Pécsi Tudományegyetem
Természettudományi Kar

A MESTERSÉGES INTELLIGENCIA ALAPJAI

EGYETEMI JEGYZET

GIMESI LÁSZLÓ

PTE SMART UNIVERSITY PROGRAM

2026

Pécsi Tudományegyetem, Természettudományi Kar,
Matematikai és Informatikai Intézet

A mesterséges intelligencia alapjai

Egyetemi jegyzet

Gimesi László

Készült a PTE Smart University (SU) program "MI a közoktatásban és szakmódszertani kurzusok" című „Kiemelt Projekt” keretében

2026-ban

Tartalomjegyzék

Bevezetés

A mesterséges intelligencia (MI) az elmúlt hat-nyolc évtizedben rendkívüli módon fejlődött és változott. Az MI különböző kutatási irányokat ölel fel, amelyek célja, hogy gépek olyan módon viselkedjenek, gondolkodjanak vagy tanuljanak, amely emberi értelemben intelligensnek minősül. Célunk bemutatni az MI kialakulásának történetét és fontosabb állomásait, kezdve a filozófiai előzményektől, a korai kísérleteken át az aktuális trendekig, és rávilágítani arra, hogy a múlt sikerei és kudarca számos tanulsággal szolgálnak a jelen és jövő számára.

E téma megértéséhez elengedhetetlen, hogy néhány alapvető fogalmat és az MI működését megismerjük.

A jegyzet azok számára készült, akik meg szeretnék érteni a mesterséges intelligencia működését, de nem rendelkeznek e témában mélyebb matematikai és informatikai ismeretekkel.

A jegyzet első részében összefoglaljuk a mesterséges intelligencia kutatásának történetét. A második rész a különböző típusú neurális hálózatokkal, azok tanításával, valamint alkalmazási területeikkel foglalkozik. A harmadik rész pedig az evolúciós algoritmusokat ismerteti.

Előzmények

Az emberiség évezredes vágya, hogy olyan berendezést alkosson, amely emberi viselkedésre, gondolkodásra képes. Már az ókorban is készítettek mechanikus – elsősorban állatokat utánzó – automatákat. Egy emberformájú automatáról Athemaiosz görög író „A bölcsek lakomája” című művében ír, ahol állítólag a gép az asztalnál evett, ivott és különböző mozdulatokat végzett. (Coll, 1966)

A középkorban az automaták (automatonok) mind nagyobb számban történő megépítése tükrözte az emberiség vágyát, hogy az élőlények, illetve az értelem gépi mását létrehozza. Az automaták virágkora a reneszánsz idejére tehető, amikor az emberek érdeklődésének középpontjába az élet dolgai kerültek, így az élőlények működése egyre nagyobb figyelmet kapott. A technika fejlettsége ekkor már lehetővé tette különböző precíz mozgásokra képes berendezések létrehozását. Ebben az időben készültek az első kakukkos órák, zenét játszó, illetve táncoló alakok, és ne feledkezzünk meg Kempelen Farkas sakkozó, illetve beszélő gépéről sem.

A mesterséges intelligencia gondolata jóval a számítógépek megjelenése előtt megszületett. A filozófiai előzmények közé tartozik René Descartes felfogása a világ mechanikus működéséről (Simonyi, 1981): vajon, ha az emberi test is mechanikus működésű, akkor a tudat is leírható algoritmusok segítségével, és így gépesíthető? Vagy megemlíthetjük Thomas Hobbes gondolatát, miszerint „a gondolkodás nem más, mint számítás” (Csapó, 1992).

A 19. században George Boole megalkotta az algebrai logikát, amely lehetővé tette a logikai állítások matematikai formalizálását. Charles Babbage megfogalmazta a programozható számítógép gondolatát, analitikus gépének terve már tartalmazta a logikai műveletek mechanikus elvégzését. Ada Lovelace (Babbage munkatársa) pedig előrevetítette, hogy a számítógépek nemcsak számokat, hanem szimbólumokat is kezelhetnek.

Az 1930-as években Alan Turing publikálta a Turing-gépet, amely elvi modellként szolgált az „algoritmus” és „gép” fogalmak számára, és később az intelligencia gépi modelljének vizsgálatában is kulcsfontosságúvá vált (Turing, On computable numbers, with an application to the Entscheidungsproblem, 1936).

A mesterséges intelligencia (MI) története 1943-ban kezdődött, amikor Warren McCulloch és Walter Pitts megalkották a mesterséges neuron (idegsejt) modelljét.

(McCulloch & Pitts, 1943) Ez a cikk először mutatta be, hogy az idegsejtek működése logikai műveletekkel formálisan leírható. Ez lett az alapja a későbbi neurális hálózatoknak.

Alan Turing, 1948-ban írt, Intelligent Machinery című jelentésében (Turing, Intelligent machinery (1948), 2004) megemlíti a gépek tapasztalati úton történő tanulásának gondolatát. Elképzelése szerint ezeket a gépeket a feladatok elvégzésére inkább képeznék, mintsem explicit módon programoznák.

Turing 1952-ben megjelent cikkében (Turing, The chemical basis of morphogenesis, 1952) felveti az evolúció számítógépes szimulációját, bemutatja, hogyan írhatnak le egyszerű matematikai modellek a természetben előforduló komplex mintázatokat.

Az 1950-es évek közepén Turing a „Computing Machinery and Intelligence" című tanulmányában (Turing, Computing Machinery and Intelligence, 1950) írt a gépi viselkedésről és a tanulás lehetőségéről, felteszi a kérdést: gondolkodhatnak-e a gépek? E munkájában ismerteti a híres Turing-tesztet. (Ha egy gép kommunikációban megkülönböztethetetlen az embertől, akkor intelligensnek tekinthető.)

Az evolúciós számítások és szimuláció legkorábbi képviselője Nils Aall Barricelli volt. Az 1954-ben a Methodos folyóiratban megjelent Esempi Numerici di processi di evoluzione című tanulmánya a mesterséges élet kutatásáról szólt. (Fogel D. B., 2006)

1956-ban a New Hampshire-i Dartmouth College-ben John McCarthy, Marvin Minsky, Nathaniel Rochester és Claude Shannon megszervezték a híres „Dartmouth Konferenciát". (McCarthy, Minsky, Newell, & Simon, 1956) Ez egy workshop volt, ahol matematikusok, számítástechnikusok, villamosmérnökök, pszichológiával foglalkozó kutatók találkoztak és ötleteltek. Itt hangzott el először az „artificial intelligence" kifejezés, és itt fogalmazódott meg az a vágy, hogy néhány évtizeden belül a gépek képesek lesznek szimulálni az emberi tanulást és gondolkodást. Ezzel hivatalosan is megszületett a mesterséges intelligencia tudománya.

Ebben az időszakban fejlesztették ki az első mesterséges intelligencia (MI) programokat. Ilyen volt például a Logic Theorist (Newell & Simon, 1956), majd ennek továbbfejlesztése a General Problem Solver (Simon & Newell, 1956). Ezeket szimbolikus logika tételek bizonyítására készítették. A szerzők úgy vélték, hogy az emberi gondolkodás formális logikai szabályokkal leírható, és így számítógépen modellezhető (Crevier, 1993).

