ahol *b* Skolemizálással kapott konstans)
$C_5 : U(b)$ ($\neg F_3$-ból, ahol *b* Skolemizálással kapott konstans)

A tételbizonyítás lépéseit, az üres klóz (NIL) levezetését a 4.3. ábrán látható rezolúciós gráf mutatja (a gráf élein az úgynevezett változóillesztések láthatók). Ezzel igazoltuk, hogy az F_3 formula logikai következménye F_1 és F_2 formuláknak.□

4.3. ábra. A bizonyítási eljárás szemléltetése



(Megjegyzés: A fejezethez tartozó gyakorló feladatok a Tetelbizonyitas_az_iteletkalkulusban_feladatok.pdf fájlban találhatók.)

5. fejezet

Bizonytalanságkezelés

5.1. Alapok

Az intelligens problémamegoldás során használt szimbolikus adatok gyakran közelítőek, pontatlanok, bizonytalanok. Ezért rendszereinket olyan technikákkal kell ellátnunk, amelyek segítségével bizonytalan, hiányos, nehezen formalizálható vagy ellentmondásos ismeretek esetén is racionális döntések meghozatalára képesek.

A bizonytalanság oka többféle lehet. Eredhet egyrészt a világ bizonyos részeinek „elrejtéséből”, amelynek eredete lehet hiányos tudás, biztos, de nem megfigyelhető vagy még be nem következett adat. Másrészt tudásunk lehet nem teljesen megbízható akár egy hibás mérési adat vagy bizonytalan fogalom miatt. Problémát okozhat továbbá a nem elég precíz reprezentáló nyelv használata miatti bizonytalan fogalmak, valamint az ellentmondásos tudás kezelése.

A bizonytalanság kezelése felveti azokat a kérdéseket, hogy hogyan reprezentáljuk a bizonytalan információt, hogyan adjuk meg a bizonytalan állítások megbízhatósági mértékét, hogyan kezeljük összetett állítások, kifejezések bizonytalanságát, valamint hogyan következtessünk bizonytalan információ esetén.

A bizonytalanság kezelésére alkalmas módszereket a következőképpen csoportosíthatjuk:

- **Valószínűségszámítási módszerek (numerikus módszerek)**

  Ezek lényege, hogy minden rendszerelemhez (adathoz, fogalomhoz, eljáráshoz) egy, a bizonytalanságot kifejező számértéket rendelnek, az AND, OR és NOT műveletekkel összekapcsolt rendszerelemekhez, valamint a kövezkeztetésekhez kombinációs függvényeket definiálnak.

  – Bayes-modell

    Alapja a klasszikus valószínűségszámítás, amely jól definiált események előfordulásának valószínűségével foglalkozik, ezt számértékkel jellemzi.

  – Bayes-hálók

    Alapja a klasszikus valószínűségszámítás. Az események oksági kapcsolatainak szerkezetét irányított gráffal reprezentálja, majd a valószínűségszámítás konkrét módszereit használja.

  - Fuzzy modell

    Pontatlan, gyengén definiált, úgynevezett fuzzy halmazokkal és az ezekhez kapcsolódó fuzzy logikával foglalkozik.

  - Heurisztikus modellek

    Formailag az előzőekhez hasonlítanak, azonban elméletileg nem megalapozottak. Legismertebb képviselőjük a MYCIN modell (bizonyossági tényező modell), amely egy orvosi szakértői rendszer bizonytalanságkezelő modellje.

- **Nemmonoton logikák (szimbolikus módszerek)**

  Ezek lényege, hogy a hiányos ismereteket feltételezésekkel, hiedelmekkel helyettesítik, amelyek ellentmondás esetén visszavonhatók.

A következő fejezetek a Bayes-modellt, a Bayes-hálót és a fuzzy modellt mutatják be.

5.2. Bayes-modell

A *valószínűségszámítás*i feladatok véletlen kísérletek kimeneteleivel foglalkoznak (pédául mi a valószínűsége, hogy egy dobókockával hatost dobunk). Egy kísérletet elvégezve annak lehetséges kimenetelei az elemi események, amelyek összessége alkotja az eseményteret, amelynek jele az Ω (például a kockadobás eseménytere: $\Omega = \{1, 2, 3, 4, 5, 6\}$). Az eseményeket Ω részhalmazaiként értelmezzük, s minden eseményhez egy nulla és egy közötti számot, az esemény valószínűségét rendelünk.

A **valószínűségi mérték** (P) tulajdonságai a következők:

- $0 \leq P(A) \leq 1$ (egy A esemény valószínűsége 0 és 1 közé esik)
- $P(\Omega) = 1$ (a biztos esemény valószínűsége 1)
- $P(\emptyset) = 0$ (a lehetetlen esemény valószínűsége 0)
- $P(A \cup B) = P(A) + P(B)$ (amennyiben A és B egymást kizáró események)
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$ (amennyiben A és B nem egymást kizáró események).

A **feltételes valószínűség** azt fogalmazza meg, hogy mennyi A esemény bekövetkezésének a valószínűsége abban az esetben, ha tudjuk, hogy B bekövetkezett. Azaz A esemény bekövetkezését mennyire befolyásolja B bekövetkezése (például mi a valószínűsége annak, hogy hatost dobunk a kockával, feltéve, hogy párosat dobunk).

A *feltételes valószínűség* definíciója:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \qquad P(B) > 0$$

Az egyenlet átrendezésével az úgynevezett *szorzatszabály*t kapjuk:

$$P(A \cap B) = P(A|B)P(B)$$

B esemény bekövetkezésének valószínűségét *A* bekövetkezése esetén a feltételes valószínűség definíciója alapján a következőképpen adhatjuk meg:

$$P(B|A) = \frac{P(A \cap B)}{P(A)}, \qquad P(A) > 0$$

Az egyenlet átrendezésével kapott szorzatszabály:

$$P(A \cap B) = P(B|A)P(A)$$

A szorzatszabályok bal oldalainak azonosságát felhasználva a *Bayes-tétel*hez jutunk:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \qquad P(B) > 0$$

Az egyenlet átrendezett alakjából

$$\frac{P(A|B)}{P(A)} = \frac{P(B|A)}{P(B)}$$

látható, hogy a tétel szimmetrikus, azaz *A* és *B* események között nincs oksági irány, akár *A*, akár *B* lehet ok vagy okozat.

Ha egy $\Omega = \{A_1, A_2, \ldots A_n\}$ teljes eseményrendszert vizsgálunk (azaz az események páronként kizárják egymást és uniójuk a biztos esemény), ahol $P(A_i) > 0$ minden *i*-re, akkor bármely további *B* esemény esetén

$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

Ezt felhasználva a *Bayes-tétel általánosítása* a következő:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_{i=1}^{n} P(B|A_i)P(A_i)}$$

A Bayes-tétel segítségével következtetni lehet *B* eseményből A_i esemény feltételes vagy „kikövetkeztetett” vagy a posteriori valószínűségére. Ehhez azonban ismerni kell a $P(A_i)$ valószínűségi értékeket és a $P(B|A_i)$ feltételes valószínűségeket.

Példa: Egy település lakosságának 20%-a gyerek, 30%-a időskorú, 50%-a pedig aktív korú felnőtt. A gyermekek 0,3, az aktív korú felnőttek 0,1, az időskorúak 0,4 valószínűséggel kapják el az influenzát egy nagy járvány idején. Mi a valószínűsége, hogy egy kiválasztott influenzás beteg gyermek? Másképpen megfogalmazva: mi a valószínűsége, hogy a kiválasztott lakos gyermek, feltéve, hogy influenzás? $P(Gy|I)$

I: a kiválasztott lakos influenzás (evidencia)
Gy: a kiválasztott lakos gyermek (evidencia)

A következő adatokat ismerjük:

$P(Gy) = 0{,}2$ (a lakos gyermek)
$P(ID) = 0{,}3$ (a lakos időskorú)
$P(F) = 0{,}5$ (a lakos aktív korú felnőtt)
$P(I|Gy) = 0{,}3$ (a lakos influenzás, feltéve hogy gyermek)
$P(I|ID) = 0{,}4$ (a lakos influenzás, feltéve hogy időskorú)
$P(I|F) = 0{,}1$ (a lakos influenzás, feltéve hogy aktív korú felnőtt)

A Bayes-tétel alkalmazása a feladatra:

$$P(Gy|I) = \frac{P(I|Gy)P(Gy)}{P(I|Gy)P(Gy) + P(I|ID)P(ID) + P(I|F)P(F)}$$

A konkrét valószínűségi értékek behelyettesítésével:

$$P(Gy|I) = \frac{0{,}3 * 0{,}2}{0{,}3 * 0{,}2 + 0{,}4 * 0{,}3 + 0{,}1 * 0{,}5} = 0{,}26$$

Tehát annak valószínűsége, hogy a lakos gyermek, feltéve, hogy influenzás: $P(Gy|I) = 0{,}26$.□

A Bayes-modell alkalmazásának előnye, hogy szilárd elméleti alapokon (valószínűségszámítás) nyugszik, s jól definiált szemantikával rendelkezik. Hátránya, hogy nagy mennyiségű háttér ismeretet (a priori valószínűségek, feltételes valószínűségek) kell megadni a használatához, amelyek közül nem hiányozhat egy sem. Ezeket az a priori valószínűségeket nehéz megadni, meghatározásuk statisztikai mintavételezéssel történik, ami sok munkát, kísérletet jelent. Hátránya továbbá, hogy a tárgyterület változásainak követése nehéz, új bizonyíték, új hipotézis esetén a rendszer bővítése nem inkrementális, valamint a kiszámolt valószínűségek nem magyarázhatóak.

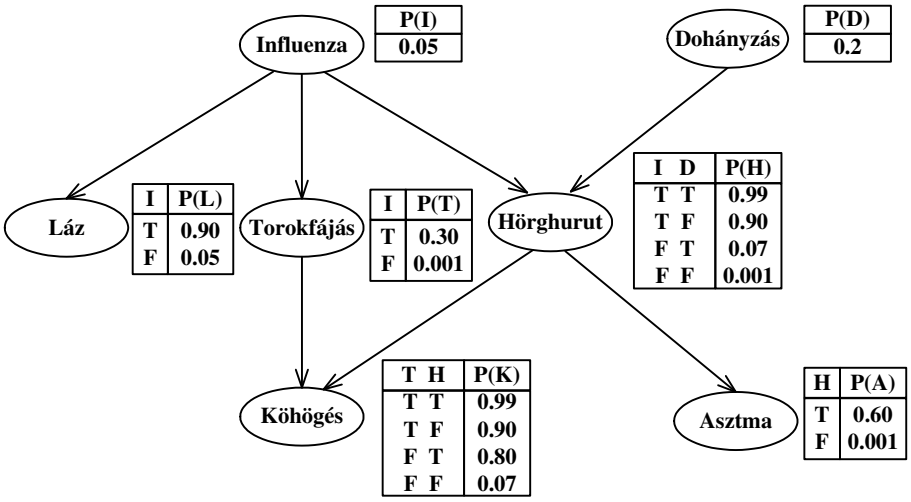
5.3. Bayes-hálók

A statisztikai adatokon alapuló Bayes-modell az oksági kapcsolatok irányáról nem nyújt információt. Az oksági modellek reprezentálására a **Bayes-hálók** (más néven **vélekedéshálók**) használhatók, amelyek egyrészt megjelenítik a valószínűségi paramétereket, másrészt lehetővé teszik a változók között függőségi kapcsolatok ábrázolását.

A Bayes-háló egy *irányított körmentes gráf*, amelynek csomópontjai állításokat, eseményeket reprezentálnak, élei pedig ezek közvetlen ok-okozati kapcsolatait írják le. A háló paraméterei a gráf gyökér csomópontjaihoz rendelt a priori valószínűségi táblák, valamint a nemgyökér csomópontjaihoz rendelt feltételes valószínűségi táblák (lásd az 5.1. ábrát).
A Bayes-hálók a következőképpen használhatók következtetésre:

- hatásokból az okokra történő következtetés (diagnosztizálás)

  pl. a köhögés közvetlen oka lehet hörghurut, amelynek közvetlen oka lehet dohányzás – így a köhögést okozhatja dohányzás



5.1. ábra. Bayes-háló

- hatásokra oksági alapon történő következtetés

  pl. influenza miatt a betegnek fájhat a torka, ami miatt köhöghet

- kölcsönös oksági kapcsolat alapján történő következtetés

  pl. a köhögés oka lehet a torokfájás és a hörghurut

A kikövetkeztetett események valószínűségének meghatározása a Bayes-modell alapján történik. A Bayes-hálók a következtetések mellett használhatók a kapott eredmények indoklására, magyarázatadásra is.

5.4. Fuzzy logika

A mindennapi életben számtalanszor előfordul, hogy nem tudjuk bizonyos dolgokról kategorikusan eldönteni, hogy igazak vagy hamisak, adott halmazba tartoznak-e vagy sem. Próbáljuk meg definiálni például a „magas emberek” halmazát! A klasszikus halmazelmélet szerint ebben az esetben meg kellene adnunk azt a magasság értéket (pl. 180 cm), amelynél kisebb emberek egyike sem eleme a halmaznak, a többiek pedig elemei. Ez a fajta éles szétválasztás azonban a példa esetében erőltetett, hiszen egyrészt a határérték megválasztása szubjektív, másrészt például egy 179,9 cm-es ember már nem tartozna a „magas emberek” halmazába. Ezért az ilyen típusú fogalmak esetében a gyakorlati életben bizonytalanságot kifejező szavakat (pl. többé-kevésbé, talán, lehet) használunk.

A *fuzzy* angol szó magyar jelentése homályos, ködös, életlen, bizonytalan. A folytonos értékkészletű *fuzzy logika* szükségességét és az ehhez kapcsolódó *fuzzy halmaz*ok elméletét, amely a klasszikus halmazelmélet kiterjesztésének tekinthető, Lofti Zadeh fogalmazta meg az 1960-as években. A nyelvi fogalmakban levő bizonytalanság matematikai kezelésére megalkotta a *parciális tagság* fogalmát, amely azt fejezi ki, hogy egy adott objektum mennyire van benne egy adott halmazban.

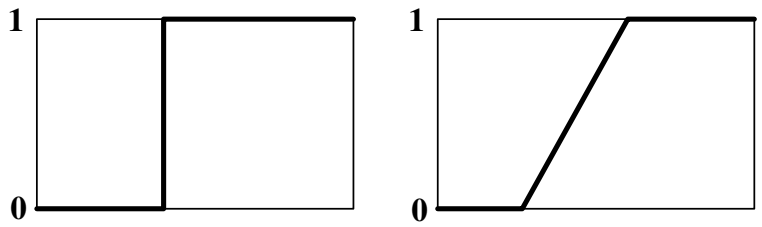
A halmazhoz tartozás mértéke úgynevezett tagsági függvénnyel adható meg a következőképpen:

$\mu_A(x) : U \rightarrow [0, 1]$

ahol:

μ : tagsági függvény
U : alaphalmaz, univerzum
$A \subset U$: fuzzy halmaz
$x \in U$: az alaphalmaz eleme

Az 5.2. ábra egy közönséges (éles) és egy fuzzy halmazt mutat be.



5.2. ábra. Éles és fuzzy halmazok

Amennyiben $\mu_A(x) = 1$: x definit módon A halmazba tartozik, ha $\mu_A(x) = 0$: x definit módon nem tartozik A halmazba. $\mu_A(x_1) > \mu_A(x_2)$ esetén x_1 „jobban beletartozik" A halmazba, mint x_2.

A fuzzy halmazok az 5.3. ábrán látható módon különböző típusú tagsági függvényekkel írhatók le (az ábra a MATLAB® [16] Fuzzy Logic Toolbox segítségével készült). A fuzzy halmazok nemcsak folytonos univerzumon értelmezhetők. Amennyiben az univerzum diszkrét elemekből áll, egy A fuzzy halmaz $\langle x_i, \mu_A(x_i) \rangle$ párokból álló tagsági függvénnyel, úgynevezett singletonokkal adható meg.

A fuzzy halmazokon is értelmezhetők az alapvető halmazelméleti műveletek a következő módon (lásd 5.4. ábra):

Azonos univerzumon értelmezett fuzzy halmazok uniója:

$\mu_{A \cup B}(x) = max\{\mu_A(x), \mu_B(x)\}$

Azonos univerzumon értelmezett fuzzy halmazok metszete:

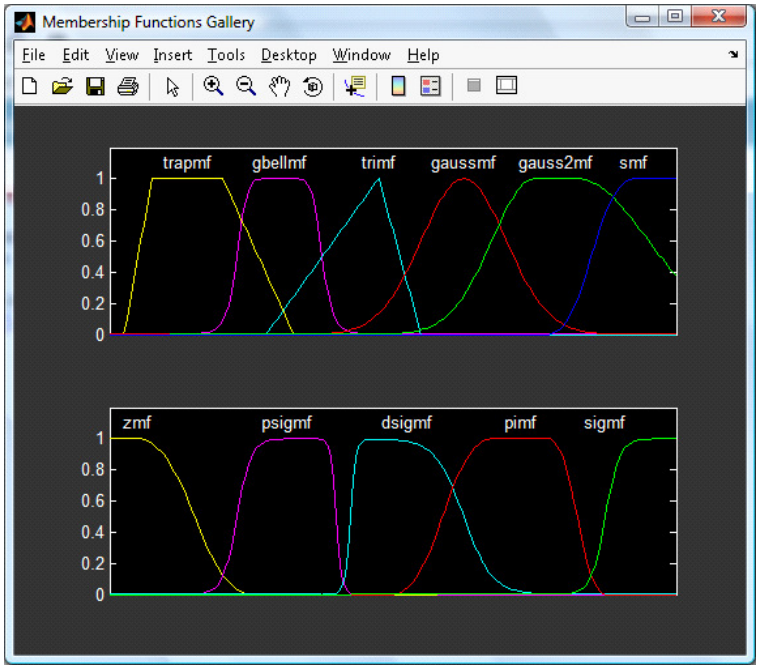
$\mu_{A \cap B}(x) = min\{\mu_A(x), \mu_B(x)\}$

Fuzzy halmaz komplemense:

$\mu_{\neg A}(x) = 1 - \mu_A(x)$

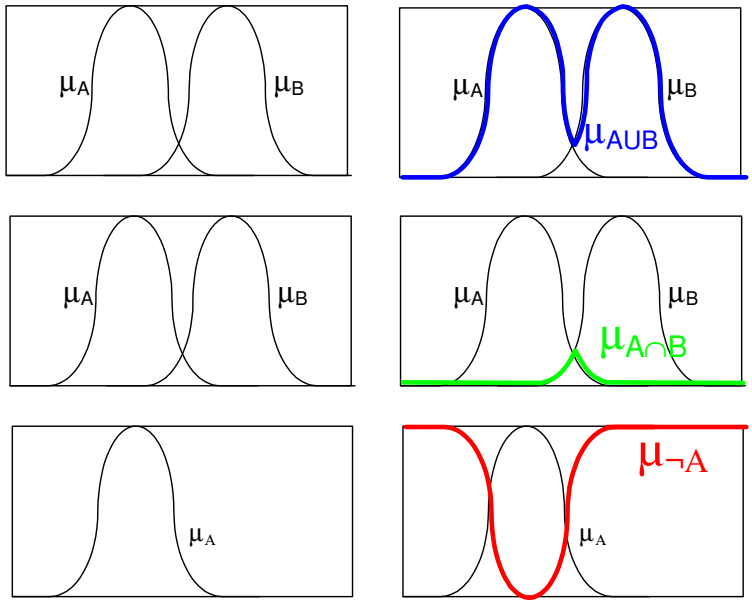
A bemutatott alapműveleteken kívül további műveletek, például fuzzy relációk, fuzzy kompozíciók is értelmezhetők. Az érdeklődő olvasó erről részletesebben a [15] elektronikus tankönyvben olvashat.

A fuzzy halmazok és az ezeken értelmezhető műveletek segítségével bonyolultabb rendszerek is leírhatók, működtethetők. Legfontosabb gyakorlati alkalmazási területei a



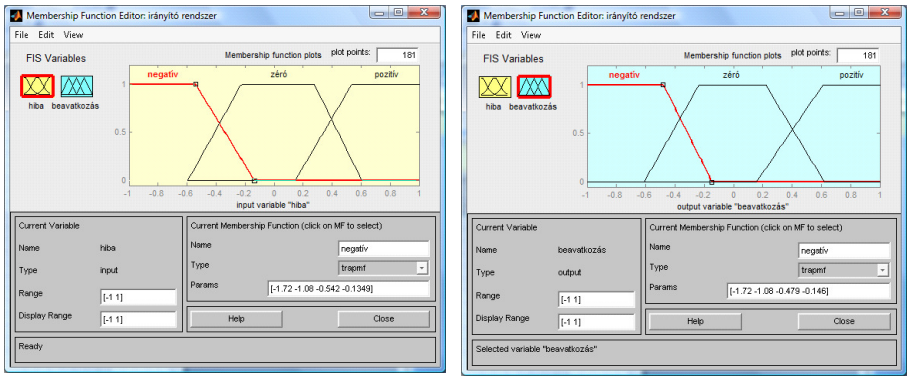
5.3. ábra. Fuzzy tagsági függvény családok

5.4. ábra. Fuzzy halmazokon értelmezett műveletek



fuzzy irányító és szakértői rendszerek (szakértői rendszerekről a 7. fejezetben lesz szó), amelyekben természetes nyelvi változókkal és fuzzy tagsági függvényekkel kombinált „ha-akkor” típusú szabályokkal fogalmazható meg a szakértői ismeret, a gyakorlati tapasztalat.

Példa: Egy irányító rendszerben az alapjeltől való eltérés (hiba) és a beavatkozás univerzumon értelmezzük a „negatív”, a „zéró” és a „pozitív” fuzzy halmazokat. A fuzzy halmazok a MATLAB® [16] Fuzzy Toolbox segítségével készített 5.5. ábrán láthatók.



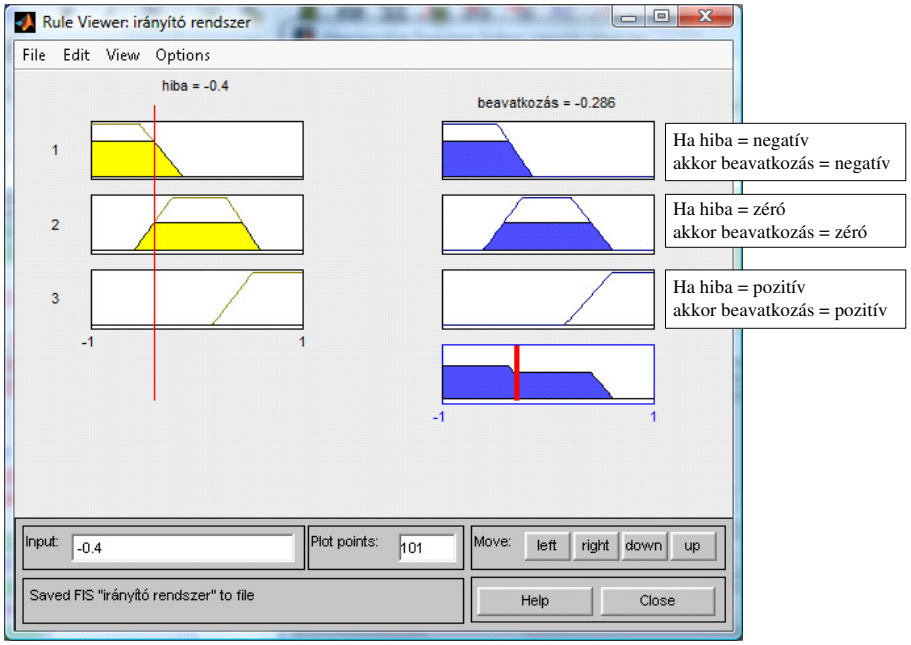
5.5. ábra. Fuzzy halmazok

Legyenek a fuzzy szabályok a következők:

1. Ha hiba = negatív akkor beavatkozás = negatív
2. Ha hiba = zéró akkor beavatkozás = zéró
3. Ha hiba = pozitív akkor beavatkozás = pozitív

A fuzzy következtetés lépései az 5.6. ábrán láthatók. A szabályozott rendszertől érkező bemeneti változó értékéről (hiba = -0,4) meghatározzuk, hogy ez mennyire tartozik a hiba univerzumon értelmezett fuzzy halmazokba. Ez a lépés a fuzzifikálás. Látható, hogy fuzzifikált értéknek az 1. és a 2. szabályok feltételi részével van nemüres metszete, ezért ezek a szabályok alkalmazhatók a következtetés során (lásd ábra bal oldala). Az egyes szabályokkal meghatározott következtetések eredményei az ábra jobb oldalán láthatók, amelynek legalsó részábrája a három szabály általi következtetések unióját mutatja. A következtetés eredményeként, a beavatkozás értékére kapott fuzzy halmaz átalakítása után továbbíthatjuk a szabályozott rendszer felé a beavatkozó jelet. Az átalakítás különböző defuzzifikációs módszerek (pl. súlypont módszer, geometriai középpont módszer) alkalmazásával történhet.

(Megjegyzés: A fejezethez tartozó feladat az FLMatlab.pdf fájlban található.)



5.6. ábra. Fuzzy következtetés

6. fejezet

Tudásalapú rendszerek

6.1. Alapok

A mesterséges intelligencia kutatások első korszakában (60-as évek) kidolgoztak egy általános célú problémamegoldó keretet, a GPS-t (General Problem Solver), amely a feladat adott kezdeti állapotából kiindulva műveletek egymás utáni alkalmazásával próbált meg eljutni a megoldást jelentő célállapothoz. A rendszer szisztematikus módon, a lehetséges utak vakon történő bejárásával (lásd 3. fejezet, keresések) dolgozott, amely nagyméretű, bonyolult feladatok esetén a kombinatorikus robbanás problémája miatt nem lehetett eredményes.