Ezt az időszakot gyakran nevezik az MI „aranykorának”, amikor optimizmus uralkodott, úgy tűnt, hogy az általános intelligencia elérhető közelségbe került.

1966-ban Joseph Weizenbaum kifejlesztette az ELIZA első természetesnyelv-feldolgozó számítógépes programot (chatbot-ot), amely szövegbevitel alapján párbeszédet imitált (Weizenbaum, 1966). Az ELIZA mintaillesztési és helyettesítési módszerrel szimulálta a beszélgetést, azt az illúziót keltve, hogy a program megérti a szöveget.

Az 1960-70-es években megjelentek az úgynevezett MI-alapú szakértői rendszerek, amelyek az emberi szakértők tudását, formalizált szabályrendszerbe építették. Ezek elsősorban az orvostudományban és a vegyiparban hoztak áttörést, például:

- DENDRA: az 1960-as években a Stanford Egyetemen fejlesztették ki, elsődleges célja az ismeretlen szerves vegyületek molekulaszerkezetének feltárása volt tömegspektrometriai adatok alapján.
- MYCIN: az 1970-es években a Stanford Egyetemen kifejlesztett orvosi szakértői rendszer volt, a bakteriális fertőzések diagnosztizálására és a megfelelő kezelés kiválasztására.

Ugyanakkor a szimbolikus megközelítés korlátai is nyilvánvalóvá váltak, az elme túl komplex ahhoz, hogy minden tudást explicit szabályokkal leírjunk. (Dreyfus, 1992)

Ebben az időszakban jelentek meg az első evolúciós algoritmusok a biológiai evolúció szabályainak utánzására. Jeles képviselői Ingo Rechenberg és Hans Paul Schwefel voltak. Rechenberg 1973-ban megjelent *Evolúciós stratégiák* című tanulmányban (Rechenberg, The Evolution Strategy. A Mathematical Model of Darwinian Evolution, 1973) felveti, hogy a biológiai evolúció szabályainak utánzása kiváló kísérleti módszert eredményezhet a mérnöki tudományokban és a műszaki berendezések tervezésében. Schwefel (Schwefel, 1981) optimalizálási feladatokra használta az evolúciós algoritmusokat. Az evolúciós stratégiák (Evolution Strategies – ES) fő jellemzője a valós értékű paraméterek kezelése és a mutáció domináns szerepe volt.

Ezzel párhuzamosan az Egyesült Államokban Lawrence J. Fogel kidolgozta az evolúciós programozást (Evolutionary Programming – EP), amely eredetileg véges állapotú automaták viselkedésének optimalizálására szolgált. (Fogel, Owens, & Walsh, 1966)

A genetikus algoritmusokat (Genetic Algorithm – GA), a michigani egyetemen Holland a diákjaival együtt fejlesztette ki. Céljuk kezdetben nem egy optimalizáló módszer megalkotása volt, hanem a szelekció és az adaptáció matematikai és számítógépes modellezése. (Holland, 1992)

Az 1973-ban publikált Lighthill-jelentés (Lighthill, 1972) bírálta az MI-kutatások gyakorlati eredményeit. Lighthill azt állította, hogy az akkori mesterségesintelligencia-kutatás nem érte el a kitűzött célokat, több területen is hiányzott a valódi gyakorlati eredmények felmutatása. Ennek hatására az Egyesült Királyság kormánya jelentősen csökkentette az MI-kutatások finanszírozását, több laboratórium és projekt támogatása megszűnt. Ez jelentősen visszavetette az MI kutatásokat.

A szimbolikus irányzat alternatívájaként már az 1950-es években felbukkant a kapcsolatizmus, amely az agy ideghálózatait próbálta modellezni. Frank Rosenblatt 1958-ban fejlesztette ki a perceptront, a biológiai neuronok leegyszerűsített modelljét (Rosenblatt, 1958), amely képes volt egyszerű minták felismerésére.

Ebben az időben a kutatók és a fejlesztők hitték, hogy a gépi intelligencia teljeskörűen megvalósítható néhány évtizeden belül. (A kutatást az amerikai DARPA is támogatta.) A korai lelkesedést azonban visszavetette Minsky és Papert (Minsky & Papert, 1969) kritikája, amely rámutatott a perceptron matematikai korlátaira.

Az 1980-as években a visszaterjesztés (backpropagation) algoritmus (Rumelhart, Hinton, & Williams, 1986) újraélesztette a neurális hálózatok kutatását. A kapcsolatizmus megközelítése – szemben a szimbolikus MI-vel – nem explicit szabályokat, hanem mintázatok tanulását hangsúlyozta. Ez közelebb állt az emberi tanulás és észlelés képességéhez, és ezzel az MI fejlődése lendületet kapott.

Az 1980-as és 1990-es évek az evolúciós algoritmusok konszolidációjának időszaka volt. Bäck és Schwefel az 1993-ban megjelent cikkükben az evolúciós algoritmusok három fő irányzatát (ES, EP, GA), a természetes evolúció modelljén alapuló valószínűségi optimalizáló algoritmusokat hasonlították össze. (Bäck & Schwefel, An Overview of Evolutionary Algorithms for Parameter Optimization, 1993). Világossá vált, hogy az ES, EP, GA és a később megjelenő genetikus programozás (Genetic Programming, - GP) közös alapelveken nyugszanak. Ezt az egységes szemléletet evolúciós számításnak (Evolutionary Computation – EC) nevezték. (Bäck, Fogel, & Michalewicz, Handbook of Evolutionary Computation, 1997) E korszak kutatói

bemutatták a populációalapú keresés lehetőségeit a determinisztikus matematikai módszerekkel szemben.

A 1990-es évektől az MI kutatás fókusza fokozatosan áttevődött a gépi tanulásra (machine learning). A hangsúly immár nem a szabályalapú tudásreprezentáción, hanem az adatokból való mintázatfelismerésen és predikción volt. A növekvő számítási kapacitás és a digitalizált adatok mennyiségének robbanása lehetővé tette az olyan algoritmusok elterjedését, mint a döntési fák, a támogatóvektor-gépek (SVM), valamint a Bayes-hálók.

A 2000-es évektől kezdve a gépi tanulás az ipari alkalmazások alapja lett: keresőmotorok, ajánlórendszerek, pénzügyi előrejelzések és önvezető járművek mind ezen elvekre épülnek. Ez a korszak az MI „második reneszánszaként" is ismert. (Domingos, 2015)

Ekkor az evolúciós algoritmusok is egyre szélesebb körben kezdtek elterjedni a gyakorlatban. Kiemelkedő jelentőségű volt a többcélú evolúciós optimalizálás (Multi-Objective Evolutionary Algorithms, - MOEA) megjelenése, különösen a Pareto-alapú megközelítések, mint az NSGA-II (Deb, Pratap, Agarwal, & Meyarivan, 2002). Az evolúciós algoritmusokat többek között a mérnöki tervezésben, a bioinformatikában, a pénzügyi optimalizálásban, valamint az ütemezési és a logisztikai feladatoknál használták.

A 2010-es évektől a mélytanulás (deep learning) forradalma következett be. A több rétegű neurális hálók – köszönhetően a grafikus processzorok (GPU) teljesítményének és az adatmennyiség növekedésének – képesek lettek összetett vizuális, nyelvi és döntési feladatok megoldására. E területet Geoffrey Hinton, Yann LeCun és Yoshua Bengio munkái alapozták meg, amiért 2018-ban megkapták az ACM (Association for Computing Machinery) Turing-díját.

A 2020-as évekre megjelentek a generatív modellek, mint a GPT (Generative Pre-trained Transformer) és a Stable Diffusion, amelyek nem csupán elemeznek, hanem új szövegeket, képeket, zenéket hoznak létre. Ez a „kreatív MI" korszaka, amely új kérdéseket vet fel a szerzői jogról, az emberi alkotásról és a tudás fogalmáról (Boden, 2016) (Floridi, 2019).

Napjainkban az evolúciós algoritmusok fejlődése egyre inkább az integráció és automatizálás irányába mozdult el. Megjelentek a hibrid algoritmusok, amelyek

evolúciós algoritmusokat kombinálnak például megerősítéses tanulással vagy mélytanuló neurális hálózatokkal. Kiemelt terület a neuroevolúció, ahol evolúciós algoritmusok segítségével optimalizálják neurális hálózatok súlyait, architektúráját, illetve paramétereit (Stanley & Miikkulainen, 2002). Ez a megközelítés fontos szerepet játszik az automatizált gépi tanulás (AutoML) és a neurális architektúra keresés (NAS) területén is.

A mesterséges intelligencia története jól mutatja, hogy az intelligencia gépi modellezése nem lineáris fejlődési folyamat, hanem egymást váltó paradigmák, kudarcok és újrafelfedezések sorozata. A szimbolikus MI formalizmusai, a kapcsolatista tanulási modellek, valamint az evolúciós számítás populációalapú szemlélete mind hozzájárultak ahhoz, hogy napjaink komplex rendszerei létrejöhessenek.

A jelenlegi MI-rendszerek – különösen a mélytanulásra és generatív modellekre épülők – nem általános értelemben intelligensek, hanem szűk feladatokra optimalizált, statisztikai mintázatfelismerő rendszerek. Ennek felismerése kulcsfontosságú az irreális várakozások és a túlzott félelmek elkerülése érdekében (Russell & Norvig, 2020).

A jövő mesterséges intelligenciája várhatóan nem egyetlen módszertan diadala lesz, hanem hibrid megközelítések eredménye, ahol a tanulás, az optimalizáció, a szimbolikus következtetés és az ember–gép együttműködés egyaránt szerepet kap. Ezzel párhuzamosan az etikai, jogi és társadalmi kérdések egyre hangsúlyosabbá válnak, és az MI fejlesztése elválaszthatatlanná válik a felelős tervezés elvétől (Luciano, és mtsai., 2018).

Az idegrendszer modellezése

Az élőlények törzsfejlődése során alapvető jelentőségűvé vált a környezethez való alkalmazkodás képessége. A helyváltoztatás, a táplálékszerzés, a veszélyek elkerülése és a szaporodás mind olyan tevékenységek, amelyek hatékony információfeldolgozást igényelnek. A bonyolultabb környezeti kihívások és a komplexebb mozgásformák szoros kölcsönhatásban állnak az idegrendszer fejlődésével. Minél összetettebbé vált az élőlények viselkedése, annál nagyobb szerephez jutott az idegrendszer (és annak központosulása), amely képes a környezetből érkező ingerek érzékelésére, feldolgozására, tárolására és a megfelelő válaszok kiválasztására.

Az idegrendszer működése

Az idegrendszer az élő szervezetek legösszetettebb szabályozó rendszere, amelynek evolúciós sikere a környezeti ingerekre adott gyors és specifikus válaszreakciókban rejlik. Működésének alapja nem csupán a morfológiai egységek (neuronok) hálózatos elrendeződése, hanem a szinaptikus plaszticitás[1] és a molekuláris jelátviteli útvonalak dinamikája[2]. Az idegrendszer feladata az interoceptív[3] és exteroceptív[4] információk kódolása, integrálása, majd effektor[5] válaszok generálása (Kandel, Koester, Mack, & Siegelbaum, 2021).

Az idegrendszer az érzékelés (szenzoros percepció) segítségével nyer információt a külső és belső környezetről. Az érzékelési folyamat első lépése az ingerfelvétel, amelyet specializált receptorok végeznek. Ezek a receptorok képesek fizikai (például fény, hang, mechanikai nyomás) vagy kémiai ingerek (szag- és ízanyagok) felismerésére és átalakítására idegi (elektromos) jellé, ez a folyamat a transzdukció (Kandel, Koester, Mack, & Siegelbaum, 2021).

Az érzékszervi receptorok által generált elektromos jel (ingerület) idegpályákon keresztül jut el a központi idegrendszerbe, ahol a jelfeldolgozás eredményeként

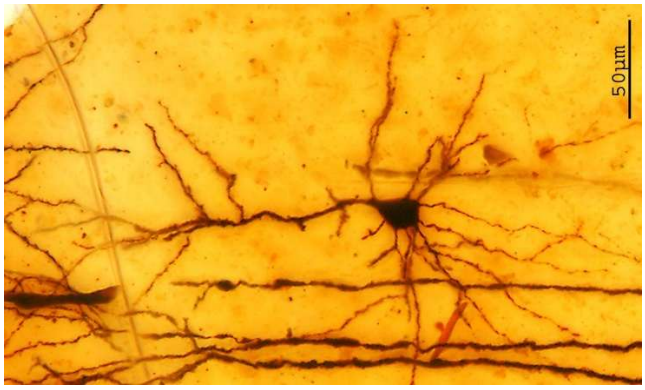
[1] Szinaptikus plaszticitás: az idegsejtek kapcsolódásainak rugalmas alakulásra való képessége – ez a tanulás idegélettani alapja.
[2] A sejtek közötti kommunikáció és a belső válaszok komplex, időben változó folyamatai, magában foglalva a jeldetektálást, az erősítését és terjedését.
[3] A test belső állapotára, szerveire és fiziológiai folyamataira vonatkozó érzékelés.
[4] A környezetből származó ingereket érzékelése.
[5] Effektor: biológiai értelemben egy végrehajtó szerv vagy sejt (pl. izom, mirigy).

kialakul az érzet és az ingerre adott válasz. A válasz szintén idegpályákon keresztül jut el az effektor szervre.