A kutatók felismerték, hogy szűkített feladatosztályok megoldására speciális technikák kifejlesztése szükséges, valamint felismerték a „tudás elvét". Ez utóbbi szerint a feladatmegoldás képessége attól függ, hogy mennyi és milyen minőségű tárgyterület-specifikus ismeret (tudás) áll rendelkezésre. Bebizonyosodott, hogy a nagy méretű és nagy bonyolultságú feladatok jobban kezelhetők, ha magát a problémaleírást és annak összefüggéseit ábrázolják megfelelően, amely alapján a problémamegoldás egyszerű mechanizmusokkal hatékonyan megvalósítható. Így különválik a feladatspecifikus ismeret (tudás) és a végrehajtó mechanizmus (következtetés). Ezeket a rendszereket *tudásalapú* (vagy *ismeretalapú*) *rendszerek*nek (angolul knowledge based system) nevezik.

6.2. Tudásalapú rendszerek jellemzői

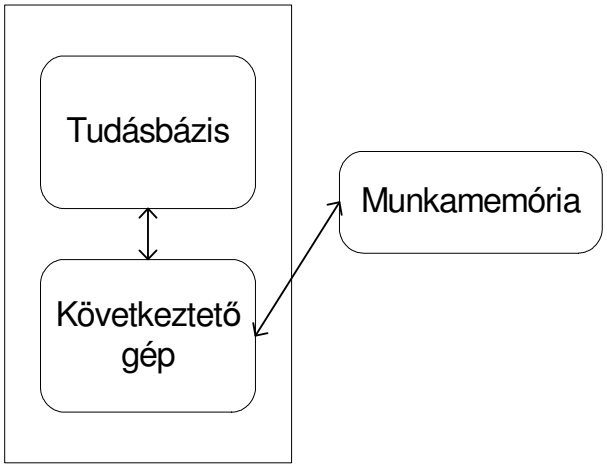
A **tudásalapú rendszer**ek tehát olyan mesterséges intelligencia rendszereknek tekinthetőek, amelyek újszerű programstruktúrával rendelkeznek. Ebben a feladatspecifikus ismeretet tartalmazó *tudásbázis* jól elhatárolódik a rendszer egyéb komponenseitől, így a feladatmegoldást végző *következtető mechanizmus*tól is.

Ennek megfelelően egy tudásalapú rendszer alapvető komponensei a következők (lásd 6.1. ábra):

- *Tudásbázis*, amely általában természetes nyelvhez közeli formalizmussal leírva tartalmazza a problématerületet leíró ismereteket (tudást). Ez tulajdonképpen szimbolikus módon leírt rendszerspecifikáció, amelynek hatékony megvalósításához megfelelő *tudásreprezentációs módszer* szükséges.

- *Következtető gép*, amely a feladatmegoldás „motorja”. Ennek megfelően általános problémamegoldó ismereteket, megoldáskereső stratégiát tartalmaz, amelyet kiegészíthetünk egyéb szolgáltatásokkal is. Fő jellemzője a *megoldáskereső módszer*.
- *Munkamemória*, amely egy kisegítő komponens a konkrét feladat kiinduló, valamint közbülső adatainak tárolására.

Előfordul, hogy a tudásbázis részeként tekintik a munkamemóriát, szerepük alapján azonban érdemes megkülönböztetni az adott típusú problémákra jellemző feladatspecifikus tudást (amely állandó és erősen összefüggő), valamint konkrét esetre vonatkozó tudást (amely a következtetés során, esetleg az időben is változik).



6.1. ábra. A tudásalapú rendszer alapvető komponensei

Egy tudásalapú rendszer intelligens információfeldolgozó rendszerként tekinthető, amelyben a tárgyköri ismeretek ábrázolása természetes nyelvhez közeli formalizmussal, szimbolikus módon történik. A feladatmegoldás a tudásbázisban tárolt ismeretdarabkák mozgósításával, szimbólum-manipulációval történik, ezért ezeket a rendszereket szimbolikus programoknak is nevezik.

6.2.1. Tudásreprezentációs módszerek

A tudásbázis az adott problémára, tárgykörre vonatkozó specifikus ismeretek szimbolikus leírását tartalmazza valamilyen tudásreprezentációs eszközzel megvalósítva. A tudásbázisú rendszerek elkészítésekor feladatunk az ismeret megszerzése, „kinyerése”, az ismeret ábrázolása, reprezentálása, és az ismeret hasznosítása, azaz feladatmegoldásra való felhasználása. Ezen feladatok megvalósításához elengedhetetlenül szükséges egy megfelelő leíró eszköz (szintaxis) és az elemek jelentését meghatározó szemantika (pl. következtető módszer).

A reprezentációs módszereket a következőképpen osztályozhatjuk:

- A problémaleírás módja szerint:

  - procedurális/ algoritmikus reprezentáció

    Ennél a leírási módnál a probléma megoldását vagy annak stratégiáját adjuk meg, azaz arra a kérdésre adjuk meg a választ, hogy HOGYAN oldjuk meg a feladatot.

  - leíró/ deklaratív reprezentáció

    Ekkor magát a problémát írjuk le, azaz, hogy MIT kell megoldanunk. Ebben az esetben a feladatmegoldást a következtető rendszer végzi. Ezeket logika alapú leírásoknak is nevezzük.

  Tudásalapú rendszereknél jellemzően deklaratív leírást alkalmaznak, amely szükség esetén kiegészíthető hatékonyságnövelő algoritmikus elemekkel (pl. démonokkal, metaszabályokkal)

- A problémaleírás szerkezete szerint:

  - egyszerű/ elemi reprezentáció

    Ezek egyszerű, struktúra nélküli elemek, valamint ezek kapcsolatait, a rajtuk elvégezhető műveleteket tartalmazzák. Leggyakrabban logika alapú rendszerek.

  - strukturált reprezentáció

    Ebben az esetben belső struktúrával, attribútumokkal rendelkező objektumokkal dolgozunk, ahol az objektumok kapcsolata lehet taxonomikus vagy osztályozó hierarchia. Gyakran keretalapú rendszerekkel valósítják meg.

6.2.2. Megoldáskereső módszerek

A következtető gép a tudásbázison működő megoldáskereső stratégia implementációja. A keresés a 3. fejezetben bemutatott általános problémamegoldó mechanizmus, amely az állapottér szisztematikus bejárásával oldja meg a feladatot. A lehetséges akciók közötti választás módját meghatározza a keresési stratégia. Ez kiegészíthető a feladatról szóló extra tudással, a heurisztikus ismeretekkel, amelynek figyelembe vétele a keresés hatékonyságát jelentősen növelheti, azonban az esetek többségében jól alkalmazható heurisztikát nem könnyű találni.

A keresési stratégiákat a következőképpen osztályozhatjuk:

- Felhasznált ismeretek alapján

  - Véletlenszerű keresés, amelynél nem biztosított a véges időn belüli célbaérés.
  - Vak keresések (neminformált keresések), amely teljes körű szisztematikus bejárást biztosít, azonban nem ad információt az út/csúcs „jóságáról”. Az algoritmus csupán a cél és nemcél csúcsokat különbözteti meg.
  - Heurisztikus keresések (informált keresések), amely feladatspecifikus heurisztikus ismeretek felhasználásával ad becslés az út/csúcs „jóságáról”.

- Módosíthatóság alapján

  - Nem-módosítható vezérlési stratégiák, amelyek esetén a kiválasztott akció nem vonható vissza, másik alkalmazható akció kipróbálására nincs lehetőség, azaz minden lépés végérvényes.
  - Módosítható vezérlési stratégiák, amelyek felismerik a hibás vagy nem megfelelő akciókat. Az algoritmus egy korábbi állapotba lép vissza, ha olyan végállapotba ér, amely nem célállapot vagy az adott irány nem tűnik ígéretesnek.

6.2.3. Az ismeretalapú rendszerek alaptechnikái

Az ismeretalapú rendszerekben megvalósított tudásreprezentációs módszerek és következtetési és keresési stratégiák szerint az alábbi alaptechnikákat különböztetjük meg:

- Induktív technikák

  Az induktív következtetés a gépi tanulás problémakörébe tartozó módszer, amely során egyedi esetekből (tanulási példákból) általános érvényű következtetések levonása történik. Erről részletesebben a 8. fejezetben esik szó.

- Szabályalapú technikák

  A leggyakrabban alkalmazott tudásalapú rendszer alaptechnika, amely HA-AKKOR szerkezetű szabályok alaján adat- vagy célvezérelt következtetést valósít meg. Ezt a módszert a 6.3. fejezet mutatja be.

- Hibrid technikák

  Többféle alaptechnika együttes használatát biztosítja, leggyakrabban szabályokat és keret-struktúrákat alkalmaz.

- Szimbólum-manipulációs technikák

  A tudásalapú rendszerek működése szimbólum-manipulácival történik, ezért leírásukra a mesterséges intelligencia programnyelvei, pl. LISP, Prolog is használhatóak.

- További következtető technikák

  - modell-alapú következtetési technikák
  - kvalitatív technikák
  - eset-alapú technikák
  - temporális következtetési technikák
  - neurális hálózatok.

6.3. Szabályalapú következtetés

A **szabályalapú reprezentáció** a legkorábban kialakult, s egyszerűsége és hatékony megvalósíthatósága miatt a ma is leggyakrabban alkalmazott tudásreprezentációs módszer. Tényállítások és HA-AKKOR típusú feltételes állítások írhatók le segítségükkel, amely az emberi

gondolkodásmód modellezésére, a szakértői tapasztalatok, heurisztikák megfogalmazására alkalmas.

Egy **szabály** egy ismeretdarabkát reprezentál

HA <feltétel> AKKOR <következmény>
vagy
IF <feltétel> THEN <következmény>

alakban, ahol a *feltétel* a szabály alkalmazásának előfeltételeit, a *következmény* pedig a végrehajtásának hatását adja meg tények (predikátumok) vagy tényekből ÉS/VAGY műveletekkel előállított kifejezések formájában.

Egy **szabályalapú rendszer** működése kétféle következtetési stratégia alapján történhet. Az *adatvezérelt* (vagy *előrefelé haladó*) *következtetés* során feladat egy kezdőállapotból kiindulva egy megoldást jelentő célállapot elérése vagy megkonstruálása. A *célvezérelt* (vagy *visszafelé haladó*) *következtetés* esetén a feladat egy feltételezett célállapot igazolása kezdetben érvényes tényekre támaszkodva.

A szabályokat mindkét esetben a következtető gép működteti új ismeret vagy információ levezetése céljából. Egy elemi következtetési lépés egy szabály alkalmazását jelenti, amely a következő részlépésekből áll:

1. *Mintaillesztés*

   Ebben a részlépésben a következtető gép mintaillesztéssel megkeresi az alkalmazható szabályokat, majd a végrehajtható szabályokat egy úgynevezett konfliktushalmazba teszi.

2. *Szabály kiválasztása/konfliktusfeloldás*

   A következtető gép beépített vezérlési stratégiája alapján a konfliktushalmazban levő szabályok közül választ egyet.

3. *Szabály alkalmazása*

   A következtető gép a kiválasztott szabályt végrehajtja. Ezt a lépést más néven a szabály tüzelésének nevezzük.

A következtető gép ezt 3 részlépésből álló elemi következtetési lépést ismétli ciklikusan a leállási feltétel bekövetkezéséig vagy amíg nincs több alkalmazható szabály.

6.3.1. Adatvezérelt következtetés

Az **adatvezérelt következtetés** lépései a következők:

1. A *mintaillesztés* a munkamemória tartalma és a szabályok feltételi része között történik a modus ponens levezetési szabály alkalmazásával (lásd 4.2.3. fejezet).

2. A *szabály kiválasztása/konfliktusfeloldás* egy tetszőleges konfliktusfeloldó statégia alapján történik.

3. A *szabály alkalmazása* során a kiválasztott szabály következmény részének alkalmazása/igazzá tétele történik.

6.3.1. példa: Tekintsük a következő szabályokat:

R1 *IF A(X) and B(Y) THEN C(X)*
R2 *IF C(X) and D(Y) THEN P(X)*
R3 *IF D(X) and not E THEN P(X)*

A munkamemória tartalma legyen: *A(a), B(b), D(d)*,
ahol *A, B, C, D, E, P* predikátum szimbólumok, *X* és *Y* változók, *a, b* és *d* konstansok.

Az adatvezérelt következtetés lépései a következők:

- A munkamemória tartalmára illeszkedő szabályok: *R1 X=a* és *Y=b* illesztéssel, valamint *R3 X=d* illesztéssel. A végrehajtható szabályok közül *R1*-t alkalmazva a munkamemóriába bekerül *C(a)*.

  A munkamemória tartalma: *A(a), B(b), D(d), C(a)*.

- Az alkalmazható szabályok: *R1 X=a* és *Y=b* illesztéssel, *R2 X=a* és *Y=d* illesztéssel, valamint *R3 X=d* illesztéssel. (A vezérlés ciklusmentesítésére a továbbiakban az ugyanazzal az illesztéssel korábban már alkalmazott szabályokat ismételten nem hajtjuk végre, ezért *R1*-t nem végrehajthatónak tekintjük.) *R2* szabály alkalmazásával a munkamemóriához adjuk *P(a)*-t.

  A munkamemória tartalma: *A(a), B(b), D(d), C(a), P(a)*.

- A végrehajtható szabály: *R3 X=d* illesztéssel. Ezt végrehajtva *P(d)* a munkamemóriába kerül.

  A munkamemória tartalma: *A(a), B(b), D(d), C(a), P(a), P(d)*.

- Nincs több alkalmazható szabály.

Az adatvezérelt következtetés lépéseit a 6.2. ábra illusztrálja.

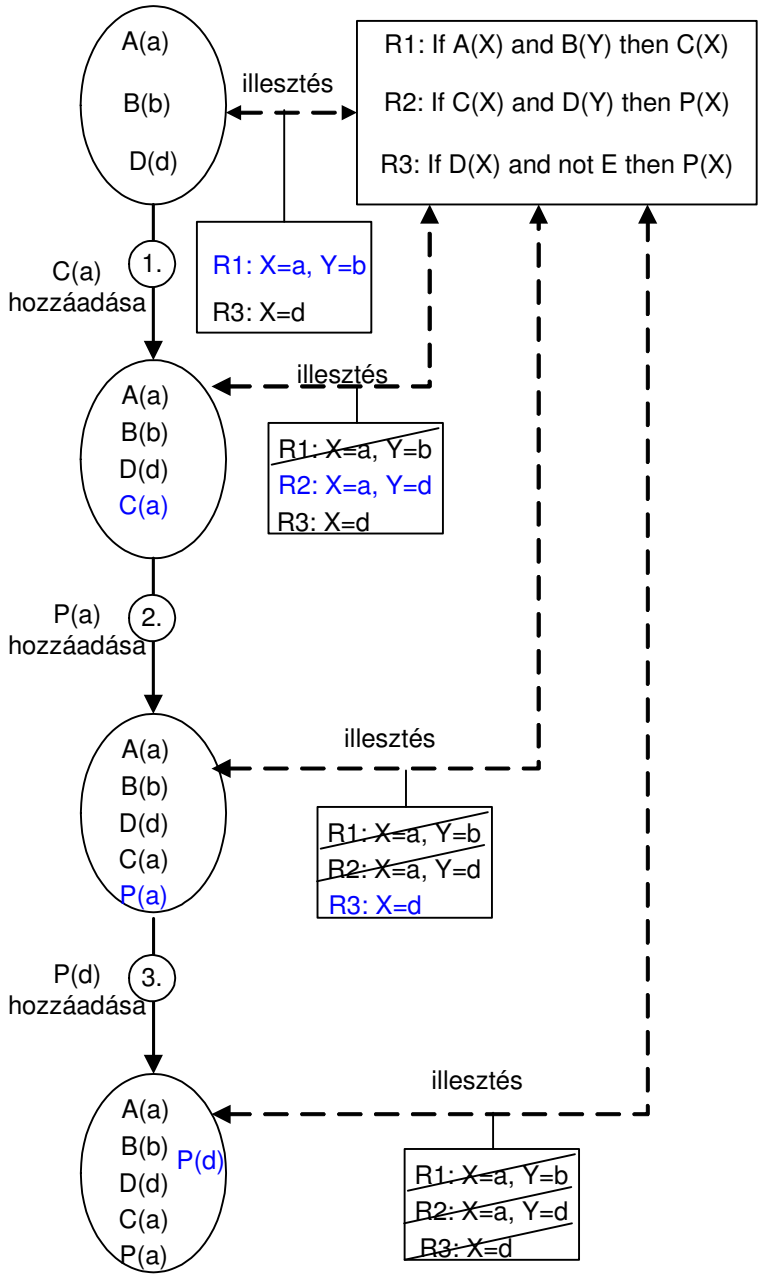
6.3.2. Célvezérelt következtetés

Célvezérelt következtetésnél feladatunk egy célállapot igazolása, amelynek lépései a következők:

1. A *mintaillesztés* egy igazolandó cél és a szabályok következmény része között történik a modus ponens levezetési szabály (lásd 4.2.3. fejezet) „fordított irányú” alkalmazásával. Ez alapján az alkalmazható szabályokat a konfliktushalmazba tesszük.

2. A *szabály kiválasztása/konfliktusfeloldás* az elsőként alkalmazható szabály kiválasztásával történik.

3. A *szabály alkalmazása* során a kiválasztott szabály feltételi részében szereplő állítások lesznek az új részcélok.



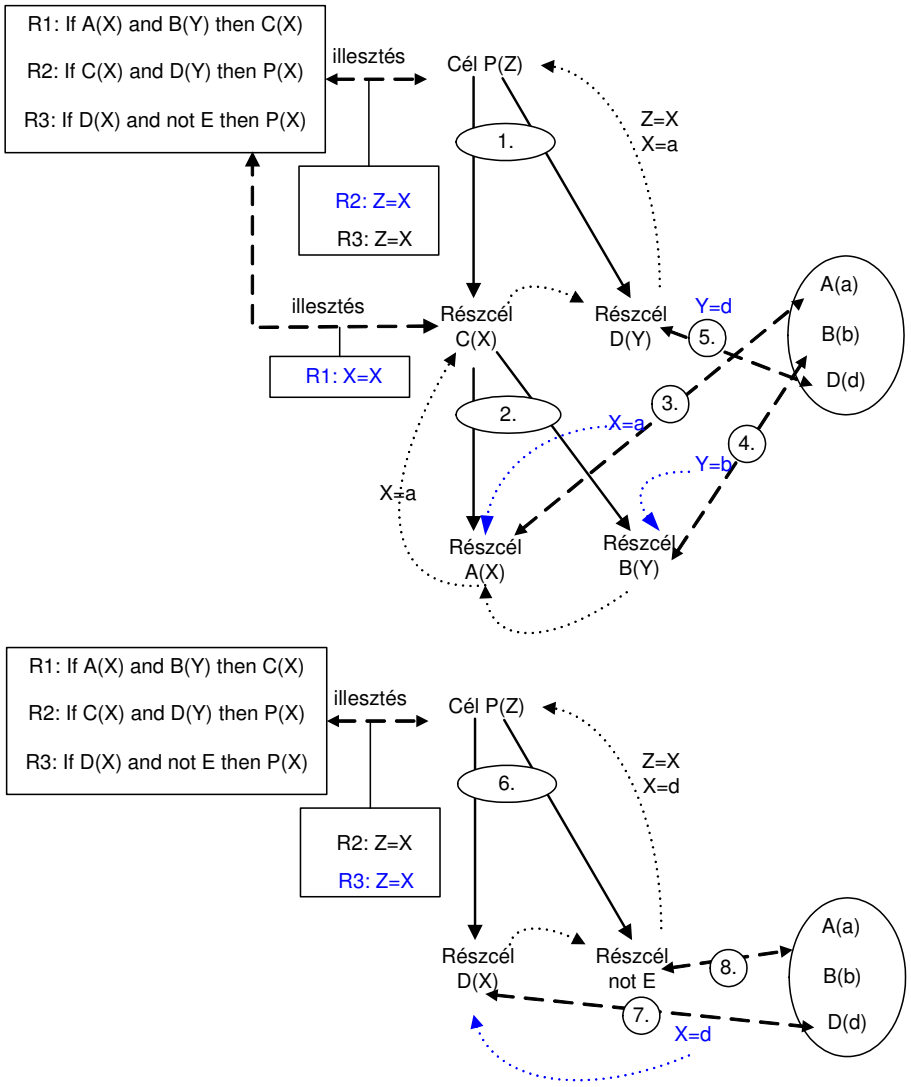
6.2. ábra. Az adatvezérelt következtetés lépései

A következtető gép ezeket a lépéseket ismétli ciklikusan az összes részcél igazolásáig, vagy amíg nincs több alkalmazható szabály. Célvezérelt következtetésnél zsákutca esetén (amennyiben az adott állapotban nincs több alkalmazható szabály), visszalépést alkalmazva lehetőség van új irányok kipróbálására.

6.3.1. példa folytatása: A szabályhalmaz és a munkamemória kezdeti állapotának ismeretében célvezérelt következtetés során feladatunk *P(Z)* igazolása (ahol *Z* változó). Ennek lépései a következők:

- A *P(Z)* célra illeszkedő szabályok: *R2 Z=X* illesztéssel, valamint *R3 Z=X* illesztéssel. A végrehajtható szabályok közül a továbbiakban az első megtalált szabályt alkalmazzuk. *R2* végrehajtásával *C(Z)* és *D(Y)* új részcélok lesznek.
  - *C(X)* részcél igazolására alkalmazható szabály: *R1 X=X* illesztéssel, amelynek alkalmazásával *A(X)* és *B(Y)* lesznek az új részcélok.
    * *A(X)* részcél igaz, mert a munkamemóriában megtalálható *X=a* egyesítéssel.
    * *B(Y)* részcél igaz, mert a munkamemóriában megtalálható *Y=b* egyesítéssel.
    * Mivel igazoltuk *A(a)*-t és *B(b)*-t, igazoltuk *C(a)*-t.
  - *D(Y)* részcél igaz, mert a munkamemóriában megtalálható *Y=d* egyesítéssel.
  - Mivel igazoltuk *C(a)*-t és *D(d)*-t, igazoltuk *P(a)* célt.
- Amennyiben célunk az összes megoldás előállítása, az első lépésre visszalépünk és a *P(Z)* célra illeszkedő szabályok közül *R3*-t alkalmazzuk. Ebben az esetben *D(X)* és *not(E)* lesznek az új részcélok.
  - *D(X)* részcél igaz, mert a munkamemóriában megtalálható *X=d* egyesítéssel.
  - *not(E)* részcél igaz, mert *E* nem szerepel a munkamemóriában.
  - Mivel igazoltuk *D(d)*-t és *not(E)*-t, igazoltuk *P(d)* célt.

A célvezérelt következtetés lépései a 6.3. ábrán láthatóak.



6.3. ábra. A célvezérelt következtetés lépései

7. fejezet

Szakértői ágens

7.1. Alapok

A **szakértői rendszer** olyan szoftver ágens, amely a felhasználó kérdéseire az emberi szakértőhöz hasonlóan javaslatot, tanácsot vagy valamilyen konkrét értékelést szolgáltat. Egy szűk szakterület ismereteit magába foglalva, különböző mesterséges intelligencia technikákat felhasználva az emberi problémamegoldás folyamatát modellezi.

A szakértői rendszerek fejlesztésével és használatával célunk, hogy a szakértő(k) ismereteit tárolva viszonylag gyors problémamegoldást biztosítsunk az emberi tényezőktől független módon. Alkalmazásukkal egy feladatot kevesebb emberrel és/vagy kevesebb idő alatt oldhatunk meg, a szakértői javaslatok figyelembe vételével a hibás döntések száma csökkenthető.

7.2. Szakértői rendszer

A szakértői rendszerek olyan tudásalapú rendszereknek (lásd 6. fejezet) tekinthetők, amelyek szakértői szintű ismeretek felhasználásával magas szintű teljesítményt nyújtanak egy szűk problémakör kezelésében.

Szakértői rendszerek alkalmazása abban az esetben lehetséges, ha a megoldandó feladat az alábbi jellemzőkkel rendelkezik:

- szűk problématerületet ír le
- ennek ellenére elég bonyolult, azaz igény van szakértelemre
- emberi szakértők rendelkezésre állnak
- legalább a szakterület alapkérdéseiben egyetértés van a szakértők között
- tanpéldák, alapadatok rendelkezésre állnak a rendszer elkészítéséhez és teszteléséhez
- előny továbbá, ha a feladat részproblémákra osztható.

A problémamegoldás folyamata az emberi problémamegoldáshoz hasonló módon, a szakértői szintű ismeretek (tudásdarabkák, sémák, ökölszabályok) tudáselemeinek szituációfüggő mozgósításával történik, amelynek eredménye magyarázattal ellátott szakértői szintű javaslat, tanács. Ez alapján a szakértői rendszer egy intelligens információfeldolgozó rendszerként tekinthető.

A szakértői rendszereket alapvetően két csoportba sorolják: lehetnek döntéshozó és döntéstámogató rendszerek. A **döntéshozó rendszer**ek általában a problématerületen kevésbé járatos felhasználónak nyújtanak segítséget egy adott probléma megoldásának megkeresésében. A **döntéstámogató rendszer**ek a felhasználó számára megfontolandó alternatívákat, javaslatokat kínálnak fel és a megoldás elemzéseként ehhez fűződő magyarázataikkal segítik a felhasználót a döntéshozatalban.

Ennek megfelelően egy szakértői rendszertől az elvárásaink (a szakértőtől elvártakhoz hasonló módon) a következők:

- adjon javaslatot
- megadott javaslatait szükség esetén indokolja
- természetes nyelven (kérdés/válasz formájában) kommunikáljon a felhasználóval, legyen „egyenrangú beszélgető partner”
- feltett kérdéseit szükség esetén magyarázza meg
- bizonytalan szituációban is elfogadható javaslatot adjon.

Az itt felsorolt tulajdonságok biztosításához a szakértői rendszer az alábbi komponenseket tartalmazza:

- *Tudásbázis*

  A feldolgozandó problématerület szakértői szintű ismereteit tárolja a tudásdarabkák szimbolikus leírására alkalmas reprezentációs technika felhasználásával.

- *Következtető gép*

  A tudásbázis ismereteit felhasználva, valamint a hiányzó ismereteket a felhasználótól bekérve új ismereteket határoz meg. A következtető rendszer által nyújtott feladatmegoldás eredménye szakvélemény, javaslat vagy értékelés. A problémamegoldáshoz megoldáskereső stratégiával rendelkezik, valamint támogatja a többi komponens működését.

- *Munkamemória*

  Specifikus információt, a külvilágból érkező és onnan kért adatokat, valamint a következtetések során kapott ismereteket tartalmazza.