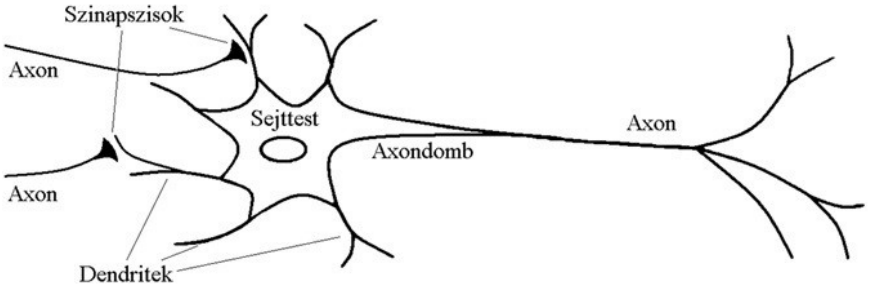
Az idegsejt (neuron)

Az idegsejtek az ingerület vezetésére specializálódott sejtek (1. ábra), sematikus képét a 2. ábra szemlélteti. Az ábrákon megfigyelhető az idegsejt három fő alkotó eleme:

1. A dendritek az idegsejtek faágszerűen elágazó, rövid nyúlványai, amelyek a többi sejttől kapják az ingerületet és továbbítják azt a sejttest felé.
2. A sejttest (szoma) felel az információ feldolgozásáért és a dendritektől jövő jelek összegzéséért.
3. Az idegsejtnek egyetlen kimeneti nyúlványa van, az axon: feladata az idegsejt által generált elektromos információ továbbítása. Az axon vége faágszerűen elágazik, ez biztosítja, hogy az impulzusok egyszerre több más idegsejthez vagy effektorhoz is eljussanak.



1. ábra. Idegsejt preparátum (Garcıa-Lopez, Garcıa-Marın, & Freire, 2006)



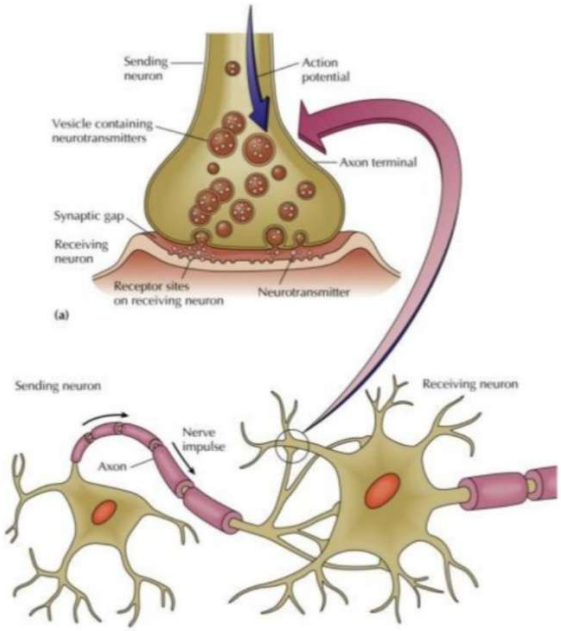
2. ábra. Idegsejt sematikus ábrája

Az idegsejt működése: a dendriteken beérkező információt a sejtest összegzi, majd az axondombon (axon hillock) keresztül az axonra továbbítja, amely a végén szétágazva (axonfa) kapcsolódhat más idegsejtekhez. Az axondomb az idegsejt sejttestének és az axonnak a találkozásánál található, szerepe az akciós potenciál kezdeményezése.

Ami azt jelenti, hogy az axonra csak akkor jut ingerület, ha az összegzett jelek erőssége (aktivációs potenciálja) egy meghatározott szintet elér.

Az idegsejtek közötti információátadást a szinapszis biztosítja, amelynek két fő típusa van:

1. Az elektromos szinapszisokban a sejtek közötti kommunikáció közvetlen elektromos kapcsolat útján történik, réskapcsolatok (gap junctionok) segítségével. Itt az ionáramlás közvetlenül megy át az egyik sejtből a másikba, ami rendkívül gyors és szinkronizált jelátvitelt tesz lehetővé. Az elektromos szinapszisok elsősorban olyan idegrendszeri hálózatokban jellemzőek, ahol a gyors összehangolt aktivitás elengedhetetlen.
2. A kémiai szinapszisok esetén az elektromos ingerület hatására neurotranszmitterek szabadulnak fel, amelyek a szinapszisrésen át jutnak el a receptorokhoz, ahol ismét elektromos ingerületté alakulnak (3. ábra). Ez a folyamat lassúbb, de rendkívül plasztikus és szabályozható, ezért a központi idegrendszer legtöbb szinapszisa ilyen típusú.



3. ábra. A szinapszis (Neuron Structure, 2014)

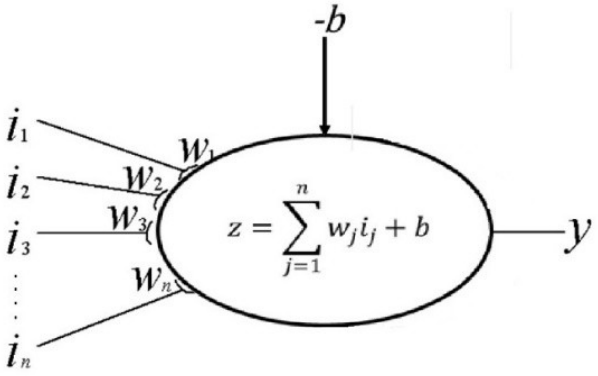
A továbbiakban az idegrendszer modellezéséhez, a kémiai szinapszist fogjuk használni, ahol az információ irányítottan, egyirányban terjed. A szinapszisokban a jel erőssége nem állandó, az csökkenhet (gátlás) vagy erősödhet. A továbbiakban ezt az erősítési tényezőt (súlyt) jelöljük *w*-vel (weight).

Az idegsejt modellje

A mesterséges neuronokat és az azokból felépülő hálózatokat az információt hordozó jel típusa, aktivációs függvénye, tanulási mechanizmusa és biológiai realizmusa alapján különböző generációkba sorolhatjuk. A generációk nem feltétlenül időrendben váltják egymást, hanem inkább egymásra épülő paradigmák, amelyek a számítási képességek és a biológiai realizmus különböző szintjeit képviselik.

Első generáció – küszöbérték modell (*McCulloch–Pitts (MCP) neuron)* (McCulloch & Pitts, 1943)

Az elsőgenerációs matematikai modell, logikai műveletek modellezésére alkalmas. Képes egyszerű osztályozásra, lineárisan szeparálható problémák megoldására. A neuron modelljét a 4. ábra mutatja.

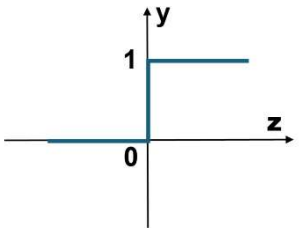


4. ábra. Küszöbérték modell

A neuron *n* darab bemenetét (bemeneti ingerületét) jelöljük i_k-val, ahol $k=1,2,\ldots,n$. A bemeneteken lévő értéket megszorozzuk az adott bemenethez tartozó súllyal (w_k), majd ezeket összegezzük. A végén hozzáadjuk a *b* (bias – eltolás) értéket.

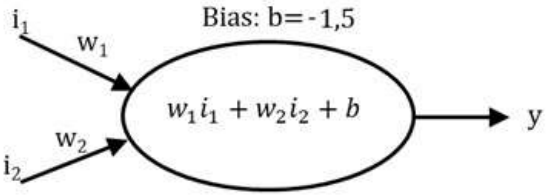
Ha a bemenetek súlyozott összege eléri az küszöbértékét (b), a neuron aktiválódik (tüzel), ekkor a kimenete: 1, különben inaktív (0) (5. ábra).

$$y = \begin{cases} 1, ha\ z > 0 \\ 0, ha\ z \leq 0 \end{cases}$$



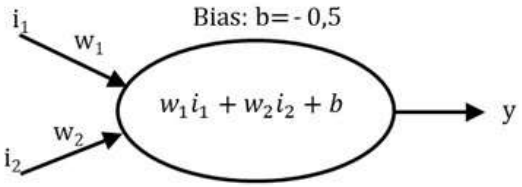
5. ábra. A neuron kimenete

A modell tanítása során a súlyszámokat úgy változtatjuk, hogy a kimenet megfeleljen az elvárásnak. Egyszerű példa a logikai ÉS (6. ábra), illetve a VAGY (7. ábra) művelet modellezése. (A példákban a súlyszámok értéke 1.)



Igazságtábla		
i_1	i_2	y
1	1	1
1	0	0
0	1	0
0	0	0

6. ábra. ÉS (AND) művelet



Igazságtábla		
i_1	i_2	y
1	1	1
1	0	1
0	1	1
0	0	0

7. ábra. VAGY (OR) művelet

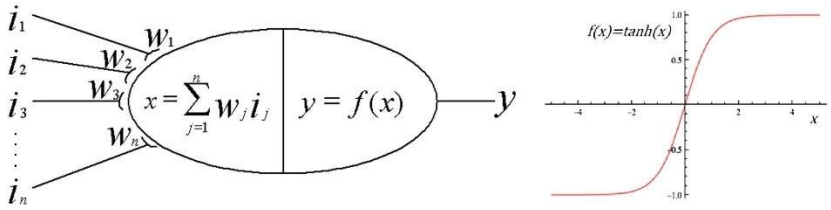
Az 1. generációs modell jellemzői:

- bináris bemenetek és kimenetek,
- egyszerű, de nem képes nemlineáris problémák megoldására (pl. XOR),
- aktivációs függvény: egységugrás,
- a modellekből felépített hálózat fő problémája a betanítási szabályok hiánya volt, amelyek lehetővé tették volna a bonyolultabb hálózatok kialakítását (Markowska-Kaczmar & Koldowski, 2015).

Második generációs neurális hálózatok

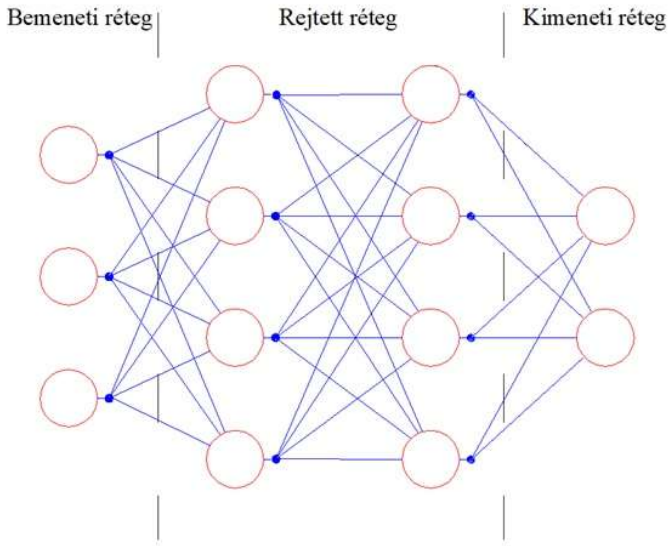
A második generációs neurális hálózatok olyan mesterséges neuronokat (8. ábra) használnak, amelyek folyamatos értékű (analóg) aktivációs függvényeket (szigmoid, tanh stb.) tartalmaznak, és ezáltal lehetővé teszik a nemlineáris transzformációkat. Valamint támogatja a gradiens-alapú tanulási algoritmusokat (pl. backpropagation), amely lehetővé teszi, hogy a hálózat bonyolult elválasztási vagy leképezési feladatokat tanuljon meg. (Megjegyezzük, hogy ez a neuron is tartalmazhat eltolást (biast).) A második generációs hálózatok – ellentétben a bináris kimeneteket adó első generációs perceptronokkal – képesek folytonos bemenet-kimenet térben működni, nemlineáris

leképezéseket tanulni, nemlineáris mintázatokat modellezni, és így univerzális függvényközelítést biztosítani.



8. ábra. Második generációs idegsejt modell tangens hiperbolikusz aktivációs függvénnyel

A neurális hálózattal történő feladatmegoldásnál leggyakoribb a többrétegű perceptron (MLP – multi-layer perceptron) használata (Altrichter, és mtsai., 2006). A 9. ábra egy többrétegű, előrecsatolt (feedforward) hálózati topológiát mutat be.



9. ábra. Előrecsatolt hálózati topológia

A hálózatot általában irányított gráffal reprezentálhatjuk, ahol a gráf csúcspontjai a neuronok, az élek pedig a neuronok közötti kapcsolatok, amelyek a bemenettől a kimenet felé irányítottak. A hálózatban háromféle neuront különbözetettünk meg:

1. **bemeneti neuronok** (bemeneti réteg): egybemenetű neuronok, a bemenetük a hálózat bemenete, kimenetük más neuronok meghajtására szolgál, jelfeldolgozást nem végeznek,
2. **rejtett neuronok** (rejtett rétegek – hidden layers): mind bemeneteikkel, mind kimenetükkel kizárólag más neuronokhoz kapcsolódnak, processzáló (jelfeldolgozó) típusúak, elvileg tetszőleges számú rétegbe rendeződhetnek,
3. **kimeneti neuronok** (kimeneti réteg): a kimenetük a környezet felé továbbítja az információt, típusuk általában megegyezik a rejtett rétegben található neuronokéval.

A neurális hálózatokat az egyes neuronok közötti összeköttetés alapján két fő csoportba sorolhatjuk:

1. **előrecsatolt** hálózatok, amikor a neuronok kimenetei csak valamelyik utána következő réteg neuronjának bemenetéhez kapcsolódik,
2. **visszacsatolt** hálózatok, amikor hálózat topológiáját reprezentáló irányított gráf hurkokat tartalmaz, amely lehet:
   a. elemi visszacsatolás, amikor egy neuron kimenete saját bemenetéhez kapcsolódik,
   b. laterális visszacsatolás esetében a neuron kimenete ugyanazon réteg valamelyik másik neuronjához kapcsolódik,
   c. rétegek közötti visszacsatolás, amikor több rétegen keresztül történik a visszacsatolás.