- *Magyarázó alrendszer*

  A rendszer akcióit, javaslatait magyarázza meg felhasználói kérésre.

- *Tudásbázis kezelő/fejlesztő alrendszer*

  A komponens a tudásbázis építését, tesztelését és módosítását segíti.

- *Felhasználói felület*

  Természetes nyelvű párbeszédet, konzultációt biztosít a felhasználó és a következtető gép között.

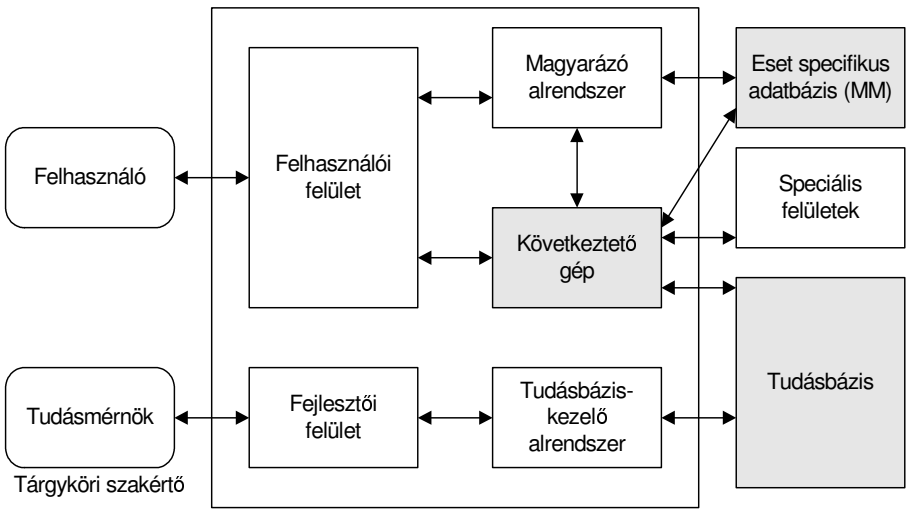
- *Fejlesztői felület*

  A tudásbázis „gazdája”, a tudásmérnök, valamint a tárgyköri szakértő hozzáférését biztosítja a tudásbázis kezelő/fejlesztő alrendszerhez.

- *Speciális felület*ek

  Adatbázisok elérését és egyéb kapcsolatokat (pl. real-time adatszolgáltatást) valósítanak meg.

A szakértő rendszer alapvető komponensei és ezek kapcsolatai a 7.1. ábrán láthatóak.



7.1. ábra. A szakértői rendszer alapvető komponensei

A 6. fejezetben bemutattuk a tudásbázis és a következtető gép fő funkcióit és megvalósításához alkalmazható technikákat. A következő alfejezetekben a szakértői rendszerek magyarázó alrendszerének és tudásbázis kezelő/fejlesztő alrendszerének legfontosabb feladatait foglaljuk össze.

7.2.1. A magyarázó alrendszer

A **magyarázó alrendszer** képessége, hogy feladatmegoldás során felhasználói kérésre feltett kérdéseihez magyarázatot fűz és információt szolgáltat a következtetés aktuális állapotáról, valamint feladatmegoldás után javaslatait indokolja.

A jellegzetes magyarázatadási módok, a felhasználó által kérhető szolgáltatások a következők:

- Magyarázatadás feladatmegoldás közben:
  - WHY …típusú kérések

    A rendszer magyarázó következtetést végez, amely során az aktuális tudáselemet és a hozzá vezető következtetési láncot megmutatva, úgynevezett intelligens súgó-help-ként viselkedik.
  - WHAT IF …típusú kérések

    A felhasználó tesztelheti lehetséges válaszait. Amennyiben számára egy válasz következménye kedvezőtlen, lehetőség van annak visszavonására és új lehetőség kipróbálására. Ez a szolgáltatás hipotetikus következtetést valósít meg.
  - WHAT IS …típusú kérések

    A felhasználó információt kérhet a tudásbázis és a munkamemória állapotáról.
- Magyarázatadás, indoklás a feladatmegoldás után:
  - HOW …típusú kérések

    A felhasználó a rendszer által adott javaslatok elemzéséhez az adott válaszhoz vezető következtetési lépéseket, indoklást kérhet. Ez a magyarázatadási mód visszatekintő következtetést valósít meg.
  - WHY NOT …típusú kérések

    Magyarázó következtetés annak ellenpéldákkal történő megkeresésére, hogy miért nem a felhasználó által kérdezett eredmény valósult meg.
  - WHAT IS …típusú kérések

    Felhasználói információ kérése a tudásbázis és a munkamemória állapotáról.

7.2.2. A tudásbázis kezelő/fejlesztő alrendszer

Egy szakértői rendszer elkészítésének legkritikusabb része az ismeretszerzés és a tudás adott formába öltése, amelynek kulcsszereplője a **tudásmérnök**, aki a **tárgyköri szakértő**vel együttműködve dolgozik. A tudásmérnök feldata a tudás beszerzése, megfelelő reprezentációs technika és következtetési stratégia kiválasztása után a tudásbázis megépítése, annak tesztelése, finomítása, módosítása, valamint ellenőrzése és karbantartása. A tudásbázis elkészítését a **tudásbázis kezelő/fejlesztő alrendszer** támogatja.

A tudásbázis kezelő/fejlesztő alrendszer szolgáltatásai eszközfüggőek, ezek közül néhány tipikus szolgáltatás a következő:

- Eszközök a tudásbázis interaktív fejlesztéséhez:
  - Integrált editor a tudáselemek létrehozásához. Ez általában szöveg editor, amelyet a grafikus editor egészíthet ki.
  - Interaktív szövegorientált fejlesztői környezet, amely a tudáselemek szintaktikai ellenőrzését is elvégzi.

  - A tudásbázis vizsgálata, amely megmutatja a megadott tulajdonsággal rendelkező tudáselemeket.
  - Dokumentáció készítése a tudáselemekhez.
  - Online help.

- Eszközök nagy méretű tudásbázisok építéséhez és a tudásbázis ellenőrzéséhez:
  - Nyomkövetési lehetőségek a tudásbázis működése során (például tulajdonság-értékek megjelenítése, megszakítási pontok generálása).
  - Tudásbázis hierarchikus/moduláris tervezése.
  - Tudásbázis particionálása, tudásbázisok összeépítése.
  - Tudáselemek tulajdonság-értékeinek statikus/dinamikus ellenőrzése.
  - Tudáselemek szemantikai ellenőrzése.

7.3. Szakértői keretrendszer (shell)

Az üres tudásbázissal és erőteljes tudásbázis kezelő/fejlesztő alrendszerrel rendelkező tudásalapú rendszereket **szakértői keretrendszer**eknek vagy **shell**-eknek nevezzük. Fő komponenseit a 7.2. ábra mutatja be. Ezek a rendszerek tárgyterülettől független szolgáltatásokat nyújtanak szakértői rendszerek létrehozásához és működtetéséhez. Általában támogatják a gyors prototípuskészítést és az inkrementális rendszerépítést. Számos ingyenes és kereskedelmi forgalomban levő szakértői keretrendszer érhető el napjainkban, amelyek méreteit tekintve PC-ken, munkaállomáson és többfelhasználós nagyszámítógépeken használhatók.

A szakértői keretrendszereket a következőképpen csoportosíthatjuk:

- Általános célú shellek

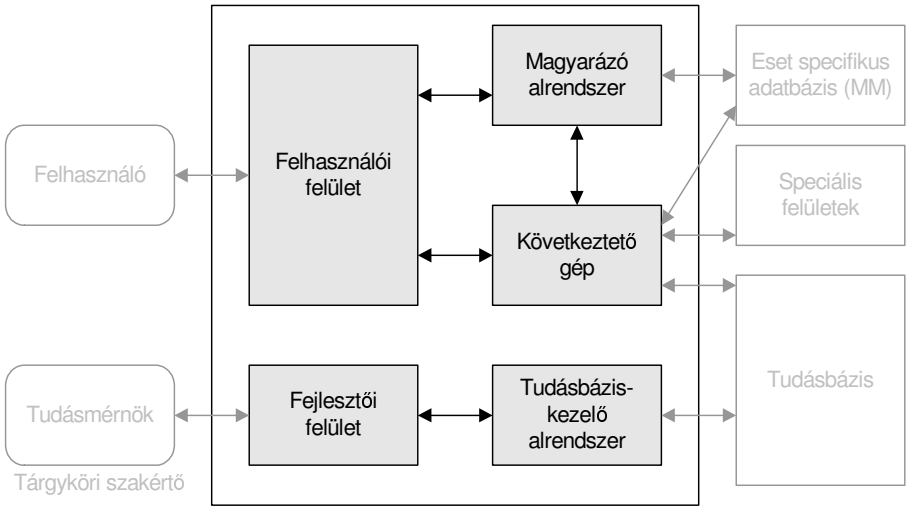
  Valamilyen tudásalapú rendszer alaptechnikát (lásd 6. fejezet) megvalósító szoftver rendszerek. Ilyen például az adatvezérelt szabályalapú következtetést megvalósító CLIPS és JESS, a GENSYM cég terméke, a G2 valós idejű szakértői keretrendszer, amely mindkét irányú következtetést támogatja, valamint a keret- és szabályalapú programozási technikával rendelkező GoldWorks. A hazai alkalmazások közül ide tartozik például a Prolog alapú ALL-EX Plus és MProlog Shell.

- Problémafüggő shellek

  Egy adott problématerület (pl. diagnosztizálás, rendszer konfigurálás, tevékenységtervezés) „kulcsrakész” rendszerei, amelyben a tudásbázist a konkrét feladatra vonatkozó ismeretekkel kell feltölteni. Pl. egy hardver konfiguráló rendszer esetén a hardverelemekkel kapcsolatos ismereteket kell formalizálni.

- Szakterületfüggő shellek

  Egy adott szakterület (pl. orvosi, jogi terület) alapismereteivel rendelkező rendszerek, amelyeket speciális szakismeretekkel kell feltölteni.



7.2. ábra. A szakértői keretrendszer alapvető komponensei

7.4. A szakértői rendszerek előnyei, hátrányai

A szakértői rendszerek előnyei az emberi szakértőkhöz képest a következők:

- Az emberi szakértők nem mindig elérhetőek, míg a szakértői rendszerek bármikor, bárhol rendelkezésre állnak, veszélyes helyeken is alkalmazhatók.
- Az emberi szakértők nem mindig megbízhatóak, nem konzisztensek, míg a szakértői rendszerek következetesek, egyenletes teljesítményt nyújtanak.
- Az emberi szakértő nem mindig tudja döntéseit megindokolni.
- A szakértői rendszer költséghatékony, elérhető áron terjeszthető, fokozza a szakértő produktivitását és megőrzi a szakértelmet.

A szakértői rendszerek hátrányai az alábbiak:

- Ismereteik egy szűk problématerületet definiálnak, korlátaikat nem ismerik.
- Tudásuk nem mindig naprakész, nem képesek tanulásra.
- Javaslataikat, tanácsaikat elemezni kell.
- Nincs hétköznapi józan eszük.
- Tudásbázisuk karbantartásához emberi szakértő, tudásmérnök szükséges.

8. fejezet

Gépi tanulás

8.1. Alapok

A számítógépek megjelenése óta az egyik legérdekesebb kérdés, hogy vajon tudjuk-e a számítógépet valamilyen módon tanítani. Képesek vagyunk-e olyan rendszereket építeni vagy akár csak olyan tanuló algoritmusokat kifejleszteni, amelyek bizonyos tapasztalatok alapján automatikusan képesek működésük hatékonyságának a javítására. Mikor mondhatjuk azt, hogy egy algoritmus tanul? Egy algoritmus tanul ha egy feladat megoldása során olyan változások következnek be a működésben, hogy később ugyanazt a feladatot vagy ahhoz hasonló más feladatokat jobb eredménnyel, nagyobb hatékonysággal képes megoldani, mint korábban.

A gépi tanulás a mesterséges intelligencia olyan részterületének tekinthető, amely a tapasztalatok feldolgozása alapján tanuló ágensek kifejlesztésével foglalkozik.

8.2. Tanuló ágens

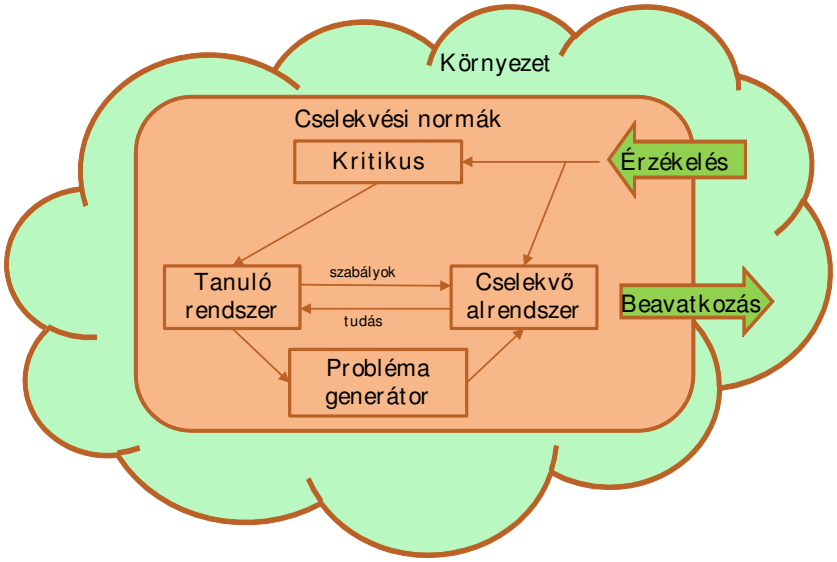
A tanuló ágenseket napjainkban sok helyen használják a fejlesztők összetett alkalmazások fejlesztésénél, így például az adatbányászati vagy folyamatbányászati alkalmazásokban. Ezen alkalmazások célja, hogy nagyméretű adatbázisokból, illetve folyamatok eseményeit rögzítő napló fájlokból értékes, nem nyilvánvaló összefüggéseket tárjanak fel algoritmikus eszközökkel.

A tanuló ágenseknek létezik egy általános modellje, amely kiindulásként szolgálhat minden tanulással felruházott ágens felépítéséhez.

A tanuló ágens 4 fő részre bontható:

1. **Cselekvő alrendszer**: Tartalmaz minden olyan ismeretet, amely a rendszer működtetéséhez szükséges. Rendelkezik egy tudásbázissal, amely felhasználható a megfelelő cselekvés kiválasztásához, amely az ágens környezetében végrehajtásra fog kerülni.

2. **Kritikus alrendszer**: Feladata annak közlése a tanuló alrendszerrel, hogy az ágens működése mennyire sikeres. A kritikus rögzített standardot alkalmaz a teljesítmény minősítésére. Erre azért van szükség, mert az észlelések önmagukban nem adnak



8.1. ábra. A tanuló ágens felépítése

információt az ágens sikerességéről, azaz maga az észlelés nem minősít. A minősítést egy ún. cselekvési norma alapján végzi el, amely alapvető elvárásokat tartalmaz a rendszer működésével, viselkedésével kapcsolatban.

3. **Tanuló alrendszer**: A kritikustól érkező visszajelzések alapján módosító szabályokat javasol a cselekvő alrendszer számára, amelynek segítségével a rendszer minőségi feladat végrehajtásán lehet javítani. A javasolt új szabályok által bővül a rendszer tudásbázisa, így a megfelelő cselekvés kiválasztása egyre bővülő tapasztalat halmaz alapján történik meg. A tanuló alrendszer így a teljesítőképesség javításáért felel.

4. **Problémagenerátor alrendszer**: Az ágens ezen részének az a fő feladata, hogy olyan cselekvést javasoljon végrehajtásra a cselekvő alrendszernek, amely új, informatív tapasztalatok megszerzéséhez vezet. Működésének a célja, hogy az ágens minőségi működését javítani lehessen. A cselekvő alrendszer ugyanis, ha azt teheti amit akar, akkor újra meg újra a jelenlegi tudása alapján fogja a végrehajtandó cselekvést kiválasztani. Ha azonban az ágens képes új területeket kutatni, akkor a hosszabb távú teljesítmény szempontjából sokkal jobb eljárás felfedezésére nyílhat lehetőség. (lásd 8.1. ábra)

Nézzünk a továbbiakban egy egyszerű példát a 4 fő egység működésére. Manapság már több kutató helyen próbálkoznak emberi beavatkozás nélkül működő járműveket fejleszteni. Ilyen járművek megmérettetésére az USA-ban versenyeket is szerveznek (lásd Grand Challenge sivatagi robotverseny), amelyeknek a célja, hogy a járművek a kitűzött feladatokat minél pontosabban végre tudják hajtani. Egy ilyen jármű is tekinthető tanuló ágensnek, amelyben a 4 fő komponens megvalósításra kell, hogy kerüljön. A cselekvő alrendszer ebben az esetben mindazon tudásnak a gyűjteményéből épül fel, amelyek szükségesek a megfelelő jármű irányítási művelet kiválasztásáért (gyorsítás, fékezés, kanyarodás stb.). A

kritikus megfigyeli az autót körülvevő világot és információkat továbbít a tanuló alrendszer felé. Ha például olyan szituációba kerül a jármű, amelyben még eddig nem volt, akkor értékeli az arra való reakciókat. Alkalmilag a problémagenerátor is javasolhat olyan új megoldásokat, amelyek eredményes kipróbálása után a megvalósítás beépülhet a rendszer működésébe. Például egy már ismert út típuson próbáljon meg az autó megnövelt sebességgel haladni. A tanuló alrendszer a cselekvő komponens hatékonyságának javításáért is felelős, így ha egy ismeretlen úton kell haladnia a járműnek, azt feltérképezve, legközelebb már a feladatvégrehajtás gyorsabb lehet.

(Megjegyzés: A tanuló ágens fogalmának megértéséhez futtassa az agens-auto.exe fájlt.)

8.3. A gépi tanulás meghatározása

Legyen egy tanulási példa egy $(x, f(x))$ adatpár, ahol x a bemenete, $f(x)$ a kimenete az x-re alkalmazott leképezésnek.

Az induktív következtetés feladata az f-re vonatkozó minták egy halmaza alapján, megadni egy olyan h leképezést, amelyik közelíti f-et. A h leképezést *hipotézis*nek nevezzük.

Az igazi f függvényt nem ismerjük, így sokféle elképzelhető választás lehet h-ra. További tudás nélkül nem tudjuk, hogy melyiket részesítsük előnyben. *Elfogultság*nak nevezzük azt, ha a példáknak való megfelelésen túl előnyben részesítjük az egyik vagy másik hipotézist. Megjegyezzük, hogy az összes tanuló algoritmus mutat valamilyen elfogultságot.

Gépi tanulás definíciója: Tekintsünk egy számítógépes programot, amely T osztályba eső feladatokat képes megoldani. A program működési hatékonyságának mérésére legyen P egy mérték, továbbá jelölje E azon tapasztalatokat, ismereteket, amelyekhez a program hozzáfér. Azt mondjuk, hogy a program a T feladatosztályon tanul, ha a P szerint mért hatékonysága javul az E ismeretek hatására.

A tanulási feladat sok esetben informatikai szemszögből úgy jelentkezik, hogy a tanuló program példákat kap (x_i, y_i) párok formájában. A programnak az a feladata, hogy olyan f függvényt keressen, amelyet használva minden adott pár esetén teljesül, hogy $f(x_i) = y_i$. A függvénytől elvárjuk, hogy alkalmazható legyen előre nem ismert x értékek esetén is az y érték meghatározására. Az ilyen típusú tanulást *felügyelt tanulás*nak nevezzük.

Ha y_i-nek két lehetséges értéke van, akkor a példákat pozitív és negatív példákra oszthatjuk. A tanulást ebben az esetben *fogalmi tanulás*nak nevezzük.

Amennyiben a tanuló algoritmus számára csak az x_i értékek állnak rendelkezésre, az y_i értékekről nincsenek ismereteink, *nem-felügyelt tanulás*ról beszélhetünk. Ilyen esetekben az algoritmus osztályozhatja a bemeneteket (osztályozási algoritmusok), vagy megpróbálhatunk bonyolultabb összefüggéseket feltárni az x_i értékek között (ismeretfeltárási algoritmusok).

8.4. Döntési fa

A gépi tanulás során használható módszerek közül az egyik a döntési fa módszer, amely sikeresen alkalmazható számos feladat megoldása esetén. A döntési fa bemenete egy tulajdonsághalmaz segítségével leírt objektum vagy szituáció, kimenete pedig egy lehetséges

döntési javaslat. A fa mindegyik belső csomópontja megfelel valamelyik tulajdonság tesztjének, és mindegyik él a tesztben résztvevő attribútum lehetséges értékeivel címkézett. A fa mindegyik levélcsomópontja megadja azt a logikai értéket, amelyet akkor kell kiadni, ha ezt a levelet elértük.

A döntési fát a szituációhoz tartozó példák felhasználásával hozhatjuk létre. A példát az attribútumok értékeivel és a célpredikátum értékeivel jellemezzük. A célpredikátum értékét a példa *besorolás*ának nevezzük.

A példákhoz tartozó döntési fa megtalálásához a következő ún. **döntési fa tanuló algoritmus**t használhatjuk fel:

Kezdetben a fa egy címkézetlen gyökér csúcsból áll, amelyhez hozzárendeljük az összes tanító példát (P) és attribútumot (A). Ha adott egy címkézetlen n csúcs, akkor a következő esetek fordulhatnak elő:

1. Ha $P = 0$, akkor levélcsúcsot kaptunk, amelynek értékét a szülőcsúcs példáinak többségi szavazása alapján határozzuk meg.

2. Ha P csak valamely kimenetnek megfelelő értéket tartalmaz, akkor egy a kimenetnek megfelelő levélcsúcshoz érkeztünk.

3. Ha $A = 0$, akkor is levélcsúcsot kaptunk és a példák többségi besorolása alapján megkapjuk a levélcsúcs értékét.

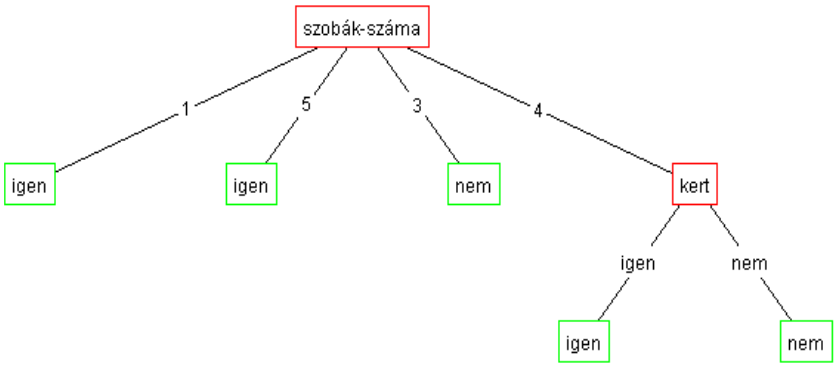
4. Egyébként a legnagyobb informativitási mérőszámmal rendelkező, még teszteletlen $a \in A$ attribútumot rendeljük az n csúcshoz, majd ezután generáljuk az összes címkézetlen utódját:

   - ezekhez az a lehetséges értékeivel címkézett élek vezetnek;
   - ha az a címkéjű n csúcsból az m csúcsba a v címkéjű él vezet, akkor az m csúcshoz rendelt
   – példák: $P_{a=v} = \{p \in P | p.a = v\}$
   – attribútumok: $A = A$-$\{a\}$
   - végül minden utódra ismételjük meg rekurzívan az 1-4 pontokat.

A 8.2. ábra egy döntési fát mutat be.

A továbbiakban nézzük meg, hogy hogyan lehet az *informativitási mérőszámot* meghatározni, és ebből hogyan adható meg az informativitási nyereség. A generáló algoritmus az összes tulajdonság összes lehetséges szétosztásai közül azt választja ki, amelyiknek a legnagyobb az informativitási nyeresége. A b tulajdonság informativitásának (I_b) kiszámítása a következő:

Legyen a csomóponthoz tartozó esetek halmaza C, a minősítő tulajdonság a, értékei $a_1 \dots a_n$, és ezek előfordulási arányai a C halmazban $w_{a1} \dots w_{an}$ $(\sum_i w_{ai} = 1)$. Ekkor a C halmaz minősítő tulajdonságának *entrópiá*ja így írható: E_C =-$\sum_i w_{ai} log_n w_{ai}$. Legyenek a b tulajdonság értékei $b_1 \dots b_n$, ezek halmaza β. Bontsuk fel β-t $\beta_1 \dots \beta_m$ nem üres részhalmazokra! Ekkor $\bigcup_i \beta_i = \beta$. Bontsuk fel C-t $C_1 \dots C_m$ részhalmazokra úgy, hogy



8.2. ábra. Egy döntési fa

C_i valamennyi elemének b tulajdonsága β_i-be essen minden i-re. Jelölje w_i a C_i súlyát C-ben ($\sum_i w_i = 1$). Ekkor $I_b = E_C$-$\sum_i w_i E_{C_i}$, így az informativitás a $\beta_1 \ldots \beta_m$ felbontásból adódó entrópianyereség. A számítás tényleges kimenete az optimális felbontáshoz tartozó $I_{b_{max}}$. Az ehhez tartozó informativitási nyereség: $D_b = w_C I_{b_{max}} / E_C$, ahol w_C a C halmaz elemeinek száma.

Az alábbi módszerrel lehetőségünk van a tanulási algoritmus teljesítményét megbecsülni, ugyanis a tanulási algoritmus akkor megfelelő, ha jó hipotéziseket szolgáltat azokban az esetekben is, amelyeket nem látott előre. A vizsgálatot a példák egy teszthalmazán végezhetjük el, amelyhez a következő lépéseket kell végig követnünk.

1. Gyűjtsük össze a példák egy nagy halmazát.
2. A példahalmazt bontsuk szét két diszjunkt halmazra: egy tanítóhalmazra és egy teszthalmazra.
3. A tanuló algoritmust a tanítóhalmaz példáira alkalmazva állítsuk elő a H hipotézist.
4. Vizsgáljuk meg, hogy H a teszthalmaz példáinak hány százalékát sorolja be helyesen.
5. Ismételjük meg az 1-4 lépéseket különböző tanítóhalmaz méretekre, és mindegyik mérethez különböző teszthalmazra.

Ennek az algoritmusnak az eredményeként kapunk egy adathalmazt, amellyel az átlagos jóslási képesség a tanítóhalmaz méretének a függvényében vizsgálható. Egy ilyen jellegű vizsgálat kapcsán megfigyelhető, hogy a tanítóhalmaz méretének a növekedésével a jóslás minősége javulni fog. (Az érdeklődő olvasó további kiegészítő információkat olvashat [17]-ben.)

8.5. Feladat és számítások

Tegyük fel, hogy ingatlan eladással foglalkozó irodánk van. Szeretnénk az eddigi eladási tapasztalatok felhasználásával azt meghatározni, hogy melyek a jól eladható ingatlanok,

8.1. táblázat. Attribútumok és lehetséges értékeik

alapterület	**szobák száma**	**kert**	**fekvés**	**kor**
25-36 m^2	1	igen	belváros	0-2 között
36-55 m^2	2	nem	kiemelt kertes városrész	2-10 között
55-80 m^2	3		kertes városrész	10-25 között
80-120 m^2	4			25 felett
120 m^2 felett	5			

milyen kritériumokkal rendelkeznek. Az attribútumok és lehetséges értékeik láthatók a 8.1. táblázatban.

A fogalmi tanulást tekinthetjük egy keresési feladatnak, mivel az a cél, hogy a hipotézisek terében megkeressük azt a hipotézist, amelyik a legjobban illeszkedik a tanulási példákra. Mivel a hipotézis tér általában nagyméretű, ezért fontos, hogy hatékony keresési módszerrel tudjuk kiválasztani a legjobban illeszkedő hipotézist. A 8.2. táblázat a rendelkezésre álló példák halmazát (P) írja le:

A döntési fa felépítésének menete:

1. lépésben kiszámoljuk az entrópia értékét a teljes példa halmazra vonatkozóan.

E(P)=-$\frac{4}{7}$*$\log_2\frac{4}{7}$-$\frac{3}{7}$*$\log_2\frac{3}{7}$= -0,5714*$\frac{\lg 0{,}5714}{\lg 2}$-0,4285*$\frac{\lg 0{,}4285}{\lg 2}$=
=0,4612+0,5238=0,985

2. lépésként minden tulajdonság esetében meghatározzuk az informativitási nyereséget.

C(P,alapterület)=0,985-$\frac{1}{7}$*$E(25-36m^2)$-$\frac{2}{7}$*$E(120m^2 felett)$-$\frac{1}{7}$*$E(55-80m^2)$-
$\frac{2}{7}$*$E(80-120m^2)$-$\frac{1}{7}$*$E(36-55m^2)$=0,985-0,2857=**0,6993**

C(P,szobák száma)=0,985-$\frac{1}{7}$*E(1)-$\frac{2}{7}$*E(5)-$\frac{2}{7}$*E(3)-$\frac{2}{7}$*E(4)=**0,6993**

C(P,kert)=0,985-$\frac{4}{7}$*E(nem)-$\frac{3}{7}$*E(igen)=0,5214

C(P,fekvés)=0,985-$\frac{3}{7}$*E(belváros)-$\frac{2}{7}$*E(kiemelt kertes városrész)-
$\frac{2}{7}$*E(kertes városrész)=0,3056

C(P,kor)=0,985-$\frac{2}{7}$*E(25 felett)-$\frac{3}{7}$*E(2-10 között)-$\frac{1}{7}$*E(0-2 között)-
$\frac{1}{7}$*E(10-25 között)=**0,6993**

8.2. táblázat. A döntési fa felépítéséhez használt példák halmaza

Példa	alapterület	szobák száma	kert	fekvés	kor	jól eladható
1.	25-36 m^2	1	nem	belváros	25 felett	**igen**
2.	120 m^2 felett	5	igen	kiemelt kertes városrész	2-10 között	**igen**
3.	55-80 m^2	3	nem	kertes városrész	0-2 között	**nem**
4.	80-120 m^2	4	igen	kiemelt kertes városrész	2-10 között	**igen**
5.	120 m^2 felett	4	nem	belváros	25 felett	**nem**
6.	36-55 m^2	3	nem	belváros	10-25 között	**nem**
7.	80-120 m^2	5	igen	kertes városrész	2-10 között	**igen**

3. Látható, hogy a kiszámolt értékek közül a legnagyobb érték az *alapterület*, a *szobák száma* és a *kor* attribútumokhoz tartozik, ezek közül válasszuk a szobák száma attribútumot a döntési fa gyökerébe.

4. lépésként a gyökér csúcsból kiinduló éleket megcímkézzük a szobák száma attribútum lehetséges értékeivel, rendre 1, 5, 3 illetve 4 címkével. Az élekhez rendelt még címkézetlen csúcsokhoz hozzárendeljük azt a példahalmazt, amelybe azon példák fognak tartozni, amelyekben a szobák száma attribútum az 1, 5, 3 illetve 4 értékeket veszi fel.

Így $P1$-be kerül az 1. tanulási példa, mivel ebben a szobák száma attribútum az 1 értéket veszi fel.

A $P2$-be kerül a 2. és 7. tanulási példa, mivel a szobák száma attribútum ezekben az 5 értéket veszi fel.

A $P3$-ba kerül a 3. és 6. tanulási példa és végül a $P4$-be a 4. és 5. tanulási példa.

5. lépésként megvizsgáljuk a $P1$, $P2$, $P3$ és $P4$ halmazokba tartozó tanulási példák besorolását, azaz a *jól eladható* attribútum értékét.

A $P1$-ben a besorolás értéke *igen*, ezért levél csúcshoz érkeztünk és a levél csúcs értéke *igen* lesz.

A $P2$-ben a besorolások értéke *igen*, ezért szintén levél csúcshoz érkeztünk és a levél csúcs értéke *igen* lesz.

A $P3$-ban a besorolások értéke *nem*, ezért szintén levél csúcshoz érkeztünk és a levél

csúcs értéke *nem* lesz.

A $P4$-ben a besorolások értéke *igen* és *nem*, ezért meg kell vizsgálni az attribútumokat, melyik a leginformatívabb, azaz melyik kerüljön a csúcsba.

6. lépésként kiszámoljuk az entrópia értékét a $P4$ példa halmazra vonatkozóan.
E(P4)=-$\frac{1}{2}$*$\log_2\frac{1}{2}$-$\frac{1}{2}$*$\log_2\frac{1}{2}$=1

7. lépésként minden tulajdonság esetében meghatározzuk az informativitási nyereséget, kivéve a *szobák száma* attribútumot.
C(P4,alapterület)=1-$\frac{1}{2}$*$E(120m^2 felett)$-$\frac{1}{2}$*$E(80-120m^2)$=1

C(P4,kert)=1-$\frac{1}{2}$*E(nem)-$\frac{1}{2}$*E(igen)=1

C(P4,fekvés)=1-$\frac{1}{2}$*E(belváros)-$\frac{1}{2}$*E(kiemelt kertes városrész)=1

C(P4,kor)=1-$\frac{1}{2}$*E(25 felett)-$\frac{1}{2}$*E(2-10 között)=1

Mivel az értékek azonosak lettek, válasszuk a csúcsba a *kert* attribútumot.

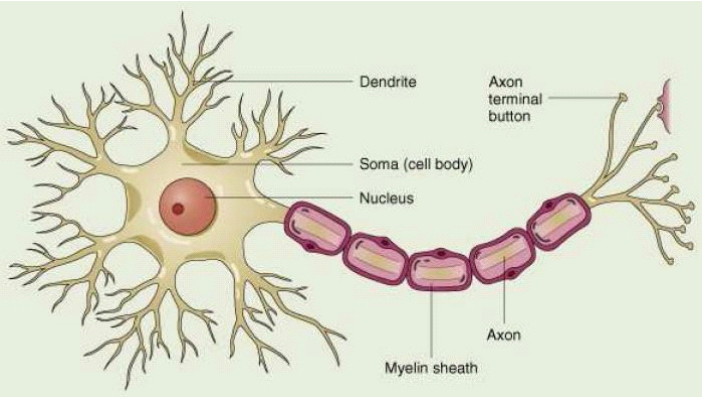
Ezek után a döntési fa algoritmus lépéseit tovább követve folytathatjuk a döntési fa felépítését, amelynek lépéseit a `dontesifa.avi` fájl mutatja be (a fájl az AI Space Decision Trees http://www.aispace.org/dTree/ honlapon található segédprogrammal készült).

9. fejezet

Neurális hálózatok

9.1. Alapok

A mesterséges neurális hálózat egy rendszer, melynek modellje az emberi agy. A területnek nagyon sok rokon értelmű elnevezése létezik, pl.: párhuzamosan elosztott feldolgozás, neuroszámítás, gépi tanulási algoritmus, természetes intelligens rendszerek és mesterséges neurális hálózatok. Tulajdonképpen a neurális hálózat egy kísérlet, amely speciális hardver elemek, és összetett szoftver segítségével szimulálni próbálja a neuronok több rétegű, de egyszerű működési elvét. Minden egyes neuron összeköttetésben áll bizonyos számú szomszédjával, ahol változó összeköttetési együtthatóval vesz részt a kapcsolatban, amely a kapcsolat erősségét reprezentálja. A tanulási folyamat úgy zajlik, hogy a kapcsolatok erősségét változtatjuk olyan irányba, ami a teljes rendszert a helyes eredmény elérésére sarkallja. A neurális hálózatok legalapvetőbb összetevőit az agy felépítése alapján modellezzük. Néhány neurális hálózati struktúra nem hozható közeli kapcsolatba az aggyal, ugyanakkor más felépítésű rendszereknek pedig a biológiai modellje nem létezik. Mégis általánosságban elmondható, hogy a neurális hálózatok erős hasonlóságot mutatnak az aggyal, ezért a terminológia nagy része az idegsebészetből lett átvéve.



9.1. ábra. Egy neuronsejt

A neurális hálózatok tervezési folyamatának lépései:

1. A neuronok rétegekbe szervezése.

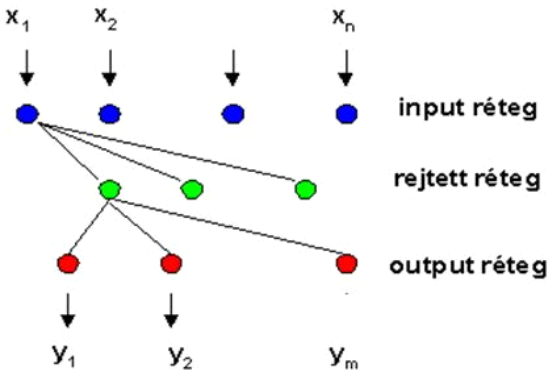
2. A neuronok kapcsolat típusának meghatározása mind a rétegen belül, mind az egyes rétegek között.

3. Meg kell határozni azt a módszert, ahogy egy neuron fogadja az inputot, és kibocsátja az outputot.

4. Meg kell határozni a kapcsolat erősségét a hálózaton belül úgy, hogy lehetővé tegyük a hálózat számára azt, hogy megtanulja a kapcsolatok megfelelő súlyozását egy ún. öntanuló adatbázist felhasználva.

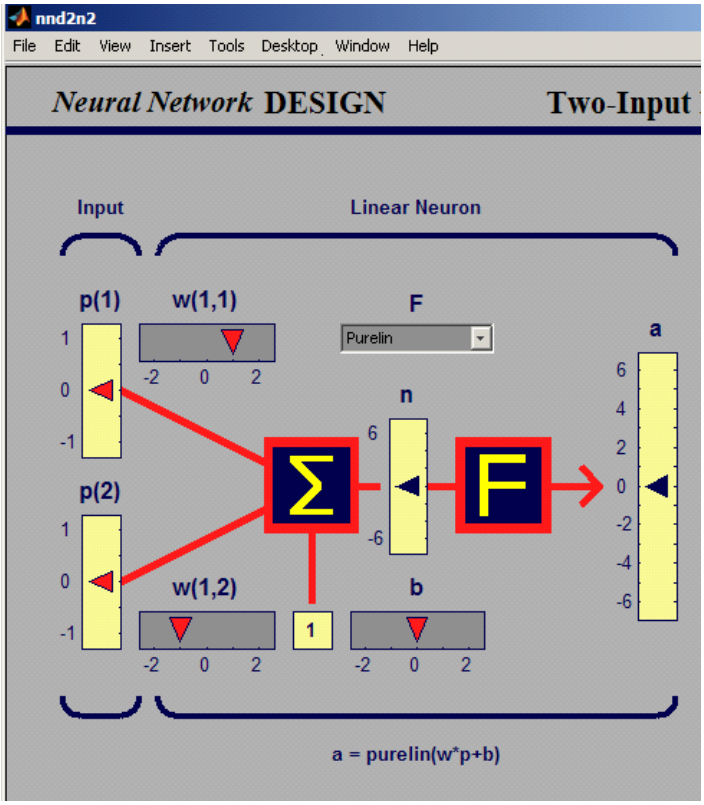
Neurális hálózat tervezéséhez és a hálózat működésének szimulációjához például használhatjuk a MATLAB® [16] Neural Network Toolboxot (lásd 9.3. ábrát).

Általánosságban elmondható, hogy minden neurális hálózatnak hasonló a topológiája. Némely neuronok kapcsolatban állnak a külvilággal, hogy fogadják a bejövő adatokat (input), mások a valóság felé kibocsájtják az eredményeket (output), míg az összes többi neuron rejtve marad a külvilág számára. A 9.2. ábra a neurális hálózat rétegeit szemlélteti.



9.2. ábra. A neurális hálózat rétegei

Az emberi agyban a neuronok (idegsejtek) számát mintegy 10^{11}-re teszik. A neuronok *dendrit*jeiken keresztül kapcsolódnak más neuronokhoz. A sejttest meghoszabbodott nyúlványa az *axon*, amely végződése közelében bokorszerűen szerteágazik. Az axon a *szinapszis*okon keresztül kapcsolódik más neuronok dendritjeihez vagy azok sejttestjéhez. A szinapszis küldő végén elhelyezkedő neuront *preszinaptikus*nak, a fogadó végén lévő neuront *posztszinaptikus*nak nevezik. Egy tipikus neuron axonja néhány ezer szinapszissal kapcsolódik más neuronokhoz. A neuronok az axonjaikon keresztül elektromos kisüléssorozatokkal kommunikálnak. A szinapszison keresztüli jelátvitel bonyolult kémiai folyamat. A folyamat során a küldő végről beérkező elektromos kisüléssorozat hatására ún. *transzmitter anyag*ok szabadulnak fel, amelyek végső soron megnövelik, vagy éppen csökkentik a fogadó végén lévő sejttest elektromos potenciálját. Ha ez a potenciál elér egy küszöböt, a neuron egy ún. *akciós potenciál*lal válaszol: egy rögzített hosszúságú elektromos kisüléssorozatot küld az axonján keresztül a vele összekötetésben álló neuronok felé. Ekkor mondjuk, hogy a sejt



9.3. ábra. Neurális hálózat létrehozása Matlab segítségével

tüzelt. Tüzelés után a sejt egy meghatározott ideig, a refraktorikus időszakban nem tud újra tüzelni.

(Megjegyzés: A neuronok információ átadó képességének bemutatásához futtassa a neuron-bio.exe fájlt.)

Amikor a bemeneti réteg fogad egy inputot a neuronjai outputtá alakítják, ami a rendszer következő rétege(i) számára lesz bemeneti adat. A folyamat addig folytatódik, amíg meg nem felel egy meghatározott célnak, vagy amíg a kimeneti réteghez nem fordul a rendszer, ami végül kiadja az eredmény(ei)t a külvilágnak. A neuronok utak hálózatán keresztül kapcsolódnak, továbbítva egy neuron kimeneti adatát a következő neuron számára bemeneti adatként. Ezek az utak általában egyirányúak, bár lehet kétirányú kommunikáció is a neuronok között, hiszen létezhet egy másik út is az ellenkező irányban. Egy neuron sok másiktól fogadhat inputot, de egy outputot produkál, amelyet továbbad a többi neuron felé. Egy neuron a rétegen belül kommunikálhat az összes többi neuronnal, de előfordulhat, hogy egyikkel sincs kapcsolatban. Egy réteg neuronjai azonban minden esetben összeköttetésben állnak minimum egy másik réteg neuronjaival.

Különböző típusú kapcsolat lehet a rétegek között, ezeket hívjuk **rétegek közti kapcsolat**nak:

- *Teljesen összekapcsolt*: Az első réteg összes neuronja összeköttetésben áll a következő réteg összes neuronjával.

- *Részlegesen összekapcsolt*: Az első réteg egy neuronjának nem kell feltétlenül kapcsolódnia a következő réteg összes neuronjához.

- *Add tovább kapcsolat*: Az első réteg neuronjai a kimenő adataikat továbbküldik a következő réteg számára, de semmiféle visszacsatolást a második rétegtől nem kapnak.

- *Kétirányú kapcsolat*: Ebben az esetben a második réteg visszacsatol az első réteg felé. Az Add tovább és a Kétirányú kapcsolat lehet teljesen összekapcsolt és részlegesen összekapcsolt.

- *Hierarchikus kapcsolat*: Ha egy neurális hálózat hierarchikus szerkezetű, az alsóbb réteg neuronjai csak egy következő (mélyebb) réteg neuronjaival kommunikálhatnak.

- *Ismétlődő kapcsolat*: A rétegek kétirányú kapcsolattal rendelkeznek, és egy meghatározott ideig folyamatosan küldhetik üzeneteiket a kapcsolatokon keresztül mindaddig, amíg egy meghatározott célt el nem ér a rendszer.

Rétegen belüli kommunikáció: Összetettebb struktúrák esetén a neuronok kommunikálnak egymással a rétegen belül, ezt nevezzük rétegen belüli kommunikációnak. Kétféle rétegen belüli kommunikációt említenénk meg:

- *Visszatérő, ismétlődő kapcsolat*: A réteg neuronjai teljesen, vagy részlegesen összekapcsoltak. Miután ezek a neuronok fogadták egy másik réteg adatait (input), a saját kimenő adataikat (output) többször „megbeszélik” egymással, mielőtt azt tovább küldhetnék egy következő réteg számára. Általában egy bizonyos követelményt a neuronoknak a rétegen belüli kommunikáció során teljesítenie kell, mielőtt tovább küldhetnék eredményeiket a következő rétegnek.

- *Központi/környezetet kizáró kapcsolat*: A rétegben lévő neuron ösztönző (támogató) kapcsolatban áll önmagával és közvetlen szomszédjaival, és gátló kapcsolatban van az összes többi neuronnal. El lehet képzelni ezt a fajta kapcsolatot úgy, mint neuronok egymással versenyző csoportjait. Minden egyes csoport ösztönzi önmagát és a csoport tagjait, és gátolja a többi csoportot, azok tagjait. Néhány ciklus adatcsere után az aktív outputtal rendelkező neuronok „győznek”, és a döntési súlyuk (és csoportjuké) a hálón belül növekszik. Kétféle kapcsolat létezik a neuronok között: támogató, és gátló. Támogató kapcsolat esetén egy neuron kimenete a vele kapcsolatban álló másik neuron döntési potenciálját (súlyát) növeli. Gátló kapcsolat létrejöttekor a kimenetet küldő neuron a befogadó neuron döntési (cselekvési) potenciálját csökkenti. Az első típus a vevő neuron összegző mechanizmusát hozzáadásra készteti, míg a második kivonásra, azaz az első erősít, míg a második gyengít.

Leegyszerűsítve a dolgot, a neurális hálózat felfogható egy olyan modellnek, amelyben minden neuron *bekapcsolt*, illetve *kikapcsolt* állapotban lehet, és amelyben az átkapcsolás *be* állapotba akkor kerül, amikor a neuront kellő számú szomszédos neuron stimulálja.

A neurális hálózatok jellemzői:

- *Taníthatók*: létezik olyan eljárás, amellyel a hálózat bemenetére adott mintákhoz tartozó kimeneti minták többszöri megadása után a hálózat valamennyi betanult mintához a kívánt legjobb kimenetet szolgáltatja.
- *Általánosító képességgel rendelkezik*: a be nem tanított esetekre is közelítően jó megoldást adnak.
- *Hibatűrők*: az előző tulajdonságokból adódóan a bemeneti hibákra kevésbé érzékenyek.
- *Gyors a válaszidejük*.

Főbb alkalmazási területek:

- mintafelismerés (hang- és képfeldolgozás, alakfelismerés, jelfeldolgozás),
- nem lineáris rendszerek szimulációja (előrebecslés, tanácsadás),
- adattömörítés (képtovábbítás),
- szabályozás.

9.2. Neurális hálók tanulása

Az agy általában a tapasztalatokból tanul. A neurális hálózatokat időnként gépi tanulási algoritmusnak is nevezik, mert a kapcsolat súlyának (tanulás) megváltoztatásával tanulja meg a rendszer egy probléma megoldását. A kapcsolat erősségét a neuronok között egy súly értékként tároljuk. A rendszer úgy tanulja meg az új ismereteket, hogy ezeken a súly értékeken változtat. Ami azt jelenti, hogy a háló súlyait „tanítjuk”, ami egy iteratív eljárás, melynek során a háló által megvalósított leképezést egy kívánt leképezéshez közelítjük. A kívánt leképezés a neuronháló tanításánál összetartozó be- és kimeneti mintapontok $\{x_i, y_i\}$ $i = 1, 2, \ldots, p$ alapján történik. A tanító pontokban meghatározva a háló kimenetét és összehasonlítva a kívánt kimenettel hiba képezhető, mely hibaértékek minimalizálása a cél.

Általános esetben egy hibafüggvény definiálható és a tanítás során a szabad paraméterek olyan beállítása a cél, mely a kritérium függvény minimumát biztosítja. Egy neurális hálózat tanulási képességét a felépítése és a tanulási algoritmusa határozza meg.

A tanulási módszer lehet:

- *Nem felügyelt tanulás*: A rejtett neuronoknak meg kell találniuk a módot az önszervezésre külső segítség nélkül. Ennél a megközelítésnél nincsenek „minta eredmények” a rendszer számára, amely segítségével megjósolhatná a teljesítményét a kapott értékek (inputok) függvényében. Itt a rendszer a folyamat végrehajtása során tanul.
- *Megerősítéses tanulás*: A módszer a külső megerősítésen alapul. A kiinduló kapcsolat a rejtett réteg neuronjai között véletlen szervezésű, amit újraszerveznek, amint a rendszernek „megmondják” milyen közel áll a probléma megoldásához.

A tanulási módszereket egy másik csoportosítás szerint is kategorizálhatjuk. Ha a rendszer arra használja a bejövő adatokat, hogy súlyarányait megváltoztassa a tudás elsajátítása érdekében, akkor a rendszer *betanuló üzemmódban* működik.

Tanulási szabályok

Nagyon sokféle szabály van használatban. Ezek matematikai algoritmusok, melyeket a kapcsolatok súlyának frissítésére használnak. A tanulási folyamat annál sokkal összetettebb, mint a ma ismert szabályok által nyújtott egyszerűsített ábrázolás. A kutatás az újabb és újabb tanulási szabályok megismerésére tovább folytatódik, új ötletek nap mint nap bukkannak fel a publikációkban. A következő szabályokat példaként mutatjuk be:

- *Hebb szabály*: Az első és legismertebb tanulási szabály D. Hebb nevéhez kapcsolódik. Alapelve: Ha egy neuron fogad egy adatot (input) egy másik neurontól, és ha mindkettő nagyon aktív (matematikailag ugyanazt az értéket tartalmazza), akkor a neuronok súlyát erősíteni kell. Ez azt jelenti, hogy az i-edik és a j-edik neuron közötti kapcsolat súlya egy t időpontban $w_{ij}(t)$. Az i-edik neuron aktivációs állapota A_i a j-edik neuron aktivációs állapota A_j és legyenek ezek Boole-értékek (pl. 0 és 1). A Hebb szabály a t és a $(t+\delta t)$ között érkező hatás esetén azt eredményezi, hogy $w_{ij}(t+\delta t) = w_{ij}+\mu A_i A_j$, $\mu > 0$ tényező a tanulás erőssége.

- *Hopfield szabály*: A szabály hasonló Hebb szabályához azzal a különbséggel, hogy meghatározza az erősítés vagy gyengítés nagyságát. Azt állítja, hogy: „...ha a tervezett output és input egyszerre aktív vagy inaktív, akkor növelje a kapcsolat súlyát a tanulás arányával, egyébként csökkentse a kapcsolat súlyát a tanulás arányával.” (A legtöbb esetben a tanulás aránya, a tanulási állandó egy pozitív, nulla és egy közé eső érték.)

- *Delta szabály*: A Delta szabály is Hebb szabályára épül, és ez az egyik legelterjedtebb a szabályok közül. A szabály a bemeneti kapcsolatok súlyának folyamatos változtatására épül, hogy csökkentse a különbséget (delta) a neuron tervezett és valós kimeneti értéke között. A szabály úgy változtatja a súlyt, hogy minimalizálja a rendszerhiba négyzetes középértékét. A hibát visszatovábbítják az eggyel előző rétegbe. A rendszerhibák visszacsatolása mindaddig folytatódik, míg el nem érik a legfelső réteget.

- *Kohonen tanulási szabálya*: Ezt az eljárást T. Kohonen fejlesztette ki. Itt a neuronok „versenyeznek” a tanulás lehetőségéért a súlyuk növelése érdekében. A legnagyobb outputtal rendelkező neuront tekintik „győztesnek” és annak van meg az a képessége, hogy gátolja versenytársait, vagy erősítse szomszédjait. Csak a győztes bocsáthat ki outputot, és csak a győztes és szomszédai növelhetik kapcsolati súlyukat.

Azt gondolhatnánk, hogy minél több neuronunk van, annál jobban tanul a háló, de ez nem így van. A hálózat érzékennyé válhat a konkrét adatokra. Függvényközelítésben ez úgy látszik, hogy a függvénygörbe nagyon jól illeszkedik az alappontokban, de közöttük feleslegesen hullámzik. Ekkor a hálózatot *túlillesztett*nek (túltanítottnak) nevezzük. Ezért meg kell találnunk a neuronok számának és a tanítóhalmaznak az optimális nagyságát.