Harmadik generációs neurális hálózatok

A harmadik generációs neurális hálózatok (tüskés neurális hálózatok – Spiking Neural Networks, SNN) lényege, hogy a neuronok időbeli tüskékkel (impulzusokkal) kommunikálnak, azaz nemcsak a jel nagysága vagy az átlagos aktivitása számít, hanem az impulzusok pontos ideje is. Ezek a hálózatok sokkal közelebb állnak a biológiai neuronok működéséhez, mint a korábbi generációk, mivel az agy neuronjai rövid elektromos impulzusok, úgynevezett akciós potenciálok vagy tüskék segítségével kommunikálnak (Gerstner & Kistler, 2002).

Az SNN neuron működése a Leaky Integrate-and-Fire (LIF) modellel írható le (Roy, Jaiswal, & Pand, 2019):

1. Integrálás: A beérkező impulzusok (tüskék) növelik a neuron belső feszültségét (membránpotenciálját).
2. Szivárgás (Leaky): Ha nem érkezik elég gyorsan újabb impulzus, a feszültség lassan elszivárog, visszaáll az alapállapotba.
3. Tüzelés (Fire): Ha az impulzusok összege elér egy bizonyos küszöbértéket, a neuron "tüzel", tovább küld egy impulzust a következő neuronoknak, majd a saját állapota nullázódik.

A 3. generációs neurális hálónak jóval alacsonyabb az energiafogyasztása, mivel a neuronok csak egy tüske leadásakor fogyasztanak energiát, valamint egy rövid impulzus energiatartalma jóval alacsonyabb, mint egy folytonos jelé. Az SNN-ek jóval kevesebb áramot fogyasztanak, mint a korábbi generációjú hálók. Ez az alacsony fogyasztás teszi lehetővé, az un. neuromorf hardverek gyártását (pl. Intel Loihi vagy az IBM TrueNorth).

Negyedik generáció (neuromorfikus és neuro-szimbolikus rendszerek)

A 4. generáció a szakirodalomban nem egységesen definiált, inkább koncepcionális kategória, amely a biológiai realizmus és a számítástechnikai implementáció új szintjeit jelöli. Néhány tipikus kategória:

- **Neuromorfikus számítás** (Neuromorphic Computing), amely az emberi agy biológiai szerkezetét és működési elveit másolja le hardveres szinten. Míg a hagyományos számítógépek a Neumann-architektúrára épülnek, ahol a processzor és a memória fizikailag elkülönül egymástól, a neuromorfikus chipekben a számítás és az adattárolás ugyanott történik úgy, ahogy a neuronok és szinapszisok működnek az agyban. A mai mesterséges intelligencia programok hatalmas szerverparkokban futnak, és elképesztő mennyiségű áramot fogyasztanak. Ennek oka, hogy az adatok folyamatosan áramlanak a memória és a CPU/GPU között. Az agyunk ezzel szemben: energiatakarékos, párhuzamos működésű (milliárdnyi műveletet végez egyszerre) és eseményvezérelt (csak akkor fogyaszt energiát, ha ingert dolgoz fel).
- **Moduláris modell** esetén a hálózatot kisebb, viszonylag autonóm alhálózatokra bontják. Minden modul egy-egy specifikus részfeladatért felel (pl. mozgásérzékelés, színfelismerés, élek detektálása).
- **Hierarchikus modellnél** az információ feldolgozása különböző mélységben történik. Alacsony szinteken a hálózat apró részleteket detektál (pl. pixelek, frekvenciák), míg a magasabb szinteken ezeket komplex fogalmakká állítja össze. Egy biológiai analógia: a látórendszerünk működése, ahol az első réteg csak vonalakat lát, a második alakzatokat, felül pedig már felismerjük az arcot.
- **Rekurrencia** az agyban a kapcsolatok nagy része nem előrehaladó (feed-forward), hanem visszacsatolt. Ez teszi lehetővé a munkaemlékezetet és a komplex időbeli sorrendek kezelését.
- **STDP** (Spike-Timing-Dependent Plasticity – tüske- vagy impulzusidőzítés) a hagyományos hardver működése órajellel ütemezett, míg ezeknél az eszközöknél nincs órajel, vagyis az aktív elemek csak akkor aktiválódnak, ha impulzust kapnak, ahogy az agy is működik.
- **Neuro-szimbolikus** integráció (Neuro-symbolic AI) célja, hogy egyesítse a két legnagyobb MI-irányzatot: a Deep Learninget, amely kiváló a mintázatfelismerésben, de fekete dobozként működik, nem tudja megmagyarázni a döntéseit, és így logikai hibákat vét, valamint a klasszikus szakértői rendszerek, amelyek szimbólumokkal, szabályokkal és logikával dolgoznak.

Néhány neuromorfikus hardver:

- **Intel Loihi** / **Loihi 2**: az Intel legújabb neuromorfikus kutatóchipje, amely programozható dinamikával, moduláris csatlakoztathatósággal rendelkező tüskés neurális hálózatokhoz használható,
- **IBM TrueNorth**: alacsony energiafogyasztású chip, amelyet valós idejű kognitív alkalmazásokhoz, például tárgyfelismeréshez terveztek,
- **SpiNNaker**: (spiking neural network architecture) egy neuromorf számítástechnikai platform, amely több mint egymillió ARM mobiltelefon-processzort tartalmazó szuperszámítógép.

A neurális hálózatok tanítása

A neurális hálózatok jellemzője az adaptációs és a tanulási képesség. Ez azt jelenti, hogy viselkedésüket a környezetből nyert ismeretek, tapasztalatok felhasználásával módosítani, javítani tudják. A tanulás lényege, hogy a hálózat struktúráját, illetve a súlyszámait, a tanuló adatok (tanító készlet) segítségével úgy változtassuk, hogy a hálózat válasza optimális legyen (Altrichter, és mtsai., 2006).

A neurális hálózatokat különböző módszerrel taníthatjuk. (Fontos, hogy a háló architektúrája nem határozza meg önmagában a tanulási módot.)

- Felügyelt (ellenőrzött) tanulás (supervised): a tanuló bemeneti adatokhoz ismerjük a kimenetet.
- Nem felügyelt tanulás (unsupervised): nem ismert a kimenet.
- Félig felügyelt tanulás (semi-supervised): kevés bemeneti adathoz ismert a kimenet.
- Önellenőrzött tanulás (self-supervised): a hálózat saját maga generál tanuló adatokat.
- Megerősítéses tanulás (reinforcement learning): jutalomfüggvény alkalmazásával.

Felügyelt tanítás

A felügyelt tanításnál összetartozó bemeneti és kimeneti adatpárok (un. címkézett adatok) állnak rendelkezésre. Az ismert bemenő adatok segítségével úgy állítják be a hálózat paramétereit, hogy a válasz a megadott hibával, vagy annál jobban közelítse az ismert, bemenő adatokhoz tartozó eredményeket. A felügyelt tanulás általában iteratív eljárás, amely a tanuló adatok fokozatos felhasználásával, illetve többszöri

felhasználásával történik. Ezt szokás tanítóval történő tanulásnak is nevezni. Ennek feltétele, hogy a bemenethez tartozó kívánt válaszok rendelkezésre álljanak. Előfordulhat, hogy nem minden bemenethez rendelkezünk korrekt kimenettel, esetleg adott bemenethez több válasz is lehetséges. Ilyenkor az adott bemenetre a hálózat válasza csak annyi, hogy helyes vagy hibás. Az ilyen eljárást szokás kritikussal történő tanulásnak nevezni.