9.3. A McCulloch-Pitts neuron modell

Legyenek a bemeneti adatok $x(n) = [x_1(n), \ldots, x_m(n)]^T$ m dimenziós vektorok, ahol $n = 1, 2, \ldots$ az időpillanatokat jelenti. Az ismeretlen rendszer minden $x(n)$-hez megad egy $d(n)$ kimeneti értéket.

A célunk az, hogy ezt az ismeretlen rendszert helyettesítsük egy olyan egységgel, amely adott input esetén „hasonló” outputot ad, mint az eredeti.

A McCulloch-Pitts neuron sematikus modellje: A neuron tüzel, ha inputjainak súlyozott összege $\sum_{j=1}^{m} w_j x_j$ meghaladja a θ küszöböt.

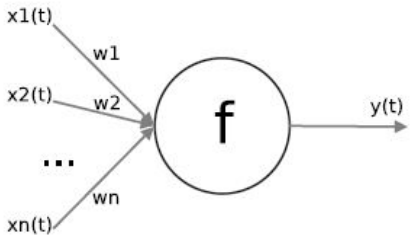
(Megjegyzés: A neuron működésének bemutatásához futtassa a neuron-mat.exe fájlt.)

A McCulloch-Pitts modellben egy modellneuron működését az

$$y(t+1) = sgn^+(\sum_{j=1}^{m} w_j x_j(t) - \theta)$$

egyenlettel írhatjuk le, ahol sgn^+ az egységugrás függvény: $sgn^+(h) = 1$, ha $h > 0$ és $sgn^+(h) = 0$ minden más esetben. $x_j(t) \in \{0, 1\}$ pontosan akkor 1, ha a j-edik preszinaptikus neuron a t. időpillanatban tüzel és $y(t+1) \in \{0, 1\}$ akkor 1, ha a modellezett neuronunk tüzelni fog a $t+1$-edik időpillanatban. A w_j súlyok modellezik a j-edik szinapszis átvivő képességét. Ha $w_j > 0$, akkor a szinapszis gerjesztő, ha $w_j = 0$, akkor a szinapszis nem ereszt át jelet, ha $w_j < 0$, akkor a szinapszis gátló. A θ paraméter a neuron tüzelési küszöbét jelzi: ha az inputok súlyozott összege meghaladja a θ értéket, a neuron a $t + 1$-edik pillanatban tüzelni fog, egyébként pedig nem tüzel. A neuron tanulását az w_j súlyok ($j = 1, \ldots, m$) változtatásával valósíthatjuk meg.

A 9.4. ábrán egy neuron sematikus ábráját láthatjuk.



9.4. ábra. Az általános neuron modell

9.4. Hopfield típusú hálózatok

A neuronhálózatok McCulloch-Pitts modelljei neuronok sokaságából felépülő rendszerek. A neuronhálózatok váza egy súlyozott irányított gráf: a gráf csúcspontjaiban vannak a neuronok, a neuronok összeköttetéseit pedig a gráf éleihez rendelt súlyok határozzák meg. Ha az i csúcsból van él a j csúcsba, akkor a j csúcshoz tartozó neuron bemenetet kap az i

neurontól, és az i. neuron outputja a j. neuron számára az (i,j) élhez tartozó valós számmal súlyozottan jelenik meg.

A Hopfield hálózatok eredetileg az asszociatív memória modellezésének céljára szolgáltak, később azonban kombinatorikai feladatok megoldására és a kiszámíthatóság modellezésére is felhasználták őket. Ha például az asszociatív memóriaegységünkben képeket tárolunk, akkor a memóriaegység már a kép egy része alapján is képes kell legyen a kép hiányzó részének kiegészítésére. A Hopfield hálózatokban a memórianyomokat, mint az $X = \{-1, 1\}^N$ tér elemeit reprezentáljuk. A hálózat N db McCulloch-Pitts-féle neuronból áll, mindegyik neuron mindegyik másik neuronnal össze van kötve.

Az egyrétegű visszacsatolt háló szerkezete:

Az alapmodell esetén a háló bemenetei binárisak. Az egyrétegű modellben a neuronok kimenetei minden neuron bemenetére súlyozott összeköttetéseken keresztül vannak visszacsatolva. A megfelelően beállított súlyokkal rendelkező hálózattól azt várjuk, hogy az adott kezdeti konfigurációból kiindulva az ehhez legközelebbi stabil állapothoz tartozó pontban stabilizálódik.

Célja egy nyugalmi helyzet előállítása:

- Minden neuron kezdő állapota egy-egy bemeneti érték.
- Egy neuron mindaddig újra számolja a többi neuron kimeneti értéke alapján a belső állapotát, ameddig az eltér a korábbi állapotától.
- A stabil helyzetben kialakult állapotok lesznek a kimeneti értékek.

A modell működése:

Az i-edik neuron az

$$x_i(t+1) = sgn(\sum_{j=1}^{N} w_{ij} x_j(t) - \theta_i)$$

képlet alapján számolja ki a következő időpillanatbeli aktivitását, ahol $sgn(h) = -1$, ha $h < 0$, egyébként $sgn(h) = 1$. Itt w_{ij} az i-edik neuront a j-edikkel összekötő kapcsolat erőssége, és θ_i az i-edik neuronhoz tartozó küszöbérték. Minden időpillanatban egy és csak egy neuron számolja újra az aktivitását.

A hálózat outputja az egyensúlyi helyzet beálltakor kialakult mintázat. Ennek előállítása a súly értékek megfelelő változtatásával lehetséges pl. a Hebb szabály alkalmázásán keresztül. Az egyensúlyi helyzet akkor áll be, amikor már minden neuron újraszámolta az aktivitációs értékét. Az ilyen hálózatokat használják, pl. karaktersorozatok felismerésére és mintázatok generálására.

A neuronhálózatok további családjai (ezekkel nem foglalkozunk ezen keretek között részletesebben, az érdeklődő olvasónak a [4]-t ajánljuk további tanulmányozásra):

- versengő neuronok,
- előrecsatolt neuronhálózatok stb.

(Megjegyzés: A neuronháló működésének bemutatásához futtassa az idojaras.exe fájlt. A fejezethez tartozó feladatok az NHMatlab1.pdf fájlban található.)

10. fejezet

Evolúciós stratégiák, genetikus algoritmusok

10.1. Alapok

Ebben a fejezetben az evolúciós stratégiák/módszerek közül csak a leggyakoribb, a biológiai evolúció által ösztönzött változatokról szólunk és azok közül is kiemelten a genetikus algoritmusokról.

Az algoritmus mechanizmusa és fogalom rendszere a darwini evolúciós elméletre és a genetika alapjaira épít. Azt mondhatjuk, a módszer mindenhol alkalmazható, ahol a feladat sok lehetséges megoldás közül a legjobbat megkeresni, ahol az értéket egy rátermettségi függvény (fitness function) adja meg. A genetikus algoritmus megoldások egy populációját tartja fenn, azaz egyszerre több megoldással (egyeddel) dolgozik. Az aktuális populációból minden lépésben egy új populációt állít elő úgy, hogy a szelekciós operátor által kiválasztott legrátermettebb elemeken alkalmazza a rekombinációs és mutációs operátorokat.

Az alapgondolat az, hogy mivel általában minden populáció az előzőnél rátermettebb elemeket tartalmaz, a keresés folyamán egyre jobb megoldások állnak rendelkezésre.

(Megjegyzés: A genetikus algoritmus működési alapelvének bemutatásához futtassa a genetikus.exe fájlt.)

A 60-as években merült fel először az a gondolat, hogy az evolúcióban megfigyelhető szelekciós folyamatok mintájára olyan számítógépes modelleket lehetne létrehozni, amelyek képesek mérnöki – elsősorban optimalizálási – feladatok megoldására. A több mint 50 év alatt mind a szűkebb értelemben vett genetikus algoritmusok, mind a tágabban értelmezett evolúciós módszerek komoly fejlődésen mentek keresztül, többféle változatuk jött létre és egyre teljesebbé vált a matematikai megalapozásuk. 4 fő kutatási terület alakult ki:

1. 1966 Fogel, Qwens, Walsh: evolúciós programozás,
2. 1973 Rechenberg: evolúciós stratégiák,
3. 1975 Holland: genetikus algoritmusok,
4. 1992 Koza: genetikus programozás.

Mi ezek közül elsősorban a genetikus algoritmusokkal foglalkozunk a továbbiakban, mivel a legtöbb alkalmazás ezen területhez kapcsolódik. (A többi területről bővebben [5]-ben és [2]-ben olvashat a kedves olvasó.)

Az életnek nagyon sok olyan területe van, ahol találkozhatunk optimalizálási feladatokkal, így például:

- a napi beosztásunk elkészítése,
- két város között az optimális út megtervezése,
- egy kórházban a betegek számára a napi menü összeállítása stb.

Amennyiben létezik alkalmazható módszer az adott problémára, akkor az jobban viselkedik a genetikus algoritmusoknál, vagyis gyorsabban és az optimumot jobban megközelítve ad eredményt. Vannak viszont problémák (lásd előbb említett feladatok), amikor a klasszikus módszerek nem alkalmazhatók, ilyenkor lehet segítségünkre a genetikus algoritmus. A genetikus algoritmusok fő jellemzői:

- többpontos keresést valósítanak meg,
- flexibilisek,
- robosztusak: ha egy keresési út zsákutcának bizonyul, az még nem jelenti az egész algoritmus kudarcát,
- biztosítják, hogy elfogadhatóan gyorsan, elfogadhatóan jó megoldást találjunk,
- a problémának nem egy, hanem több különböző közel optimális megoldása lehet, amelyből a felhasználó ki tudja választani a neki legmegfelelőbbet.

10.2. Az algoritmus általános felépítése

Az alap algoritmus lépései:

1. Hozzuk létre a P_0 kezdeti populációt
2. t := 0
3. while not Kilépés(P_t)
4. P'_t := UjElemek(P_t)
5. P_{t+1} := UjPopuláció(P'_t,P_t)
6. $t := t + 1$
7. end of while

A kezdő populáció feltöltése leggyakrabban véletlen elemekkel történik. A populációk elemszáma a futás során változatlan, az 50-100 körüli értékek a tipikusak. A Kilépés(P_t) függvény sokszor nem is függ a populációtól, mint ahogy a jelölés sejtetné, a tipikus módszer a már kiértékelt megoldások számának a vizsgálata. Az algoritmus futása ebben az esetben akkor áll le, ha egy előre adott mennyiségű különböző lehetséges megoldás rátermettségét kiszámoltuk. A megengedett kiértékelések konkrét száma függ a problémától, leggyakrabban 1000 és 500000 között van. Az UjElemek(P_t) függvény feltölti a P'_t populációt új elemekkel. Egy új megoldás előállítása legtöbbször úgy történik, hogy a szelekció segítségével szülőket választunk. A kiválasztás alapja az egyes egyedek fitneszértéke (rátermettsége), majd a rekombináció ezekből egy utódot hoz létre. Erre azután alkalmazható a mutáció, majd az új elem rátermettségének a meghatározása következik. A P'_t populáció elemszáma nem feltétlenül azonos P_t elemszámával. Ha mégis azonos, az UjPopuláció(P'_t,P_t) függvény egyszerűen a P'_t populációt adja. Ekkor a genetikus algoritmus generációs. Ha azonban P'_t elemszáma kisebb, a megfelelő mennyiségű elemet törli a régi populációból, és a helyükre P'_t elemei kerülnek. A genetikus algoritmus egymást követő lépéseit a 10.1. ábra mutatja be.

10.3. A gráfszínezési probléma

A probléma: Egy adott irányítatlan gráfhoz találnunk kell egy jó k-színezést, ami azt jelenti, hogy minden csúcshoz az előre adott k különböző szín valamelyikét kell rendelni úgy, hogy minden él két különböző színű pontot kössön össze.

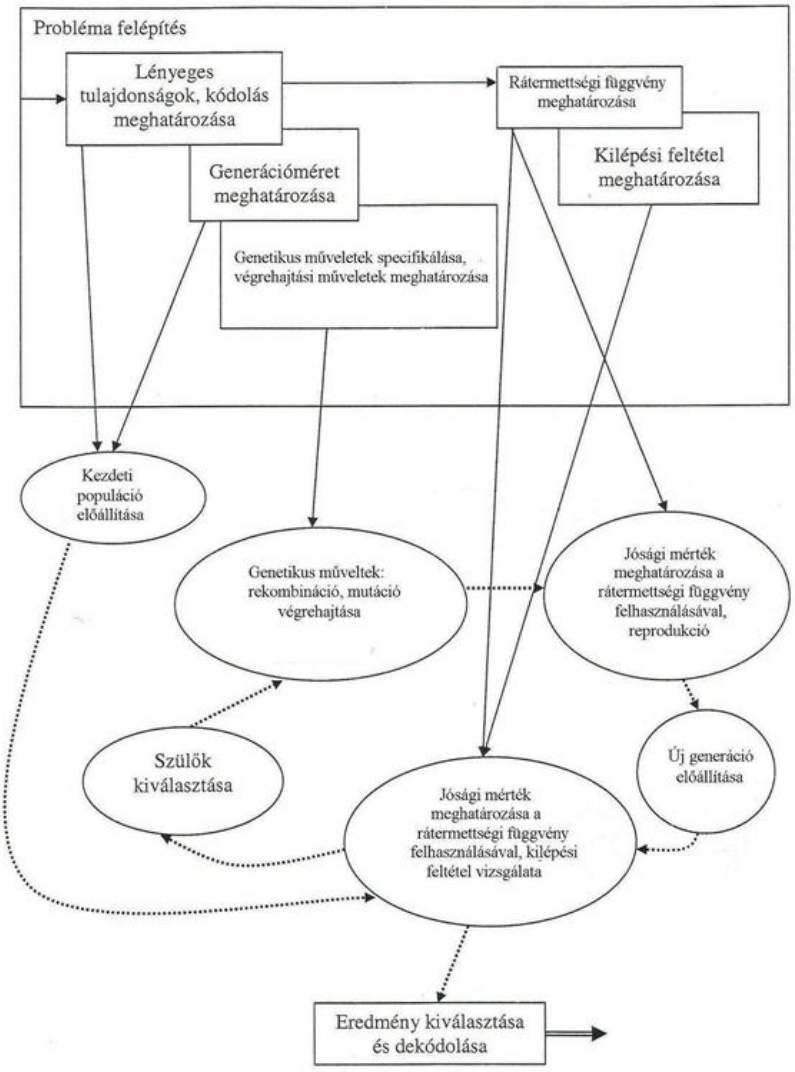
A genetikus algoritmus populációk sorozatát állítja elő, a populációk megoldásokat tartalmaznak. Itt egy megoldás a gráf egy színezése.

A megoldásokat kódolni kell, hogy újabb megoldásokat lehessen szerkeszteni. Minden megoldáshoz így hozzárendelünk egy szintaktikailag jól manipulálható betűsorozatot egy kódoló függvény segítségével. Kétféle kódolást is megemlítünk ezen feladathoz:

- *Direkt kódolás*: A gráf pontjaihoz rendelt színeket (a színekhez rendelt sorszámokat) egyszerűen felsoroljuk, ez a számsorozat lesz a kód.
- *Sorrendi kódolás*: Sorra vesszük a pontokat és minden pontot kiszínezünk arra a minimális sorszámú színre, amelynek használata az adott helyzetben lehetséges. Ha egyetlen szín sem megfelelő, színezetlenül hagyjuk a pontot. Ha létezik jó színezés, akkor minden pontnak lesz színe, egyébként maradnak színezetlen pontok.

A **rátermettségi függvény**t például úgy definálhatjuk, hogy a rosszul színezett (azaz azonos színű pontokat összekötő) élek számának a függvénye legyen. Minél kevesebb a rossz él, annál nagyobb kell, hogy legyen a rátermettsége a kérdéses megoldásnak. Egy színezés rátermettsége így lehet például a rossz élek számának a minusz egyszerese, így minden jó színezés maximális rátermettségű lesz. $f(e) = -k$, ahol a gráf éleinek a száma = e; a csúcsok száma = n; megegyező színű pontokat összekötő élek száma = k; minden jó színezésnél $f(e) = 0$. (A 10.2. és a 10.3. ábrán ugyan azon gráf kétféle színezését láthatjuk.)

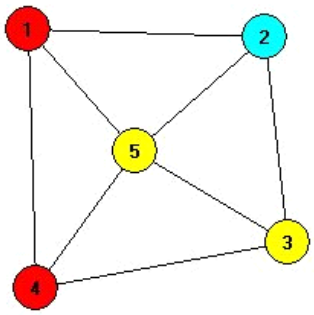
A kódolás és az operátorok akkor megfelelőek, ha rátermett megoldásokból kiindulva a leszármazott megoldások is hasonlóan jó tulajdonságokkal rendelkeznek. Feltéve, hogy a



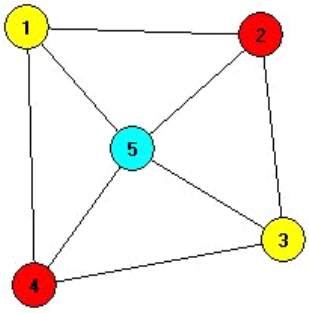
10.1. ábra. A genetikus algoritmus lépései

kódolás jó, nem mindegy, hogy az operátorainkat mely megoldásokra alkalmazzuk amikor a soron következő populációt szeretnénk előállítani. Ezeknek a szülőknek a kiválasztása a *szelekciós operátor* feladata. Többféle szelekciós algoritmust ismerünk, mint például rátermettség-arányos szelekció, párverseny szelekció, rulettkerék szelekció, rangsorolás stb. Nézzük ezek közül az első kettő működését:

- *Rátermettség-arányos szelekció*: A módszer valószínűségi mintavételt használ. Egy megoldás kiválasztásának a valószínűsége annál nagyobb, minél nagyobb a rátermettsége a populáció rátermettségi átlagához képest. A populáció minden e elemére a kiválasztódás valószínűségét a következő módon adhatjuk meg:
  $P(e) = \frac{f(e)}{n*f(Pop)}$, ahol $f(e)$ a rátermettség értéke, n a populáció elemszáma és $f(Pop)$ a populáció tagjainak átlagos rátermettsége.
- *Párverseny szelekció*: Válasszunk ki a populációból két megoldást teljesen véletlensze-



10.2. ábra. Egy gráf 3-színezése, nem megfelelő megoldás, vannak azonos színű csomópontokat összekötő élek



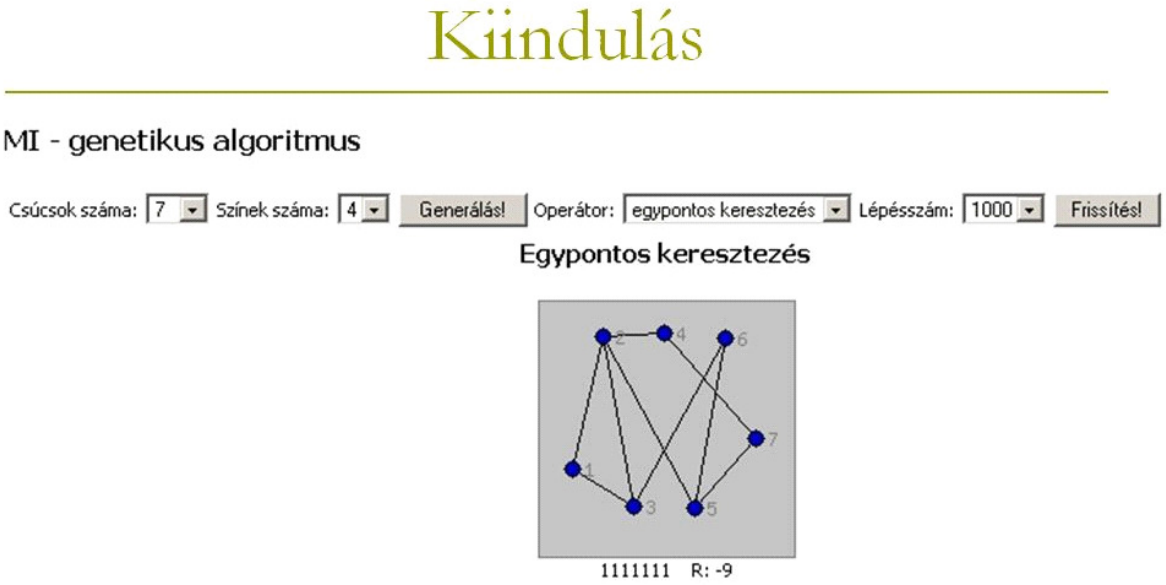
10.3. ábra. Egy gráf 3-színezése, jó megoldás

rűen és a szelekció által kiválasztott elem legyen a kettő közül a rátermettebb. Ugyanez a technika általánosítható úgy, hogy nem kettő, hanem több elem győztesét választjuk ki.

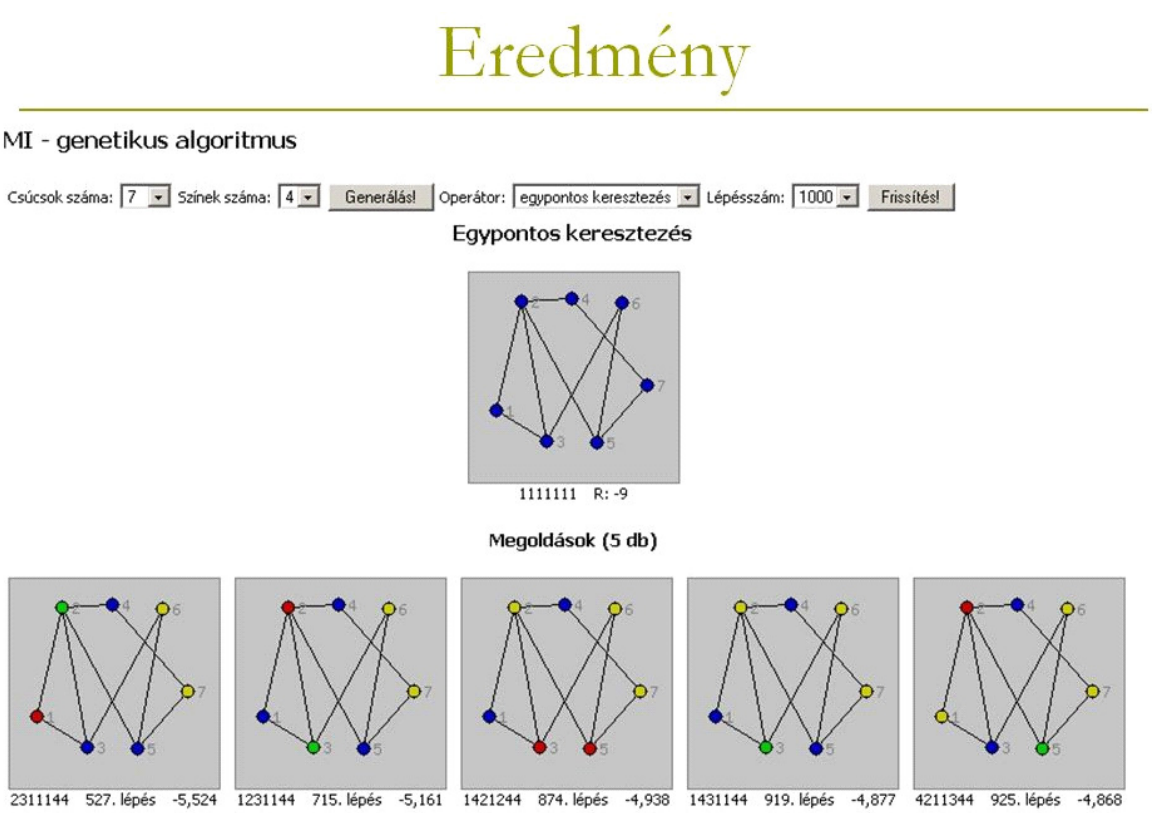
A kódokon alkalmazott operátorokat szokás két csoportba sorolni. Az elsőbe tartoznak azok, amelyek több kiindulási megoldásból (szülőből) állítanak elő újabb megoldást vagy megoldásokat a szülők kódjainak kombinációja segítségével. Ezt az operátort *rekombináció*nak nevezzük. Ilyen művelet például az egypontos keresztezés, amely esetében véletlenszerűen választunk az egyedben egy keresztezési pontot, és a keletkezett fél kódokból új megoldást rakunk össze.

A másik csoportba tartoznak azok, amelyek egy megoldásból állítanak elő egy újabbat, az ilyen operátort *mutáció*nak nevezzük. Ilyen művelet például, ha véletlenszerűen választunk ki pozíciókat és változtatunk meg. Az alábbi 10.4. ábrán egy gráfszínezési probléma kiindulási gráfját és a 10.5. ábrán pedig a genetikus algoritmus végrehajtása utáni eredményt láthatjuk.

(Megjegyzés: A genetikus algoritmus működésének bemutatásához futtassa a house.exe fájlt. A fejezethez tartozó feladatok a GAFuggveny.pdf, GAGrafszinezes.pdf, GAL-rendszerek.pdf, GAMatlab1.pdf, GAMatlab2.pdf, GATrans-GA.pdf fájlokban találhatók.)



10.4. ábra. A gráf, melynek csomópontjait megfelelő színnel kell ellátni



10.5. ábra. A genetikus algoritmus futásának eredménye

11. fejezet

Robotika

11.1. Alapok

Már az ókorban felmerült az igény, hogy az ember munkáját valamilyen géppel segítsük, bizonyos fázisokban az embert helyettesítsük. Ez azt is jelenti, hogy természeti környezetünkben kellene helytállnia egy gépnek, illetve a programjának. A környezetben nem adhatunk meg mindent pontosan az utolsó milliméterig, grammig, így a programot nem csupán bizonyos állapotokra, hanem igen gazdag lehetőségekre kell felkészíteni. Elvégre nem állhat meg egy teljes gyár, ha az egyik szerelőszalag fél centivel odébb állította meg a munkadarabot. A mesterséges intelligencia alkalmazásának az egyik legfontosabb és leglátványosabb területe a robotika.

Néhány szó a robotika szó eredetéről: Maga a robot szó 1921-ben Carel Capek Rossum Univerzális Robotjai című színdarabjában fordul elő elsőként. A robota szó csehül munkát jelent. Capek robotja önálló, döntésre képes eszköz, amely felülkerekedik alkotóján, és rabszolgasorba süllyeszti az embert.

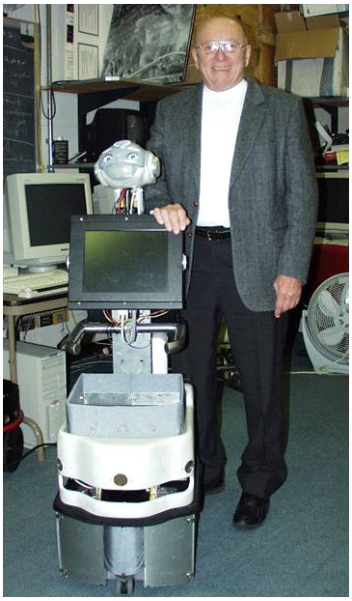
Roboton általában olyan eszközt, berendezést értünk, amely az ember fizikai és/vagy szellemi munkájához hasonló tevékenységet végez. Ehhez rendelkezniük kell számos, sok esetben az intelligencia elemei közé tartozó tulajdonsággal:

- tudás,
- emlékezet,
- tanulási képesség
- döntéseken alapuló közlési-cselekvési képesség stb.

A robotok egyik legfontosabb tulajdonsága a helyváltoztatási képesség, illetve esetlegesen annak hiánya. Ez alapján megkülönböztethetünk mobil és statikus robotokat. Előbbiek közé tartoznak az androidok, az animatok, az ember nélküli járművek, szórakoztató robotok és az általános autonóm robotok. A háztartási és ipari robotok – elsősorban a robotkarok – általában statikusak, de ez nem követelmény. Speciális robotok a nanorobotok, melyek a fizikai és kémiai területek határmezsgyéjén találhatók.

11.2. Egy kis történelem

Néhány érdekes korábbi automatizálási példától eltekintve, az első valódi robotszabadalmat C. W. Kennward angol feltaláló nyújtotta be 1954-ben. A mai ipari robotok története mégis inkább J. Engelberger (11.1. ábra) és G. C. Devol amerikai irányítástechnikai és elektronikai szakértők nevéhez fűződik, akik először éreztek rá a robotokban rejlő üzleti lehetőségekre. Engelbergernek a Consolidated Controls néven megalakuló, majd később Unimation néven működő cége 1960-ban gyártotta le az első ipari robotot Unimate (11.2. ábra) néven. A gépet a Ford Motor Co. már 1961-ben megvásárolta (fröccsöntő gép kiszolgálására használta fel, egészen az 1980-as évekig). Ugyanebben az évben jelent meg Devol robotszabadalma. Az Amerikai Egyesült Államokban 1962-ben, másodikként az American Machine and Foundry cég is megjelent egy gyártmánnyal (Versatran), s tíz év múlva, 1972-ben már 12 vállalat gyártott ipari robotot az USA-ban. Az ipari robotok fejlődésében tehát az első jellegzetes fázis az 1960-as évekre tehető. Míg a finomabb szabályozású robotok ekkor hidraulikus szervoirányítással rendelkeztek, velük párhuzamosan terjedtek a „pick and place” manipulátorok, amelyek az emberi munkaerőt monoton és fárasztó munkákban helyettesítették a nagy tömegben ismétlődő anyag- és munkadarab-mozgatási feladatok ellátásával.

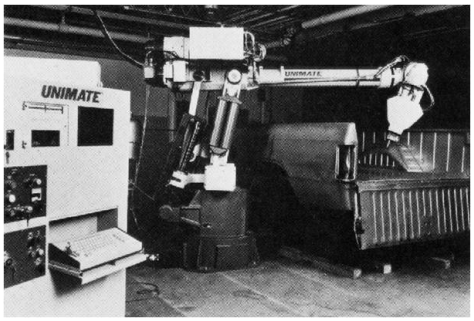


11.1. ábra. J. Engelberger, a robotok egyik atyja

Az első technológiai robotalkalmazás viszonylag korán, 1966-ban jelent meg a Trallfa norvég mezőgazdasági üzem fejlesztésének köszönhetően, egy humanoid karszerkezetű festőrobot használatával. Még az 1960-as évtized végén kiterjedt a technológiai alkalmazások köre az ívhegesztésre is.

A második jelentős fejlődési fázis az 1970-es évekre tehető japán licencvásárlásokkal és európai cégek megjelenésével, újabb műszaki megoldásokkal, s a robotalkalmazás körének jelentős bővülésével jellemezhető.

Mérföldkövek:



11.2. ábra. A robotok egyik első megvalósított darabja: Unimate

- Kawasaki 1972-es licenc-vásárlása, és a gyártósor felszerelése a Nissan Motors-nál;
- a svéd ASEA IRB-6 (1973) és IRB-60 (1974) villamos meghajtású robotjainak megjelenése és elterjedése az ívhegesztéses alkalmazásokban;
- az amerikai Cincinnati Milactron NC-gyártó vállalat megjelenése az első teljesen számítógép-irányítású CGA vagy T3 robottal, elsősorban fúrási, marási, illesztési műveletek végzésére;
- az irányítási rendszer korszerűsödése az ASEA-nál lehetővé tette köszörülési, sorjázási, polírozási feladatok robotizálását;
- az ún. playback technika kifejlesztése a Trallfa cégnél jelentősen javította a robotok festésre való felhasználhatóságát;
- az európai autógyárak saját robotfejlesztéseinek kezdete (Volkswagen, Renault, Fiat);
- új típusok kifejlesztése és gyártása belföldi piacra Japánban (Hitachi, Kawasaki, Yasakawa, Oainichi Kiko).

A robotok történetének harmadik nagy korszakát a robotok intelligenciájának növekedése, az érzékelési, a hajtási és az irányítási rendszerek nagymértékű továbbfejlesztése jellemzi. A korábbi komplikált kinematikai áttételek részben kiküszöbölhetőkké váltak a hagyományos váltóáramú motoroknál előnyösebb egyenáramú, illetve léptetőmotorok alkalmazásával. A robotok irányítástechnikájában elterjedtek az ún. hierarchikus megoldások: ezek központi vezérlőegységen és lokális hajtásszabályozókon alapultak. Megjelentek az első robotprogramozási nyelvek. Általánossá vált a legkülönbözőbb szenzor-rendszerek felhasználása, és azok jeleinek kezelése a robotvezérlés részéről. E műszaki fejlesztéseknek köszönhető, hogy a robotok alkalmazási köre a hagyományos anyagmozgatási, szerszámgép-kiszolgálási, festési és hegesztési funkciókon túl kibővülhetett a laboratóriumi, mélytengeri és űrkutatásbeli, mezőgazdasági, és főleg a szerelési folyamatokkal is.

11.3. Robot definíció

Végül eljutottunk a mai értelemben vett ipari robot fogalmáig, amelyet azonban a különböző országokban ma még kissé más és más módon határoznak meg. Példaként idézzük a

Nemzetközi Szabványosítási Szervezet definícióját.

A **Nemzetközi Szabványosítási Szervezet (ISO) robotdefiníciója**: „Az ipari robot automatikus helyzetszabályozással ellátott, újraprogramozható, többfunkciós, több szabadsági fokú manipulátor, amely képes anyagok, szerszámok, alkatrészek vagy speciális készülékek mozgatására változtathatóan programozott mozgások sorozatán át, különféle feladatok megoldása céljából. Egy vagy több karja van, mely(ek) csuklóban végződik (végződnek). Irányítóegysége memóriát tartalmaz, és gyakran érzékelő és adaptációs eszközöket alkalmaz, hogy figyelembe tudja venni a környezetet vagy a körülményeket. Ezeket a többfunkciós gépeket általában ismétlődő feladatok elvégzésére tervezik, és alkalmazhatók más feladatok ellátására is, a berendezés állandó átalakítása nélkül."

Az eddigi fejlődés megfigyeléséből levonható egyik következtetés, hogy a robotizáció alakulásában a legfontosabb tényező maga a műszaki fejlesztés és annak eredményessége. Az alkalmazások körének további bővülését egy-egy konkrét típus esetében általában nem gazdasági korlátok akadályozták meg, hanem magának az adott konstrukciónak a műszaki alkalmatlansága komplikáltabb feladattípusok megoldására. Amint a műszaki fejlesztésben döntő áttörés következett be valamilyen területen, s ez műszakilag lehetővé tette az alkalmazási területek kibővítését, a gyakorlat mindig élt is az új lehetőségekkel.

A robotot tekinthetjük egy aktív *mesterséges ágens*nek, aminek környezete a teljes fizikai világ. A robotok azon beavatkozók és érzékelők alapján különböztethetők meg egymástól, amikkel fel vannak szerelve.

1. **Érzékelők**: az érzékelés eszközei

   - *Önérzékelés*: Az emberhez hasonlóan a robotoknak is van egy saját, proprioceptív (saját belső érzékelés) érzete, ami megmondja neki, hogy saját csuklói hol találhatók. A csuklókhoz tartozó kódolók nagyon pontos adatokat szolgáltatnak azok szögéről és kiterjedéséről. Amennyiben mozgás alatt a kódoló kimenetét visszacsatoljuk a mechanizmus vezérlőjéhez, a robot az embernél sokkal nagyobb pontossággal tud pozicionálni.
   - *Erőérzékelés*: Noha a robotok az embernél sokkal pontosabban képesek érzékelni és vezérelni saját csuklóik pozícióját, nagyon sok olyan probléma marad, amelyeket csupán pozícióérzékeléssel nem lehet megoldani. Például tekintsük azt a feladatot, amikor borotvapengével az ablaküvegről kell lekaparnunk a festékmaradékot. Az összes festék leszedéséhez az szükséges, hogy a pozicionálás az üvegre merőlegesen, ezredmilliméter pontossággal történjen. Egy milliméteres pozicionálási hiba azt okozhatja, hogy a robot egyáltalán hozzá sem ér a festékhez vagy betöri az üveget.

     Ez és sok más érintkezést igénylő feladat, így az írás, az ajtónyitás, a járműösszeszerelés szükségessé teszi az erő pontos meghatározását, vezérlését. A pontos szabályozáshoz erőérzékelőkre van szükség. Ezeket a villanymotorokat rendszerint a manipulátor és a beavatkozók közé helyezik el, és ezek az erőt és a nyomatékot hat irányban tudják érzékelni.
   - *Taktilis érzékelés*: A taktilis érzékelés az emberi tapintás robot változata. Egy papír pohár felemeléséhez vagy egy pici csavarnak a becsavarásához a saját

maga érzékelésénél többre van szükség. Az alkalmazandó erőnek elegendően nagynak kell lennie ahhoz, hogy a pohár ne csússzon el, de nem lehet túl nagy, nehogy összenyomja azt. A csavar megfogásához pontosan tudni kell, hogy az azt érintő ujjakhoz képest hol helyezkedik el. Mindkét esetben a taktilis érzékelés (tapintásalapú érzékelés) az, ami a szükséges információkat biztosítja.

- *Hanglokátor*: A Sonar(hanglokátor) mozaikszó, a Sound Navigation and Ranging, hang szerinti tájékozódás és besorolás rövidítése. A hanglokátor hasznos információkat szolgáltat a robothoz nagyon közeli tárgyakról, gyakran az ütközések elkerülését biztosító vészjelző szerepét tölti be. Időnként a robot tágabb környezetének feltérképezésére is használják. A hanglokátor azt az időt méri, amely alatt az érzékelő által kibocsátott hangimpulzus a tárgyról visszaverődve visszaérkezik.

2. **Beavatkozók**: a cselekvés eszközei

   Beavatkozónak nevezzük mindazokat az eszközöket, amelyek a robot irányítása alatt valamilyen hatást gyakorolnak környezetükre. A fizikai világra történő hatás gyakorlásához a robotokat olyan működtetőkkel kell felszerelni, amelyek a szoftver parancsokat fizikai mozgássá alakítják. Maguk a működtetők tipikusan villanymotorok, hidraulikus vagy pneumatikus munkahengerek. A beavatkozókat alapvetően kétféleképpen használhatjuk:

   - a robot helyzetének saját környezetében való megváltoztatására,
   - valamely objektumnak a környezetben történő mozgatására.

A robottechnika alapjairól az érdeklődő olvasó [8]-ben olvashat bővebben.

11.4. Robottípusok és alkalmazásuk

A gyártásautomatizálás terén ható pusztán piaci kényszereken kívül ma a technológiai fejlődéssel párhuzamosan egyéb, a robottechnika alkalmazását szintén serkentő körülmények is megjelentek. Az emberi munkaerő, egészség mint érték fenntartásának költségei jelentősen megnőttek. Ebből a szempontból az új, emberre és környezetére veszélyes, ártalmas technológiák, valamint a nagy tisztaságú, az emberi életfolyamatok által veszélyeztetett technológiák megjelenése is döntő. Tipikusan ilyen terület a korrózióvédelem, galvánozás, festés, ragasztás, a mérgező anyagaik kipárolgásai miatt veszélyes hulladékok kezelése, atomerőművek üzemeltetése, és egyéb sugárveszélyes technológiák, biotechnológia, baktérium- és vírustenyészetekkel végzett munka, nehézipari, nehézüzemi, melegüzemi alkalmazások (acélipar), nyersanyag kitermelése az emberen maradandó károsodást okozó környezetben (pl. mélytengeri robotok), űrtechnológia, s minden olyan környezetben végzett munka, amelyben az emberi élet fenntartásának költségei extrém magasak. Tipikus példák továbbá speciális élelmiszeripari alkalmazások, termékek nagy tömegben történő feldolgozására (pl. halfeldolgozás), illetve egyenletes munkatempóban a termék romlékonysága miatt nem végezhető idénymunkákra (pl. gyümölcsök csomagolása stb.).

A következőkben a definíciók felsorolása helyet néhány szempont szerint bemutatjuk a robotok csoportjait.

1. **Fejlettségük szerint**:

   I. generációs robotok: 1960-as évek. Felemelés-lerakás típusú feladatokra alkalmazták őket. A környezet változásait vizsgáló külső érzékelőkkel nem igen rendelkeztek. Ha igen, akkor azok csak védelmet biztosítottak. A programozhatóság alacsony szintű volt, a robot mozdulatait a program egyértelműen meghatározta.

   II. generációs robotok: 1970-es évek. Ezek már környezetüket érzékelők segítségével vizsgálják és tevékenységüket a vett jelek alapján a pillanatnyi szituáció figyelembevételével módosítani tudják. Feladataikat magas szintű robotprogramozási célnyelven lehet meghatározni.

   III. generációs robotok: 80-as évektől. Itt már nagyobb szerepet kapnak a mesterséges intelligencia elemei. Az érzékelőktől származó jeleket feldolgozzák, a magukról és a környezetről tárolt modellt képesek intelligens módon, önállóan módosítani, képesek információ kiválasztására és kombinálására. Megjelennek az önálló viselkedési algoritmusok és a döntési rendszerek. E generáció működésére az összetett tevékenység és a feladatok magas szintű, általános megfogalmazása jellemző.

   Az iparban alkalmazott robotok nagy része a második generációba soroltakból kerül ki. A harmadik generációs tulajdonságú robotok főleg kutatási területeken találhatók meg.

2. **Feladatkörük szerint**:

   - *A termelésben használt robotok lehetnek*:
     - Anyagkezelő robotok
       Feladatuk munkadarabok és műveleti eszközök manipulálása (általában a műveletek szünetében).
       Jellegzetes feladatok:
       * adagolás megmunkálógépre,
       * megtartás, forgatás (operációhoz, vagy operáció alatt),
       * áthelyezés program szerint,
       * válogatás, rendezés,
       * szerszámozás, szerszámcsere,
       * mérőeszköz csere.
     - Műveletvégző robotok
       Feladatuk műveleti eszközök operatív mozgatása.
       Jellegzetes alkalmazások:
       * festés, tisztítás, sorjázás,
       * pont- és vonalhegesztés, varratlerakó hegesztés,
       * fúrási, marási műveletek (11.3. ábra),
       * lézeres megmunkálási műveletek,

* láng- és plazmasugaras vágás,
* mérőeszköz mozgatás, mérés kiszolgálás.

11.3. ábra. Maró, esztergáló robot az iparban

– Szerelő robotok
Feladatuk munkadarabok és műveleti eszközök manipulálása, mozgatása. Általában szenzorokkal rendelkeznek.
Jellegzetes feladatok:
  * alkatrészek kiválasztása,
  * alkatrészek orientálása,
  * alkatrészek összeillesztése, helyezése,
  * kötések létesítése.

- *A kutatásban használt robotok (11.4. ábra) többnyire*:
telerobotok, távvezérelt manipulátorok, mobilrobotok.

11.4. ábra. Űrkutatásban használt robot

- *Speciális feladatok megoldására alkalmazott robotok és egyéb robotrendszerek*:
mikrorobotok, gyógyászatban alkalmazott robotok (11.5. ábra), oktatásban használható robotok (11.6., 11.7. ábrák), szórakoztató robotok (11.8. ábra) stb.

11.5. ábra. Gyógyászatban használt robot

11.6. ábra. Lego mindstorms robot építő eszközkészlet I.

3. **Felépítésük szerint**:

   Robotkarok, manipulátorok: Helyhez kötött robotkarok, melyeket egy művelet, vagy műveletsorozat végrehajtására programoznak. A robotkar elemei izületekkel kapcsolódnak egymáshoz. Az izületek és karok elhelyezkedése alapján a következő típusokat különböztetjük meg.

   (a) derékszögű koordinátás kar (11.9. ábra)

   (b) hengerkoordinátás kar (11.10. ábra)

   (c) gömbkoordinátás kar (11.11. ábra)

   (d) SCARA kar (11.12. ábra)

   (e) humanoid kar (11.13. ábra)

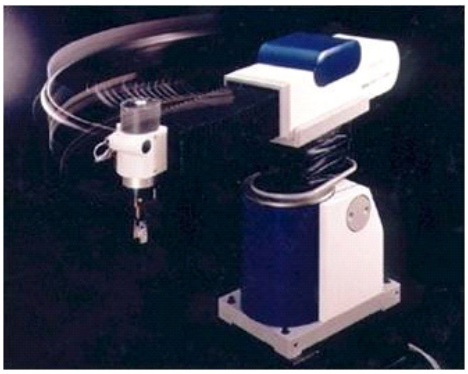
   Mobilrobotok: Helyhez nem kötött robotok. Általában nem ipari szerelési, technológiai vagy anyag-mozgatási feladatokra készülnek. Rendszerint kutatási feladatokat látnak el, ugyanis képesek olyan helyeket is felkeresni, ahová az emberek – valamilyen oknál fogva – nem juthatnak el. A mobilrobot feladata sok esetben nem egzakt

11.7. ábra. Lego mindstorms robot építő eszközkészlet II.

11.8. ábra. A szórakoztató robotok egyik típusa

módon definiált, nemegyszer csak „menj és gyűjts adatokat” típusú. A mobilrobotokat komplex problémák megoldására tervezik. Két nagy csoportjuk a kerekes és a járó robotok.

11.9. ábra. Derékszögű koordinátás kar



11.10. ábra. Hengerkoordinátás kar

11.11. ábra. Gömbkoordinátás kar

11.12. ábra. SCARA kar

11.13. ábra. Humanoid kar

12. fejezet

Virtuális valóság

12.1. Alapok

A virtuális valóság olyan ember-számítógép kapcsolat, amely a valósághű térbeli megjelenítésre és érzékelésre épül, és magas fokú interaktívitásával azt az illúziót adja a felhasználónak, mintha ő valójában részese lenne a számítógép által generált környezetnek.

A virtuális valóság kifejezés két szóból áll, ahol a virtuális látszólagos, nem létező, nem valós jelentéssel bír, míg a valóság azt fejezi ki, hogy egy tárgyat érzékszerveink közül legalább az egyikkel érzékelünk.

12.2. Definíciók

A **virtuális valóság** (virtual reality = VR) olyan számítógépes szimuláció, amely kiterjed szinte valamennyi emberi érzékszervre és azokat aktívan manipulálja. [7]

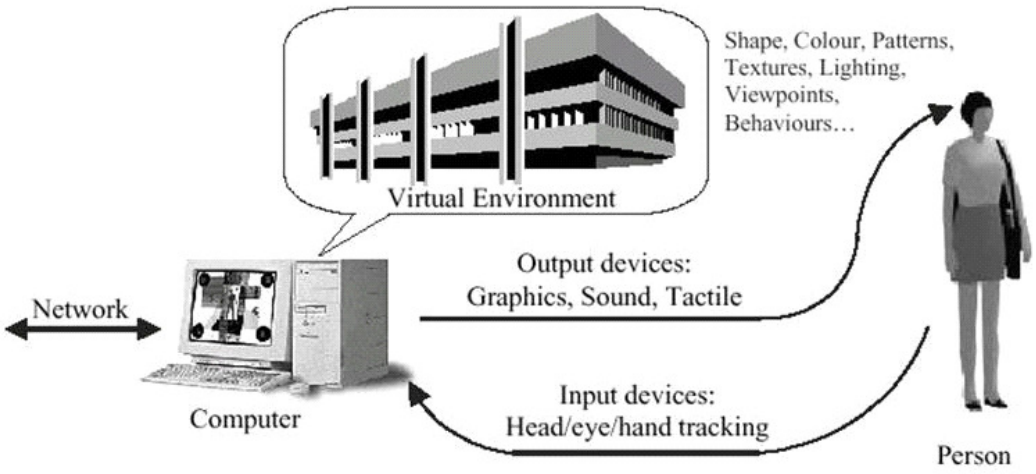
A virtuális valóság rendszer részei:

- az ember
- a számítógép
- a virtuális környezet
- a bemeneti egységek
- a kimeneti egységek
- a hálózat.

Virtuális környezet: olyan számítógépen megjelenő 3 dimenziós modell, mely különböző fizikai tényezők (alak, szín, pozíció, textúra stb.) és részletek megadásával jön létre.

Az egyik legfontosabb kérdés a tárgyak viselkedésének leírása: Leesik, ha leejtik? Bekapcsol, ha megnyomunk egy gombot?

A *valósidejű*, *dinamikus*, *interaktív* képesség tesz különbséget a virtuális valóság rendszer és a 3 dimenziós képek között. (12.1. ábra)



12.1. ábra. Virtuális környezet

12.3. Az érzékelés folyamata

Mivel a virtuális valóság rendszerek kialakításához, használatához szorosan kapcsolódnak az érzékelés különböző formái, ezért röviden kell, hogy érintsük az érzékelés folyamatát.

1. A **látás** hordozza környezetünkről a legtöbb információt.

   A látás manipulálásához

   - valós időben létre kell hozni egy állandóan változó (a nézés irányától is függő) képet,
   - a képet a szembe kell juttatni,
   - a térbeli tájékozódáshoz érzékszervi együttműködést kell szimulálni.

   Az állandóan változó képhez nagy teljesítményű, valósidejű képfeldolgozást kell megvalósítani.

   A térbeli tájékozódásban a látásnak ugyancsak fontos szerep jut. Egy bonyolult visszacsatolási folyamat eredményeként érzékeljük térbeli helyzetünket és ebben a látás, a hallás, az egyensúlyérzet és a tapintás játszik szerepet. Ez az integrált érzékszervi rendszer több millió éves evolúció eredménye, ezért manipulálása nehéz feladat. A legkisebb hiba is furcsa jelenségeket produkál, az illető émelyeg, fáradt, tériszonya van stb. Az ilyen jelenséget *szimulátor-betegség*nek nevezik.

2. A **hallás** útján való érzékelés szimulációja kevesebb gondot okoz, mivel ezzel kapcsolatban jóval több információ, kutatási eredmény áll rendelkezésre, mint a látás esetén.

3. A **tapintási és kinesztetikus ingerek** szimulációja területén jelentős a lemaradás. Ide tartozik a bőrre ható nyomás, a hőérzet és a fájdalom szimulációja. Ezeken a területeken még nagyon sok munkájuk van a kutatóknak, fejlesztőknek.

12.4. A virtuális valóság rendszerben használható eszközök, megoldások

1. **Bementi eszközök**

   A felhasználó irányítani tudja a nézőpontot és kapcsolatba tud lépni a virtuális környezettel. Fontos, hogy a természetes kommunikációhoz a lehető legközelebb álljanak.

   - A test mozdulatainak nyomon követése:
     - A legelterjedtebb eszközök rendelkeznek egy küldővel, amely mágneses mezőt hoz létre, melynek egyedi szenzorait a felhasználó testére erősítik, és ez információkat tud szolgáltatni a pozícióról és az elfordulásról.
     - Így a számítógép nem csak azt tudja, hol helyezkedik el a fej vagy a kéz, hanem azt is, hogy az merre mutat vagy néz.
     - Vannak eszközök, amelyek ultrahangos szenzorok, optikai érzékelők vagy kamerák segítségével érzékelik a kívánt információkat.
   - Mozgás a virtuális környezetben:

     Használhatunk ún. belemerítő VR irányító eszközöket. Ez azt jelenti például, hogy a mozgást nyomon követő szenzorok a felhasználó lábára erősíthetők, majd egyhelyben gyalogolva szimulálható a valós mozgás (séta, futás).
   - Asztali VR irányító eszközök:

     Sokkal több fejlettebb és pontosabb mozgást előidéző asztali irányító eszköz létezik (12.2. ábra):
     - spacemouse
     - spaceball
     - cyberpuck
     - joystick

12.2. ábra. Mozgást előidéző asztali irányító eszközök

A virtuális térben – ahogy a valóságban is – 3 szabadsági fok van az irányokra (mozgásokra) és 3 a forgásokra. Az irányok általában x, y, z, míg a forgások csavarodás, elhajlás és elfordulás. A 6 szabadsági fokú eszközök lehetővé teszik, hogy a felhasználó nézőpontját bármerre elmozdítsa, vagy fordítsa.

- Kontaktus a tárgyakkal:

  Az egyik leggyakrabban használt eszköz az adatkesztyű (dataglove), amely esetében a kéz gesztusait használhatjuk a kapcsolatteremtésre. Az ilyen kesztyűk (12.3. ábra) többsége száloptika segítségével érzékeli az ujjak behajlítását vagy ujjvégekkel történő kapcsolást (két ujjbegy összeérintését) tesz lehetővé.