A hálózat tanítása előtt meg kell határoznunk:

1. a rejtett rétegek és az azokban elhelyezkedő neuronok számát,
2. a kezdeti súlyértékeket,
3. tanulási tényező és az elfogadható hiba mértékét,
4. tanuló és tesztelő mintakészletet.

A hálózat méretének megválasztására több elmélet is született, de egzakt megoldás nincs. Annyi bizonyos, hogy a hálózat topológiája függ a bemenő adatok dimenziójától, amitől függ a szükséges tanuló adatok mennyisége is. Ha túl kicsi hálózatot választunk, akkor nem tudjuk megtanítani a hálózatot, ha pedig túl nagyot, akkor a hálózat túltanul (magol), igy nem tud általánosítani, valamint a szükségesnél hosszabb ideig fog tartani a tanulás. A gyakorlatban, a hálózat méretének meghatározásánál kétféle utat választhatunk. Egyrészt választhatunk egy kisméretű hálót, és ha azzal nem sikerül megoldani a feladatot, akkor fokozatosan bővítjük azt. Másrészt kiindulhatunk egy nagy hálózatból, amivel meg tudjuk oldani a feladatot, majd megkíséreljük a méretét csökkenteni. Ezekhez az iterációs eljárásokhoz gyakran használnak evolúciós algoritmusokat.

A tanulás konvergenciájának gyorsaságára hatással van a súlyok kezdeti értékének beállítása. Jelenleg erre sincs egzakt matematikai összefüggés. Ezért általában szokás az egyes súlyokat véletlenszerűen, egy egyenletes eloszlású valószínűségi változó különböző értékeire megválasztani (Altrichter, és mtsai., 2006).

A neurális hálózatok tanításánál a tanulási tényező (learning rate): μ $(0 < \mu < 1)$ határozza meg, hogy a modell milyen mértékben változtassa meg a belső paramétereit (súlyait) minden egyes iteráció során. Vagyis, ez a lépésköz, amellyel a modell az optimális megoldás felé halad. A tanulási tényező megválasztására sincs egyértelműen javasolható egyszerű módszer. A kis μ egy minimumhely pontosabb megtalálását, de lassúbb konvergenciát eredményez, míg a nagy μ esetén a

konvergencia gyorsabb lehet, de a minimumhely megközelítése nehezebb, esetleg a konvergencia sem biztosított (Altrichter, és mtsai., 2006).

A tanulási folyamat magától is leállhat, ha a hiba egy minimum szintet elér, de ez lehet lokális minimum is. Ezért a leállításáról általában nekünk kell gondoskodni. Ezt úgy adjuk meg, hogy meghatározzuk azt a hiba értéket, aminek elérése esetén le fog állni a tanítás.

A tanítás nehézségét az adja, hogy végesszámú tanító adat alapján kell a feladatról általános információt szerezni. Ezért a tanuló algoritmus minősítésénél nemcsak azt kell figyelembe vennünk, hogy a tanuló adatokat milyen pontosan tanulja meg a hálózat, hanem a rendszer általánosító képességét is. A **veszteségfüggvény** adott bemenet mellett minősíti a tanuló rendszert (Altrichter, és mtsai., 2006).

A rendelkezésre álló tanító adatokat célszerű két részre választani. Az egyik felével tanítjuk a hálózatot, a maradékkal pedig kiértékeljük (validáljuk) azt. Amennyiben az ellenőrzés során jelentős eltérést tapasztalunk a hálózat kimenete és az ismert eredmények között, akkor a validáló adatokkal bővítjük a bemenő adatsort, és újra tanítjuk a hálózatot.

A felügyelt tanulásnál leggyakrabban alkalmazott eljárás a hiba-visszaterjesztés (back-propagation) lényege: a láncszabály (chain rule) hatékony alkalmazása a gradiens alapú optimalizáláshoz. Segítségével meghatározzuk, hogy a hálózatban lévő minden egyes súly (w) – és ha van az eltolás (b) – milyen mértékben járult hozzá a végső hibához. A célunk a veszteségfüggvény minimalizálása. Ehhez ismernünk kell a függvény parciális deriváltját minden egyes paraméter szerint, ami megmutatja, milyen irányba és mennyit kell változtatnunk a súlyokon, hogy a hiba csökkenjen. Mivel egy összetett hálózatban a súlyok több rétegen keresztül hatnak a kimenetre, a deriválást visszafelé, a láncszabályt használva végezzük el.

Bázisfüggvényes hálózatok (RBF)

Az RBF (Radial Basis Function) hálózat szintén felügyelt tanítású. E hálózatok rejtett rétegei olyan neuronokat tartalmaznak, amelyeknél minden bemenet közvetlenül a nemlineáris elemre jut. A rejtett neuronok kiszámítják a bemenetek és egy középpont közötti távolságot. A bázisfüggvényes hálózatokra jellemző, hogy a bázisfüggvény is tartalmazhat paramétereket, amelyeket a háló felépítésénél meg kell adni. (Altrichter, és mtsai., 2006). Ezeknél a hálózatoknál csak a kimeneti rétegben van összegzés.

Konvolúciós neurális hálózatok (CNN)

A felügyelt tanítású kategóriába sorolható a CNN (Convolutional Neural Networks) is, amelyet elsősorban képfeldolgozáshoz alkalmaznak. Egy tipikus konvolúciós hálózat – a hatékonyság növelése érdekében – nem egyetlen nagy egység, hanem különböző típusú rétegek egymásra épülő sorozata (LeCun, Bottou, Bengio, & Haffner, 1998).

A neurális hálózatok tanításánál fontos paraméterek: a paraméterek száma, az adatok dimenziója, a modellkapacitás és a regularizáció foka, ettől függ a szükséges tanulóadat mennyisége is. Minél összetettebb egy feladat, annál nagyobb hálózatra van szükség, amihez pedig egyre nagyobb mennyiségű tanuló adat kell. Ez két problémát is felvet: az egyik, hogy nem biztos, hogy rendelkezésre áll a kellő mennyiségű tanulóadat, a másik pedig a hosszú tanítási idő. A megoldás a bemenő adatok egyszerűsítése.

Hasonlóan a képtömörítő eljárásokhoz, itt is a szomszédos pixeleket vizsgáljuk. (A képen két távoli pont minimális hatással van egymásra.) Ezt a feladatot látják el a konvolúciós[6] rétegek, amelyek használatával a feldolgozandó adathalmaz dimenziószáma csökken.

Minden konvolúciós kimenet után nemlineáris aktivációt alkalmaznak (pl. ReLU), ami lehetővé teszi nemlineáris mintázatok tanulását.