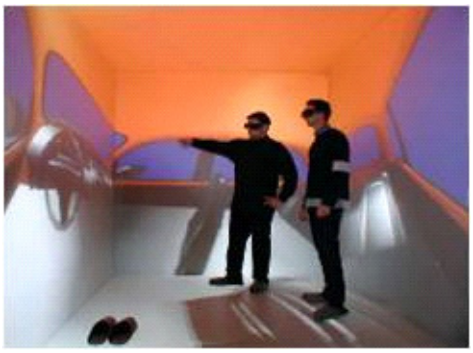
12.3. ábra. Adatkesztyű

A jövőben cél a bemeneti eszközök használhatóságának a növelése. Így például hasznos lehet adatkesztyű és szenzor nélkül a fej és a kéz mozgásának a meghatározása vagy a beszéd és az egyértelmű gesztusok felismerése.

2. **Kimeneti eszközök**

   A kimeneti eszközök a környezetnek az érzéki megtapasztalását teszi lehetővé. Olyan eszközöket terveznek és fejlesztenek ebben a körben, amelyek összeegyeztethetők az emberi érzékeléssel.

   - Vizuális eszközök
     - Belemerülés
       Alapvetően kétféle lehetőséget említhetünk meg. Az egyiknél projektorok és tükrök segítségével vetítjük magunk köré a virtuális világot. A felhasználó a középpontjában áll, így teljesen el tud merülni a virtuális térben, mivel látóterét teljesen lefedi a virtuális környezet (12.4. ábra). A másik esetben egy ún. Head Mounted Display-t (HMD) használhatunk (12.5. ábra), amely csak egy felhasználó esetén alkalmazható. Az eszköz két vékony képernyőből áll, egyik az egyik, a másik a másik szemnek. A képernyőkön kívül szükség van lencsére is ahhoz, hogy a szem megfelelően tudjon fókuszálni (sztereóban) a látottakra. Ezzel az eszközzel a teljes látóteret nem tudjuk lefedni.
     - Nem belemerítő technika
       Ide tartoznak a standard asztali számítógépek mint megjelenítők.
     - Sztereó kép létrehozása
       Ahhoz, hogy mindkét szem a számára megfelelő képet kapja, speciális hardverre van szükség. Ez lehet folyadékkristályos rekesz szemüveg (LCS),



12.4. ábra. Munka a virtuális térben

amellyel a felhasználó mélységet is lát a képernyőn. A 3D-s, PC-s videókártyákra alapozva kifejlesztettek egy olcsóbb megoldást is, amely egy vagy két projektor polarizált fényét és hozzá olcsó, polarizált szemüveget használ. Ez egy passzív sztereó eszköz, mivel nincs benne elektronika. Ilyen a 3D anaglyph szemüveg (12.6. ábra), amely esetében a képszeparáció színszűrő segítségével történik. A szűrő például piros és cián színű.

12.5. ábra. Head Mounted Display

- Audió eszközök

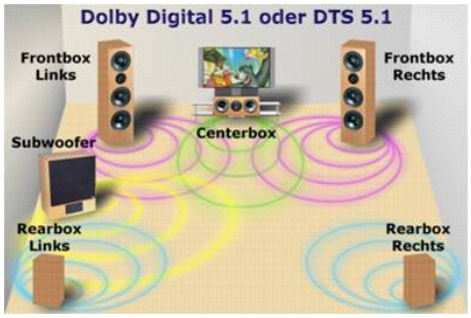
  Ezek az eszközök a hangok generálására és a térbeli hely meghatározáshoz szükségesek. A tökéletes virtuális környezet élmény létrehozásához szükségesek a hangefektek, melyek erősítik a valóságérzetet. A PC-s hangkártyák egyre nagyobb mértékben támogatják a surround hangzást. (12.7. ábra)

- Érzékelő-tapintó eszközök

  Ezen eszközök segítségével lehetőségünk nyílik a virtuális környezet érzékelésére is. Használhatunk pl. ún. erő-visszacsatoló joystick-et vagy mellényt (12.8. ábra). Az elektromos tapintást támogatják a különböző kesztyűk. Ha a hatást fokozni akarjuk ún. kevert valóságot (mixed reality) is létrehozhatunk, például ilyen lehetőséget alkalmaznak a repülés szimulátoroknál, ahol keverednek a virtuális (az ablakból látott látvány) és a valós elemek (irányító eszközök, műszerek).

- Egyéb eszközök, lehetőségek

12.6. ábra. 3D anaglyph szemüveg



12.7. ábra. Hanghatás eszközei

Itt megemlíthetjük a szagláshoz kapcsolódó ingereket, de ez a terület még gyerekcipőben jár, még sokrétű kutatásra van szükség a kellő hatás eléréséhez. De megemlíthetjük a teljes test mozgatását, amit a repülőgép szimulátorokban már régóta használnak.

3. **Hálózat**

A hálózat szerepe, hogy megteremtse annak a lehetőségét, hogy ugyanazt a virtuális környezetet többen használják a saját számítógépükön, akár az interneten keresztül.

12.8. ábra. Erő visszacsatoló mellény

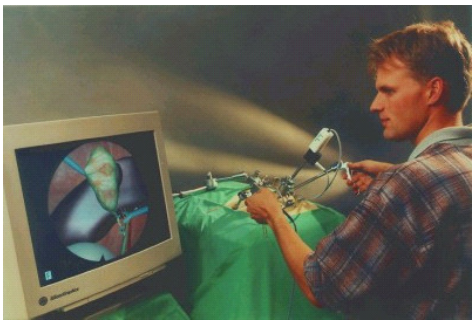
12.5. Alkalmazási területek

Biológia

Az egyik kiemelkedő terület a génsebészet, melynek a célja új, jobb és életképesebb fajok létrehozása a növénytermesztésben. A géneket a VR eszközeivel megfelelően tudják ábrázolni és tanulmányozni.

Orvostudomány

Többek között megemlíthető a virtuális valóság vírusok és rendellenességek gyógyításában játszott szerepe. Emellett a sebészetben (12.9. ábra), a sebészeti beavatkozások megtervezésében nyújthat segítséget a VR. Orvostan hallgatók virtuális gyakorlatokon vehetnek részt, ahol virtuális emberi testeken végezhetnek műtéteket, következmények nélkül.



12.9. ábra. Műtét a virtuális valóság segítségével

Pszichológia

A fóbiák gyógyítását említhetjük első helyen. A pácienseket szembesíteni tudják legsúlyosabb félelmük tárgyaival (magas épületek, hidak, bogarak stb.). De nemcsak a betegek

szemléletének a megváltoztatására tudják felhasználni, hanem az emberi magatartás tanulmányozására is. (12.10. ábra)

12.10. ábra. Fóbiák gyógyítása

Kémia

A bonyolult, nehezen elképzelhető és csak drága műszerekkel vagy költséges eljárásokkal megnézhető molekulák modelljeit lehet tanulmányozni. (12.11. ábra)

Gyógyszervegyészet

A VR segítségével a gyógyszereknek az emberi testre gyakorolt hatását vizsgálhatjuk.

Oktatás

A legelső VR alkalmazások is oktató céllal készültek, amelyet a hadsereg alkalmazott, igaz nem túl békés célokra. Az oktatásnak nincs olyan területe, ahol ne lenne szükség a 3 dimenziós megjelenítésre, korszerű szemléltetésre és magas fokú interaktivitásra. A VR berendezések és oktató programok elég költséges mivolta miatt az egyéb oktató módszerekkel szemben egyenlőre még hátrányban vannak.

12.11. ábra. Munka egy virtuális kémiai üzemben

Virtuális szolgáltatások

Itt több alkalmazási kört is megemlíthetünk, amelyek számos területen nyújthatnak könnyítést számunkra.

- üzleti alkalmazások (pl. elektronikus kereskedelem),
- reklám,

- videokonferencia (12.12. ábra),

12.12. ábra. Virtuális konferencia

- távoktatás,
- szórakoztatóipar – játékok, filmek (12.13. ábra) stb.

12.13. ábra. Virtuális valóság eszközökkel megvalósított mozi film

Mindezeken kívül még számos alkalmazási területet említhetünk, így az űrkutatást, hadi ipart, tervezést, fizikát, geográfiát, meteorológiát, informatikát, médiakutatást, nyelvészetet stb.

12.6. Virtuális valóság szerkesztő program

A virtuális valóság szerkesztő eszközök közül egy programozási nyelvet emelnénk ki. Az érdeklődő olvasó számos a témához kapcsolódó könyvet találhat, így például [3], [6] és [14]-et. A VRML nyelv (Virtual Reality Modeling Language) egy általános, szöveg alapú nyelv, amelyet speciálisan 3 dimenziós objektumok készítésére terveztek. Különböző platformokon használható, így UNIX, Mac vagy Windows rendszer alatt. Segítségével készíthetünk három-dimenziós szövegeket, speciális 3D világokat építhetünk fel és különböző 3D jelenetsorokat jeleníthetünk meg. Minden információ egyetlen egy szöveges fájlban tárolódik. A VRML programok elkészítéséhez használhatunk segítség képpen számos szerkesztő programot (pl. LVIEW, AC3D, RenderSoft, VRML Editor stb.). A VRML segítségével definiálhatunk:

- egyszerű mértani alakzatokat (kocka, gömb stb.),
- ezeknek felületet,
- tulajdonságokat adhatunk meg,
- átlátszóságot, fény visszaverő képességet szabályozhatunk,
- különböző transzformációkkal (mozgatás, forgatás) elhelyezhetjük azokat végleges helyükön,
- különböző szögekből bevilágíthatjuk,
- más terekre mutató hivatkozásokkal komplex, sok részből álló világokat hozhatunk létre.

A felhasználók a 3D világokat böngészővel vagy segédprogrammal kiegészített webböngészőből nézhetik meg és barangolhatják be (pl. Cosmo Player, Cortona Player, Live3D, VRML 2.0 Viewer stb.).

13. fejezet

Gépi látás

13.1. Alapok

Közismert tény, hogy érzékszerveink közül szemünk közvetíti számunkra környezetünkről a legtöbb információt. Az is közismert azonban, hogy látásunk nem csupán szemünknek köszönhető: bonyolult, sok összetevőből álló együttműködés eredményeként ismerjük fel a tárgyakat, tájékozódunk, olvasunk. Talán akkor tévedünk a legkisebbet, ha úgy tekintjük, hogy látásunk idegrendszerünk komplex megnyilvánulásainak egyike, melyben az érzékelés és a feldolgozás összemosódik. Ha mesterséges látásról hallunk, akkor általában annak eszközei és a legismertebb alkalmazási példák – optikai karakterfelismerés, minőségellenőrzés, robotvezérlés – jutnak eszünkbe. Ma még nem tartjuk természetesnek, hogy a mesterséges látás ugyanolyan összetettségű információgyűjtést és feldolgozást jelenthetne, mint az élőlények esetében.

A mesterséges látást célzó számítógépes képfeldolgozás a matematika, elektronika és számítástechnika egyik legizgalmasabb, leggyorsabban fejlődő alkalmazási területe. A látás teljes automatizálása egyelőre csak távlati cél, de a kutatást és a fejlesztést különösen izgalmassá teszi, hogy a lehetőségek messze túlmutatnak a biológiai látás lehetőségein.

A legerősebb számítógépnek is nagyon sok időre van szüksége, hogy egyetlen statikus képet interpretáljon. A Neumann-elvű számítógépek képtelenek erre a teljesítményre. A jövőben a technika és a számítástudomány fejlődésével ezen probléma megoldása lehet a valós idejű képfeldolgozás igényeihez szabott kép-processzor, a párhuzamos struktúrájú számítógépek, a kvantum számítógépek megjelenése, speciális eljárások, algoritmusok használata.

A mesterséges látás kezdettől fogva kiemelt szerepet játszott a számítástechnika gyakorlati alkalmazásában. A kezdeti dedikált (drága) alkalmazások korszakán túljutottunk, s ma már pl. a kiadványszerkesztés, automatikus minőségellenőrzés, orvosi képfeldolgozás hétköznapi alkalmazásnak számít. A fejlődést az is jelzi, hogy a képi információ szerepe a tárolt, továbbított adatformátumok közt folyamatosan nő.

A gépi látás témakörében a magyarországi fejlesztők már néhány világhíres termékkel is előrukkoltak. Ilyen például az optikai karakterfelismerés területén a Recognita programrendszer. Számos olyan megoldás készült ipari, orvosi, mezőgazdasági, biztonságtechnikai területekre, amelyek nemzetközi összehasonlításban is kiemelkedőek.

Nagy általánosságban azt is mondhatnánk, hogy minden olyan információközlés látás, mely számunkra a 3D (háromdimenziós) környezet objektumainak alakjáról, helyéről adatokat közvetít. A továbbiakban ugyan elsősorban a megszokott (a fény visszaverődésén és érzőkelésén alapuló) látásról lesz szó, érdemes azonban elöljáróban egy nagyon rövid kitérőt tenni, mi minden tartozhat még ebbe a témakörbe. (A továbbiakban egy összefoglalóját adjuk meg ennek a területnek. Az érdeklődő olvasó például [17]-ben és [11]-ben olvashat további részleteket.)

13.2. A gépi látás területei

Jelenlegi képalkotó eszközeink többsége felületi információ felhasználásával működik, és mi magunk is így látunk. Ez azt jelenti, hogy az objektumok belsejéről többnyire nem áll rendelkezésünkre adat. Ezért látáson eleve az objektumok felszínének érzékelését értjük. Pedig ez nem szükségszerű; elég, ha arra gondolunk, hogy átlátszó, illetve áttetsző testek esetében a belső szerkezetet is láthatjuk. Ha látásunk pontszerű kölcsönhatáson alapulna, nem pedig valamilyen közvetítő közegen keresztül hozzánk jutó sugárzás érzékelésén, akkor akár közvetlenül láthatnánk a dolgok belsejét, szerkezetét. Jelenlegi ismereteink szerint ilyen fizikai kölcsönhatás nem létezik, így ez a gondolat látszatra felesleges kitérő. Valójában azonban a látás folyamatában nem különíthető el, hogy mi köszönhető a fizikai kölcsönhatásnak, és mi a nyert adatok feldolgozásának. Megfelelő technikai eszközök és feldolgozó algoritmusok alkalmazásával ma is lehetséges a térbeli objektumok belsejének rekonstruálása, legalábbis ami az anyageloszlást és a fizikai tulajdonságokat illeti. Így könnyelműség lenne a mesterséges látást azonosítani azzal a felületlátással, amit a biológiai látás esetében megszoktunk: a mesterséges látás akár valódi térbeli látás is lehet.

Nézzünk egy másik példát. Az élőlények többsége rendelkezik a színlátás képességével, tehát a fényt viszonylag széles frekvenciatartományban érzékeli. A biológiai látás azonban messze nem nyújtja azokat a lehetőségeket, melyeket például a műholdak készítette többsávos képek számítógépes feldolgozása nyújt. Ez ugyanis a frekvenciafüggést leíró, spektrális reflektanciafüggvény osztályozása útján anyagfajták kvantitatív meghatározására is alkalmas. Továbbá az élőlények egy részénél nemcsak a látható fény, hanem pl. ultrahang terjedésén, illetve statikus elektromágneses tér torzulásának érzékelésén alapuló „látás” is kifejlődött. Technikai eszközeink ennél jelenleg is jóval többet „tudnak”: az elektromágneses hullámok teljes spektrumán kívül különböző elemi részecskék áthatoló sugaraival is látnak, így a megfigyelt objektumok mérete molekuláris szinttől a csillagrendszerekig terjedhet.

13.3. Digitalizálás

A számítógépes képfeldolgozás előfeltétele, hogy a valós világ objektumainak képét számítógéppel kezelhető adatokká lehessen alakítani. Ezt az átalakítást nevezzük *digitális képalkotás*nak, más szóval: digitalizálásnak. A digitális képalkotás célja a valóságos tér (x, y, z) koordinátájú pontjaiból t időpillanatban visszaverődött, a színű összetevőket tartalmazó – a megfigyelés irányától (λ) is függő erősségű – fénysugarak által közvetített természetes kép számítógépes megfelelőjének létrehozása. Ennek a műveletnek az $e(x, y, z, t, \lambda, n)$ folytonos

függvény adja a bemenő adatait, kimenetén pedig $[n * m]$ darab képpontot tartalmazó tömb jelenti a kimenő adatokat (ahol n és m a kép „méretei”, azaz a sor illetve oszlop irányú képpontszám). Az $f(x, y)$ képfüggvény értelmezési tartománya folytonos; a függvény az (x, y) sík azon pontjain van értelmezve, ahová a leképezés történt. (A TV-technikában szokásos megjelenítési mód miatt a képsíkon az x-tengelyt balról jobbra, az y-tengelyt pedig általában felülről lefelé értelmezik, az origó pedig a feldolgozandó kép bal-felső sarkába esik. A kép általában téglalap alakú.) A képfüggvény értékkészlete folytonos, pozitív-definit és korlátos. Mivel a leképezés a gyakorlatban nem egy adott frekvencián, hanem egy bizonyos sávban történik, a képfüggvény a leképezés spektrális tulajdonságaitól is függ. Ez adott tartományban, meghatározott alakú súlyfüggvénnyel (érzékenységgel) történő integrálásnak felel meg. Ha a leképezés egyidejűleg több – pl. más-más sávtartományból származó – kimenőjelet szolgáltat, akkor a képfüggvény elemei vektorok lesznek. Ez jellemző pl. a színes kép bevitelére.

A következő lépés a mintavételezés, majd pedig a képfüggvény diszkrét képpont-értékeinek meghatározása (kvantálás). A mintavételezés – mint művelet – lényegében integrálás: minden képpont-érték egy kis terület fényességértékeit integrálja (összegzi) úgy, hogy a súlyfüggvény a mintavételezés helyétől távolodva rohamosan csökken. A kvantálás a tetszőleges értéket felvevő képpont-értékekhez a megengedett (diszkrét) képpont-értékek valamelyikét rendeli. Az eredmény a digitális kép, melynek adatai a képpontok (ún. pixelek). A képpont-érték, ill. képpont-értékvektor az adott pontbeli világosságot, illetve színt kódolja. Soriránybanban n, oszlopirányban pedig m darab képpont alkotja a képet. A digitális kép tehát számok, illetve vektorok rendezett halmaza, s így számítógépes feldolgozásra közvetlenül alkalmas. Képszerű használatához – pl. megjelenítéséhez, képi másolat készítéséhez – újra vissza kell állítani az analóg képet, ami megfelelő eszközök és algoritmusok használatát teszi szükségessé.

13.4. Képjavítás

A digitális kép érdemi feldolgozása előtt szükség lehet bizonyos előkészítő műveletekre. A képjavítási módszerek a megvalósítandó cél szerint két csoportba sorolhatók:

- A *képhelyreállítás* célja az, hogy a különböző okok miatt torzult digitális képből előállítsuk azt a képet, amelyet a zavaró, torzító hatások nélkül kaptunk volna.

- A *képminőség javítás* célja, hogy a digitális képet a további kiértékelés vagy feldolgozás szempontjából előnyösebb formába alakítsuk. (Megjegyzendő, hogy ezen eljárásokat leggyakrabban a látvány javítása érdekéhen alkalmazzák, ami kívül esik ugyan a mesterséges látás célkitűzésein, viszont elterjedtsége miatt említést érdemel.)

Míg tehát az előbbi esetben a cél az, hogy az eredetihez legjobban hasonlító képet állítsuk elő, addig a második eljárás – akár ezzel ellentétesen – a további feldolgozás szempontjából fontos jellegzetességeket emeli ki. A továbbiakban vizsgáljuk meg egy kicsit részletesebben a képhelyreállítás lehetőségeit.

13.4.1. Képhelyreállítás

A képhelyreállításhoz ismernünk kell a zavaró hatásokat leíró képleteket, de legalábbis közelítő leírással, modellel kell rendelkeznünk. (A modell alapját például az jelentheti, hogy ismerjük a torzítás fizikai hátterét.) A teendő egyszerűen megfogalmazható: meg kell határozni azt az inverz transzformációt, amely a zavaró hatásokat közömbösíti, és ezt az inverz transzformációt kell alkalmazni a javítandó képre. A gyakorlatban a helyzet sokkal bonyolultabb: a torzítás fizikai hátterét általában csak közelítő pontossággal ismerjük, a torzítás pedig olyan információveszteséggel jár, mely a tökéletes helyreállítást lehetetlenné teszi. A képalkotás zavaró tényezői lehetnek például az optikai rendszer hibái, az érzékelők nem lineáris volta, az atmoszférikus hatások és az objektumok elmozdulása. E sokrétűségből következően reménytelen az általános modell megalkotása. A képhelyreállítási eljárások alkalmazásának másik nehézsége, hogy a felállított modell, illetve az ezen alapuló inverz transzformáció rendszerint még egyszerűsített modell esetén is bonyolult, számításigényes összefüggéseket eredményez.

13.5. Geometriai korrekció

A képhelyreállítási eljárások közül az egyik legalapvetőbb a *geometriai korrekció*. Általános esetben egy (vagy több, geometriai szempontból összetartozó) bemenő kép áttranszformálását jelenti kimenő kép(ek)re úgy, hogy a képi információ ne sérüljön a megengedettnél nagyobb mértékben, miközben a kívánt geometriai összefüggés legalább a megadott pontossággal teljesüljön. A geometriai korrekció célja szerint lehet:

- *Képjavítás*, ekkor geometriai hibákban jelentkező torzítás megszüntetése a feladat, például a képfelvevő rendszer optikai hibái, a perspektív torzítás vagy a felvétel közben történt elmozdulás miatt. (Általában elmondható, hogy a geometriai korrekcióra, mint képjavításra a korszerű képalkotó eszközök fejlettsége miatt csak a különösen nagy pontosságot igénylő felhasználási területeken van szükség.)

- *Képkorrekció*, ekkor például a képek illesztése, megadott vetületi rendszerbe való transzformációja, illetve a helyes méretarányok megvalósítása a feladat. A geometriai korrekciók az esetek jelentős részében globálisan (vagyis a teljes képre vonatkozóan) nem lineárisak. Emiatt, valamint a jelentős adatmennyiség miatt is jelentős a számításigényük. A megvalósításnak a pontossági követelmények teljesítése mellett az elfogadható végrehajtási sebesség is alapfeltétele. Szerencsére bizonyos összefüggések kihasználásával igen hatékony algoritmusokat lehet kidolgozni (pl. invertálható korrekciós összefüggések alkalmazása esetén a szomszédos pontok szomszédosak maradnak; lineáris korrekció esetén az egyenesek képe egyenes marad stb.).

13.6. Alkalmazási területek

1. **Arcfelismerés**
   Agyunk az arcokat, valamint az arckifejezéseket alakokról és mozgásokról szerzett

információinak az összekapcsolásával ismeri fel. Egyre népszerűbb kutatási területnek számít a komputeres arcfelismerés, 4 főbb aspektusra fókuszálnak:

(a) arcmodellezésre,

(b) arcfelderítésre és lokalizációra,

(c) személyek arckép alapján történő azonosítására,

(d) arckifejezés-felismerésre.

Arc- és hangulatfelismerő számítógépes rendszerek létrehozása a kísérletek célja. Az arcfeldolgozás holisztikusan, és nem lokálisan történik. A személyenként különböző (boldog, haragos, üvöltő, semleges kifejezésű, változó fényviszonyok stb. mellett felvett) képek alapján létrehoztak egy arc-adatbázist (lásd 13.1. ábra).

13.1. ábra. Példa arc-adatbázisra

A modell az emberi képfeldolgozás egyszerűsített változatát írja le. Egy beépített modul az arcizommozgást méri különböző érzések kifejezésekor, valamint azt, hogy például örömnél/bánatnál mennyivel gyorsabb/lassabb a mozgás. Ezt a számítási tevékenységet valószínűleg a mi agyunk is elvégzi, azt eredményezve, hogy akárcsak a számítógép, némi késéssel azonosít. A kevesebb mozgásból összeálló kifejezések azonosítása gépnek, embernek egyaránt könnyebben ment. Minél több volt a mozgás, annál nehezebben hajtották végre a feladatot.

Az arcokat videokamera input alapján felismerő komputer építését tervezik. Akkor se jön zavarba, ha több fotó esetén az illető más és más arckifejezéssel néz a nagyvilágba, különböző szögekből, eltérő megvilágításban kapták lencsevégre, vagy napszemüveget hord esetleg. Az ideális arcfelismerő alkalmazása különböző (tudományos, oktatási, hétköznapi stb.) területeken várható, az intelligens megfigyelő rendszerektől kezdve, eltűnt gyerekek, vagy körözött bűnözők azonosításáig.

„El tudják képzelni, amikor a komputer azt mondja: zaklatottnak tűnsz, miben segíthetek?”

Lásd Agent Portal (http://www.agent.ai).

2. **Orr-egér**

A nose (orr) és mouse (egér) szavak összevonásával kreált nouse (hozzávetőleg: orr-egér) a komputerek kéz nélküli használatát célzó eszközét az orr és a szemhéj kontrollálja. A mintafelismerésen és a gépi látáson alapuló fejlesztés webkamerákkal követi a felhasználó arcmozgását, az orr pedig hajszálpontosan „rábök” a képernyő bármelyik pixelére. Egér, joystick lesz belőle.

Lásd Agent Portal (http://www.agent.ai).

3. **Gesztus-egér**

Olyan szoftvert fejlesztettek ki, mely egy egyszerű webkamera segítségével azonosít bizonyos kézmozdulatokat. A programot akár az egér helyett is használhatjuk, ablakokat helyezhetünk át vele, nagyíthatjuk, kicsinyíthetjük őket. Amikor a hüvelyk és a mutatóujjunk összeér a mintázatfelismerő rendszer azonosítja, hogy egy nagyjából kör alakú zárt alakzat észlelhető a képen (lásd 13.2. ábra). Attól függően, hogy a

13.2. ábra. Gesztus-egér működése

billentyűzet fölött hol helyezkedik el a két összeérintett ujjunk a képernyő megfelelő pontja fölött kattintásnak felelteti meg gesztusunkat. A rendszer előnye, hogy egyszerre két kézzel is navigálhatunk, ami bizonyos műveleteket, például a képek vagy térképek nagyítását, kicsinyítését, vagy éppen forgatását lényegesen megkönnyíti. A gesztus-egér használatához ráadásul nem szükséges gépelés közben a billentyűzetről teljesen levenni a kezünket.

14. fejezet

Beszédfelismerés

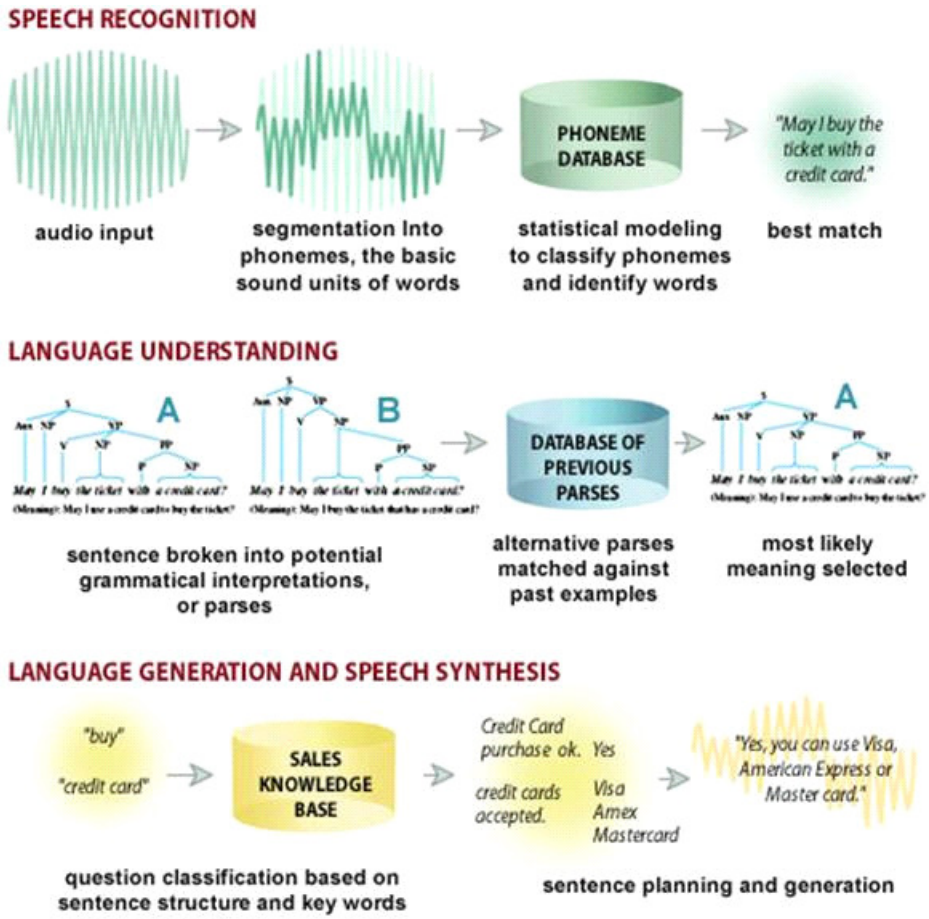
14.1. Alapok

A beszéd az emberek közötti legtermészetesebb információátviteli forma. Az ember és a gép kapcsolatában is ez lehetne talán a legcélravezetőbb, ha a számítógépekhez jó minőségű beszédperifériák állnának rendelkezésre. Manapság meglehetősen nagy az igény arra, hogy a számítógépekkel hagyományos úton, beszéd alapján kommunikáljunk. Ahhoz azonban, hogy a számítógép fel tudja dolgozni az adott nyelvet, mesterséges intelligenciát kell alkalmaznunk. Ezt a folyamatot nevezzük a természetes nyelvek feldolgozásának, idegen szóval natural language processingnek (NLP).

Ez egy rendkívül széles kutatási terület, hiszen hogy egy nyelvet és annak megértését lemodellezhessük, a nyilvánvaló programozási tudás mellett nyelvészeti, pszichológiai, matematikai ismeretekre is szükség van. A témához kapcsolódóan számos szoftver, alkalmazás született, melyek közül kiemelkedő az utóbbi években készült beszélgetőprogramok száma. Noha múltjuk egészen az 1960-as évekre nyúlik vissza, az Internet és a társalgóprogramok (MSN Messenger, IRC, ICQ stb.) fejlődésével az utóbbi évtizedben nőtt meg csak igazán az érdeklődés irántuk.

A továbbiakban vizsgáljuk meg, melyek azok a szintek, amelyeken a beszédet értelmezni lehet, s hogyan segíthetnek ezek a felismerésben?

- Fonetika: milyen hang lehet az?
- Fonológia: hogyan módosíthatták a hangot a szomszédai, állhat-e itt ilyen hang?
- Lexika, morfológia: van-e ilyen szó, szóalak?
- Szintaktika: helyes-e nyelvtanilag ez a szerkezet?
- Szemantika: van-e értelme?
- Pragmatika: vajon ebben a szituációban, szövegkörnyezetben miért ezt mondta?



14.1. ábra. A beszédfelismerés folyamata

14.2. Egy kis történelem

Az automatikus beszédfelismerés (Automatic Speech Recognition, ASR) célkitűzése olyan szoftver/hardver készítése, amely a beszédjelet írott alakra konvertálja. Az „írógép, amelynek diktálni lehet”, már a beszéd elektromos jellé alakításának, illetve rögzítésének feltalálása óta szerepel a kutatók, feltalálók vágyálmai között, amit mi sem bizonyít ékesebben, mint hogy 1930-ban Nemes Tihamér szabadalmi leírást nyújtott be egy berendezésre, amely optoelektronikai úton leírja a beszédet.

1946-ban merült fel először a számítógépes fordítás ötlete, ám csaknem tizenöt évnyi kutatás és kísérletezés kellett ahhoz, hogy belássák, a gép csak akkor tud jó fordítást készíteni, ha „érti” is a nyelvet, azaz szükség van szemantikai, sőt pragmatikai ismeretekre is. A következő évek kutatásaiban a nyelvészek vették át a vezető szerepet, s a kutatás súlypontja az elméleti kérdésekre helyeződött át. Megszületett egy új tudományos diszciplína is, a *számítógépes nyelvészet*.

A század első felében a beszéd vizsgálatát legfőképpen a távközlés motiválta; a telefontársaságoknak ugyanis nagyon komoly érdekük fűződött ahhoz, hogy minél több beszélgetést tudjanak továbbítani az adott képességű átviteli csatornákon. A minél hatékonyabb tömörítés

(szakszóval: kódolás) érdekében kezdték vizsgálni a beszédhangok szerkezetét, illetve a hallás működését. A beszédfelismerésben használt feldolgozási eljárások legtöbbje a beszédkódolásból származó módszereken alapszik (14.1. ábra), és maga az automatikus beszédfelismerés, mint kutatási ág jelenleg is inkább a villamosmérnöki tudományokhoz, semmint a számítástudományhoz tartozik.

Az ötvenes, hatvanas években a digitális technológia, a számítógépek elterjedése a területen zajló kutatásnak újabb lökést adott. A használt módszerek valamilyen kódolási átalakítást végeznek a jelen, majd egyszerű szabályokra vagy letárolt mintákkal való összehasonlításra (alakfelismerésre) alapozva hoznak döntést. Többnyire rövid beszédszeleteket próbálnak *fonéma*ként besorolni, esetleg hosszabb egységeket vesznek (szótagokat, szavakat), de az utóbbi esetben az időbeli változatosságot (rövidülés, nyúlás) még nem képesek kezelni. Az utóbbira két megoldási javaslat született: az egyik szerint a beszédet fonémákra kell szegmentálni, majd pedig a szegmenseket kell felismerni; a másik szerint nagyobb egységeket (pl. szavakat) kell venni és az időtengely menti lehetséges torzulásokat ún. dinamikus idővetemítéssel kell kezelni.

A hetvenes éveket az utóbbi megoldás letisztulása és elterjedése jellemzi. Előfeldolgozási módszerként megjelenik a *lineáris predikció*. A szavak beszélőfüggő változatosságának leküzdése érdekében többféle *klaszterező algoritmus* születik az egyetlen szóhoz tartozó „beszélőfüggetlen” minta kialakítására. Rendkívül fontos újdonságként megpróbálják a felismerés során felhasználni a magasabb szintű (lexikális, szintaktikai, szemantikai stb.) információkat. Az ekkoriban fejlesztett, ide sorolandó rendszerek ismeretalapú megközelítéssel igyekeznek az említett szinteket integrálni, struktúrájuk pedig a szakértői rendszerekre ekkortájt jellemző munkatábla, esetleg bottom-up vagy top-down architektúra. További jellemzője volt e projekteknek, hogy beszédfelismerés helyett beszédmegértésre törekedtek. Célkitűzésük az volt, hogy a rendszer helyesen reagáljon az elhangzott utasításra.

A nyolcvanas években az ismeretalapú rendszerek iránti érdeklődés megcsappant. Folyamatos beszéd felismerésére a dinamikus idővetemítési módszert ún. *kapcsolt szavas felismerés*sé egészítették ki az eljárás hívei, a háttérben azonban már készülődött, majd a 80-as évek derekán történt elterjedésével minden egyéb megközelítést háttérbe szorított a *rejtett Markov-modell* (Hidden Markov Model, HMM) alapú felismerés. Ennek alapja, hogy minden felismerendő egységhez (pl. szóhoz) tartozik egy valószínűségi modell, amely egy adott megfigyelést (felismerendő szót) valamilyen valószínűséggel generál. A felismerés kimeneteként a legnagyobb valószínűséget adó modellt választjuk. Ebből látható, hogy itt is összehasonlításon alapuló alakfelismerésről van szó, akárcsak az idővetemítésnél, a HMM azonban sokkal jobban tudja kezelni az egyes szavak esetleges kiejtési variánsait; ehhez az egyes szavakhoz rendelt modelleket be kell tanítani a szó lehetőleg minél többszöri (és – ha van ilyen – többféle) kiejtését tartalmazó mintákkal. E mintákból azután a modell statisztikai úton beállítja a paramétereit.

A kilencvenes években a HMM-alapú rendszerek dominálnak. Az évtized első felének talán leglényegesebb eredménye, hogy óriási adatbázisok készültek/készülnek (a világnyelvekre), amelyek segítségével a felismerők rejtett Markov-modelljei egyre több tanítandó paramétert tartalmazhatnak, illetve egyre jobban betaníthatók. Az egységes adatbázisok lehetővé teszik továbbá a különböző fejlesztőcsoportok felismerőinek megbízható összehasonlítását.

Az utóbbi évek legfontosabb története pedig az, hogy a multimédia elterjedése, illetőleg a processzorok, háttértárak, memóriák kapacitásának növekedése révén immár a személyi számítógépek is képesek beszédfelismerésen alapuló alkalmazások futtatására. Ez a terület látványos fejlődését hozta, gombamód szaporodnak a beszédtechnológiát kínáló cégek, a hangsúly pedig egyre inkább a kutatásról áttevődik az alkalmazásokra. [17], [11]

14.3. A beszédfelismerés alapproblémája

Az automatikus beszédfelismerés alapproblémáját rendkívül egyszerű megfogalmazni: készítsünk olyan berendezést (programot), amely leírja a bemenetét képező, hanginformációként kódolt szöveget. Első hallásra azt gondolhatná az ember, hogy ez az átalakítás mechanikusan elvégezhető, de ez sajnos nem igaz. Képzeljük el, amint valaki egy számunkra teljesen ismeretlen nyelven beszél, a mi feladatunk pedig egyszerűen csak annyi, hogy leírjuk, amit mond. Már a hangok 80-85 százalékának eltalálása is rendkívül jó eredménynek számítana. Ebből látható, hogy a beszéd felismerése elválaszthatatlan a nyelvi feldolgozástól, sőt talán magától az intelligens gondolkodástól is. Amikor egy szó elhangzik, az ember automatikusan összeveti a vélt megfejtést a lehetséges megoldásokkal: milyen hasonló hangzású szavak vannak egyáltalán, és melyik a legvalószínűbb ebben a szövegkörnyezetben, sőt: ebben a társalgási szituációban? Úgy tűnik, hogy az akusztikai-fonetikai, a szintaktikus és a szemantikus feldolgozás az emberi elmében oszthatatlan egységet alkot, és ez teszi a beszédfelismerést az egyik legnehezebb problémává a mesterséges intelligencia tárgykörében.

Az alábbi tudományok (a teljesség igénye nélkül) mind érintettek az automatikus beszédfelismerés kifejlesztésében:

- villamosmérnöki tudományok,
- számítástudomány,
- akusztika, pszichoakusztika,
- neurofiziológia,
- kognitív pszichológia,
- fonetika, fonológia,
- nyelvészet.

Az automatikus beszédfelismeréssel foglalkozó szakembernek ezek mindegyikéhez kellene valamennyit érteni, de legalábbis ezen tudományágak között megfelelő információ áramlásnak kellene lennie, ami jelenleg nem igazán mondható el.
A fejlesztők számos kihívással találkozhatnak munkájuk során:

- **Akusztikai változatosság**: a beszédjelbe bekerülhet a környezetből beszűrődő zaj, vagy a jel torzulhat a mikrofon vagy az átviteli csatorna paramétereitől függően.
- **Beszélők közti változatosság**: különböző beszélők hangmagassága, szájüregmérete, beszédsebessége, dialektusa stb. meglehetősen különbözhet.

- **Adott beszélő esetén fennálló változatosság**: még ha rögzítjük is a beszélőt, akkor is meglehetős eltéréseket mutat a beszéd, hisz a beszélő fizikai és lelki állapota is bele kódolódik a beszéd sebességébe, a hang minőségébe, a hanglejtésbe.

- **Fonetikai változatosság**: a beszédkeltés során nem rögzített alakú fonémákat fűzünk egymás után, hanem folytonosan változtatjuk hangkeltő szerveink alakját, amiből következően a szomszédos hangoktól függően a beszédhang megváltozhat.

14.4. Természetes nyelvű szövegek számítógépes feldolgozása

Az emberek és a gépek nyelve illetve nyelvfeldolgozása közti szakadék áthidalására a nyelvtechnológusok különböző módszereket dolgoztak ki a szöveg különböző szintjein. Ebben a fejezetben megnézzük, hogy milyen fázisokon halad át a feldolgozás folyamata, mígnem a számítógép számára is érthetővé válik egy természetes nyelven megfogalmazott szöveg.

14.4.1. Szegmentálás

Az első művelet a szöveg szintjén megy végbe. Descartes egyik alapelve az volt, hogy az összetettebb problémákat kisebb, kezelhető részekre bontva könnyebb megoldani. Nincs ez másként egy szöveg feldolgozása esetén sem. Első lépésként meg kell tudnunk különböztetni az egymástól különálló elemeket – egy szöveg esetén a mondatokat, egy mondat esetén a szavakat. Ezt nevezzük *szegmentálás*nak.

Ez könnyen megvalósítható a szóköz (vagy egyéb elválasztó jelek) kódjának ismerete mellett. Ügyelnünk kell mindeközben arra, hogy bizonyos központozó karakterek (például a vessző) szintén összetartozó egységek határát jelölhetik, amiknek tagjait a maguk szintjén egyben kell kezelnünk, különben könnyen téves következtetéseket vonhatunk le szűklátókörűségünk miatt. Azaz fel kell tudnunk ismerni, hol húzódnak az egyes egységek határai, és hogy az egyes szinteken mit tekinthetünk egységnek.

14.4.2. Morfológiai elemzés

A szöveg megfelelő tagolása után a szavak szintjén folytatjuk elemzésünket, mert ahhoz, hogy nagyobb egységeket (mondatokat) tudjunk értelmezni, szükségünk van az elemeinek világos és egyértelmű megkülönböztetésére. Egy nyelv önálló – tovább nem bontható – egységeit, alapelemeit nevezzük *morfémá*knak. Egy ragozó nyelvben ilyennek minősülnek a toldalékok és a szótöves formában (ún. alapalakban) álló szavak. Egy morfémának több alakja is lehet, ezeket *allomorf*oknak hívjuk. A magyar nyelvben például a ló és lovugyanannak a szónak a különböző allomorfjai. A toldalékok közt a -hoz esetragnak három allomorfja is van: -hoz, -hez, -höz. Hogy ezek közül mikor melyiket kell használni, azt a megfelelő nyelvi szabályok döntik el.

A szavak azonban ritkán fordulnak elő ekképpen, így szükség van a toldalékok leválasztására. Ehhez általában egy olyan szótárat használunk, ahol fel vannak sorolva az adott nyelv szóalakjai, mellettük a lehetséges felbontási szabályok és információk. Azonban az olyan nyelvekben, ahol a toldalékolás során rengeteg kivétel fordulhat elő, megnövekedhet a szóalakok száma, és megfelelő számítógépes architektúra hiányában nem ábrázolható mind a gép számára. Az agglutináló nyelvek, mint a magyar, lengyel, a finn, az észt, a japán stb. ilyen problémákat vetnek fel, mert toldalékokat nem külön mintában (mint például az angol kifejezések többségénél), hanem a szavakhoz hozzáragasztva használják. A morfológiai elemzésből egyértelműen megállapíthatjuk a legtöbb kifejezés szófaját és toldalékait, ám ezzel még nem biztos, hogy meg tudjuk ragadni a jelentését is. Ha a vizsgálat a szövegkörnyezettől függetlenül történik, könnyen juthatunk téves következtetésre pusztán a toldalékokból kiindulva. A *homonímiá*k, vagyis azonos alakú, de eltérő jelentésű szavak félrevezethetik a gépi intelligenciát.

14.4.3. A szövegkörnyezet figyelembevétele

Valamely természetes nyelv nyelvészeti elemzésének alapvető célja az, hogy a nyelvtanilag helyes sorozatokat különválasszuk a nyelvtanilag helytelen sorozatoktól. A morfológiai elemzéssel elérhető *formai (szintaktikai) helyesség* nem garantálja a *tartalmi (szemantikai) helyesség*et. Hiszen egy szó előfordulhat a megfelelő hangalakkal és írásmóddal nem megfelelő közegben. Bár nyelvtanilag nem vétenénk semmilyen hibát, nyilvánvalóan összezavarnánk beszélgetőtársunkat, ha azt mondanánk neki, hogy „leviszem a tejet sétálni”. A morfológiai elemzés során felmerült homonímia-probléma kezelése lehetővé válik a mondatok szintjén, a kontextus megragadásával.

Az említett két nyelvi szinten kívül létezik még kettő: *pragmatikai* és *intencionális szint*. Az előbbi a kifejezés gyakorlati, valódi értelmét jelzi, vagyis többértelmű jelentés esetén a megfelelő jelentésre utal, figyelembe véve a beszédhelyzetet (szituációt) és a szövegkörnyezetet (kontextust). Az intencionális szint a szándékolt jelentést foglalja magában, vagyis azt, amire a beszélő utalni szeretne.

Magához a nyelvtanhoz el lehet jutni intuíció, sejtés, mindenféle esetleges módszertani javaslatok, korábbi tapasztalatra való hagyatkozás stb. útján, a nyelvtan azonban összetett rendszer, részei közt sok és sokféle kapcsolat áll fenn. A homonímiák mellett a szinonimák is gondot okozhatnak az értelmezésben. Az utóbbiak alatt különböző alakú, ám hasonló jelentésű szavakat, szókapcsolatokat értünk. Egy gép számára az eb és a kutya szó kódjai különbözőek, így nem ismeri fel a kettő kapcsolatát.

14.5. A felismerők képességeinek csoportosítási szempontjai

- **A beszédjel minősége**: ezt befolyásolja a zajszint, a zaj típusa (stacionárius vagy gyorsan változó), a mikrofon ill. az átviteli csatorna minősége. 3 kategóriára lehet egyszerűsíteni: stúdióminőség, irodai minőség, telefonminőség.

- **A beszédmódja**: lehet izolált szavas (egyetlen szót kell csak felismerni, ill. a szavak között rövid szünetet kell tartani), ill. folyamatos beszéd.

- **Beszélőfüggőség**: egyetlen beszélő hangjára kell csak figyelni, beszélő függetlenség esetén bárkiére. A kettő között képez átmenetet az adaptív felismerő, amely fokozatosan megtanulja a beszélő hangját, azaz beszélő függetlenségből beszélő függővé alakul.

- **A szótár mérete**: hány szó fordulhat elő a beszédben. Kis szótáras felismerő 10-100, közepes 1-2 ezer, nagy szótáras pedig több tízezer szót képes felismerni.

- **A nyelvi kötöttség foka**: egy speciális szituációban (vonatjegy rendelése) a lehetséges mondatok mind szintaktikailag, mind szemantikailag rendkívül kötöttek, sőt a párbeszéd is modellezhető.

  Az emberekhez hasonlóan a gépi felismerőnek is szüksége van tanulásra, mind a nyelvi, mind az akusztikus információt valamilyen formában előre be kell vinni a rendszerbe. Ha egy nyelv szókészletének egy részével és hangjainak paramétereivel (spektrum, időbeli lefolyás) és kiejtési szabályaival betanítunk egy gépi felismerőt, akkor lehet esély arra, hogy önálló szavakat vagy hosszabb kifejezéseket gépi úton felismertessünk.

  Kötetlen, folyamatos beszéd felismeréséhez vagy a nagy háttérzajban történő felismeréshez szükséges a nyelvi és tartalmi elemzés is, mint ahogy az ember is csak azt ismeri fel biztonságosan, amit megért.

14.6. A gépi beszédfelismerés folyamata

- **Akusztikus előfeldolgozás**: melynek során a beszéd információtartalmát jellemző paramétereket határozzák meg. Ennek során eltávolítják a beszélőre, annak hangulatára, és a környezetre vonatkozó adatokat. A beszédfelismerés célja a beszéd információtartalmának kinyerése.

- **Mintaillesztés**: Az előfeldolgozás után kapott paramétereket mintaillesztéssel vetik össze a referenciamintákkal vagy modellekkel, amelyeket a betanítás során készítenek és tárolnak el. A felismerés alapegységei lehetnek az egyes beszédhangok és ezek kombinációi (kettőshangok, hármas hangok, félszótagok, szótagok, szavak vagy akár hosszabb kifejezések). Az angolban és számos más nyelvben a szavak a legalkalmasabb alapegységek. A magyar nyelvben a ragozás, toldalékolás miatt minden szónak több száz vagy akár ezer alakja is lehet, ezért a szavaknál kisebb egységeket szokás választani. A beszédhangok nemcsak attól függenek milyen hang van előttük/utánuk, hanem az akusztikai környezettől, a beszélő személyétől, nemétől, szociális és regionális hovatartozásától stb.

- **Nyelvi elemzés**: az akusztikai illesztésnél legjobbnak bizonyult elemek sorozatából a legvalószínűbb szavakat vagy hosszabb szövegeket választhatjuk ki a szótárt és a nyelvtani ismereteket tároló tudásbázisból. A beszédhangokon, mint elemi egységeken alapuló, ún. nyílt szótáras felismerés lehetővé teszi, hogy új szavak egyszerűen felvehetők legyenek a szótárba. A modelleket nagymennyiségű, beszédhangokra szegmentált mintával kell betanítani.

14.7. Alkalmazási területek

A beszédfelismerő alkalmazások egy részénél csak kényelmi vagy anyagi szempontok játszanak szerepet, máshol azonban a kéz és a szem felszabadítása az alapvető szempont. Ilyen alkalmazások például:

- telefonálás vezetés közben,
- diktálás sötétben (pl. röntgenezésnél),
- leltározás terepen,
- fogyatékosok számára használható rendszerek stb.

Néhány konkrét példa:

Napjainkban elszaporodtak a chat-botok (chatting robot – csevegő robot), amelyek különböző csatornákon található csevegőszobákban teljesítenek moderátori feladatot (automatikusan jogokat adhatnak bizonyos személyeknek stb.) vagy épp szórakoztatási szerepet játszanak (vicceket mesélnek, híreket mondanak stb.).

A bűnüldözésben is használtak/használnak beszélgetőprogramokat, például pedofilok leleplezésére és kézrekerítésére. A netdadáknak (netnannies) nevezett botok fiataloknak álcázva magukat belépnek a tizenévesek által látogatott csevegőszobákba, és amikor egy pedofil-gyanús egyén megjelenik, magánbeszélgetésbe kezdenek vele, és kiderítenek róla annyi információt, amennyi csak lehetséges, majd jelentést küldenek a hatóságoknak.

A genfi Agence Virtuelle RumorBotja is hasonló módszereket alkalmaz. A feladata a rémhírek kiszűrése. Napi több millió honlapot, emailt fésül át valós időben.

Irodalomjegyzék

[1] Agent Portal http://www.agent.ai

[2] Álmos, A., Győri, S., Horváth, G., Várkonyiné Kóczy, A., Genetikus algoritmusok, Typotex, 2002.

[3] Ames, A.L., Nadeau, D.R., Moreland, J.L., VRML 2.0 alapkönyv, Panem Kiadó, 2000.

[4] Borgulya, I., Neurális hálók és fuzzy-rendszerek, Dialóg Campus, 1998.

[5] Borgulya, I., Evolúciós algoritmusok, Dialóg Campus, 2004.

[6] Csendes, B., Ablak a virtuális világra, CSKBB BT., 1998.

[7] Derakhshani, D., MAYA - 3D Modellezés és animáció, Perfact Kiadó, 2009.

[8] Donner, Cs. A robottechnika alapjai, Gura Nyomda Bt., 2008.

[9] Fekete István, Gregorics Tibor, Nagy Sára: Bevezetés a mesterséges intelligenciába. LSI, Budapest 1990.

[10] Ferber, J.: Multi-Agent Systems - An Introduction to Distributed Artificial Intelligence. Addison-Wesley, 1999.

[11] Futó, I., Mesterséges intelligencia, Aula Kiadó, 1999.

[12] Giarratano P. C.: Expert Systems: Principles and programming. PWS Publishing Co. 1999.

[13] Jackson, P.: Introduction to Expert Systems. Addison-Wesley Pub. Comp, 1986.

[14] Jamsa, K., Schmauder, P., Yee, N., VRML programok könyvtára I., Kossuth Kiadó, 1998.

[15] Kóczy László T., Tikk Domonkos: Fuzzy rendszerek. http://www.tankonyvtar.hu/informatika/fuzzy-rendszerek-fuzzy-080904 2001.

[16] MATLAB - The Language Of Technical Computing http://www.mathworks.com/products/matlab/

[17] Russell, S.J., Norvig, P., Mesterséges intelligencia modern megközelítésben, Panem-Prentice Hall, 2000.

[18] Sántáné-Tóth Edit: Tudásalapú technológia, szakértő rendszerek. Dunaújvárosi Főiskola Kiadói Hivatala, 2000.

[19] Starkné Werner Á.: Mesterséges intelligencia. Veszprémi Egyetemi Kiadó, 2004.