A konvolúció során az eredeti adatok dimenziószáma csökkenhet, viszont ennél jelentősebb redukcióra van szükség. Erre szolgál a pooling (összevonás) művelet. Leggyakrabban használt a Max-Pooling, ami egy adott konvolúciós rétegtől érkező jelek közül csak a legnagyobbat tartja meg. (Ez abban is segít, hogy a hálózat akkor is felismerje a tárgyat a képen, ha az kissé torz vagy homályos.

Ezeknek a rétegeknek egymás utáni alkalmazásával egyre absztraktabb jellemzőket tanul meg a hálózat: pl. az első rétegek még csak egyszerű éleket vagy sarkokat ismernek fel, a további rétegek viszont már bonyolult formákat, alakzatokat.

A hálózat vége egy hagyományos MLP neurális hálózat, ami meghozza a végső döntést (pl. mit ábrázol a kép).

[6] A konvolúció matematikai értelemben két függvényen végzett művelet, amelyből egy harmadik függvény jön létre. Neurális hálók esetében a konvolúció nem más, mint egy matematikai szűrő.

Nem felügyelt tanulás

Nem felügyelt tanulás (felügyelet nélküli tanulás) olyan adatelemzés, amikor azt próbáljuk felderíteni, hogy az adatok hordoznak-e valamilyen információt (összefüggést). Ekkor nem állnak rendelkezésre adott bemenethez tartozó válaszok. A hálózatnak a bemenetek és a kimenetek alapján kell valamilyen viselkedést kialakítania. A hálózatnak fel kell derítenie a bemeneti adatok közötti mintákat, hasonlóságokat, korrelációt, csoportokat. Ilyenkor a hálózat által megvalósítandó leképzés pontosan nem definiálható. Ezért a hálózat képes önmaga módosítására, annak érdekében, hogy a kategorizálás egyre sikeresebb legyen. Ezeket a hálózatokat szokás önszervező hálózatoknak is nevezni.

Az előre ismert válaszok hiánya miatt a nem felügyelt tanítású hálózatok konstrukciója és alkalmazási területük jelentősen eltér a felügyelt tanulásúéktól. A nem felügyelt tanítású hálózatok jelentős része lineáris háló, azaz a neuronok csak egyszerű súlyozott összegzést végeznek. Ezen hálók főbb alkalmazási területei (Altrichter, és mtsai., 2006):

- a bemenetre kerülő minták közötti hasonlóság megállapítása,
- a bemeneti adatokban csoportok, klaszterek kialakítása,
- adattömörítés, főkomponens analízis,
- független komponensek meghatározása.

A felügyelet nélküli tanulású hálózatok többségének tanítása a **Hebb szabályon** alapul. ami egy egyszerű, biológiai ihletésű tanulási szabály, amely neuronok együttes aktivitása alapján erősíti a kapcsolataikat (Hebb, 1949). Manapság a Hebb tanulási szabálynak különböző változatait alkalmazzák, amelyek közös jellemzője, hogy beépítettek egy normalizáló eljárást, ugyanis az eredeti Hebb szabály mellett a súlyok korlátlanul nőhetnek.

A **versengő tanulás célja**, hogy hálózat neuronjai közül egy győztest válasszunk ki. A győztes neuron kimenete aktív lesz (bináris 1), a többi neuron passzív marad (bináris 0). A tanulás két lépésből áll. Először a háló összes neuronjának kimenetét meghatározzuk az aktuális súlyok alapján, majd kiválasztjuk a győztes neuront. A tényleges tanítást (a súlyvektorok módosítást) csak a győztes neuronnál, illetve annak közvetlen környezetében végezzük, általában a Hebb szabály alkalmazásával.

A **Kohonen háló** szintén biológiai eredetű modell, amely a szomszédos neuronok közvetlen kapcsolatán alapul. A háló alapeleme egy lineáris összegző funkciót megvalósító processzáló tag, amely egyetlen rétegben (rácsban) elhelyezkedő neuronokat tartalmaz. Minden bemenet az összes neuronhoz kapcsolódik. A hálózat nemcsak előre csatolást, de visszacsatolást is tartalmaz, ez azonban csak a közvetlen szomszédos neuronokra, illetve önmagukra vonatkozik. A Kohonen háló tanítása a versengés felhasználásával, a Hebb szabályon alapul. Ahhoz, hogy a háló többi neuronját is bevonjuk a tanításba, nemcsak a győztes neuron súlyát változtatjuk, hanem a többiét is, de kisebb μ-vel, így a vesztes neuronok is esélyt kaphatnak, hogy győztessé váljanak (Altrichter, és mtsai., 2006).

Önellenőrzött tanulás

Az önellenőrzött (self-supervised learning – SSL) tanulás formálisan a nem felügyelt tanulás egyik modern irányzata, ahol a rendszer saját maga generál címkézett tanulási adatokat.

A hagyományos felügyelt tanításnál nagy nehézséget és költséget jelent az adatok beszerzése (a költség logaritmikusan nő a címkézett adatok növekedésével). Ez szűk keresztmetszetet jelent a hálózat folyamatos fejlesztése szempontjából, és alapvető akadályt jelentett az olyan alkalmazási területeken, ahol a címkézett adatok eredendően ritkák, költséges, veszélyes vagy időigényes a gyűjtésük. Szemléltetésképpen vegyünk egy példát az orvostudomány területéről: ahol a címkézetlen adatok könnyen beszerezhetők rutinszerű orvosi vizsgálatok vagy laboreredmények segítségével, viszont az adatokhoz kapcsolódó diagnózisok a nagyszámú esethez való egyedi hozzárendelése (adatok címkézése) jelentős terhet ró az orvosszakértőkre (Gui, és mtsai., 2024).

Az önellenőrzött tanulás minőségét leggyakrabban a kevés címkézett adattal, előzetesen betanított modell segítségével értékeljük.

Tipikus felhasználás: kontrasztív tanulás[7], maszkolt bemenetek, nagy nyelvi modellek, képfelismerés előtanítása.

[7] A kontrasztív tanulás olyan módszer, amely a hasonló mintákat közel, a különbözőket pedig távol helyezi el a tanult reprezentációs térben.

Félig felügyelt tanulás

A gépi tanulás területén a félig felügyelt tanulás (semi-supervised learning – SSL) köztes helyet foglalja el a felügyelt tanulás (amelyben minden tanuló adat címkézett) és a felügyelet nélküli tanulás (amelyben nincsenek címkék) között. Vagyis, mind címkézett, mind címkézetlen adatokat használ. A félig felügyelt tanulás esetén kevés bemeneti adathoz ismert a kimenet. Ez lehetőséget ad modellek felépítésére akkor is, ha a címkék hiánya vagy magas költsége miatt kevés címkézett adat, ugyanakkor nagy mennyiségű címkézetlen adat áll rendelkezésre.

A félig felügyelt tanulás iránti érdeklődés az elmúlt években megnőtt, különösen az olyan alkalmazási területeken, ahol nagy mennyiségben találhatók címkézetlen adatok, mint például a képek, a szöveg és a bioinformatika (Chapelle, Scholkopf, & Zien, 2009).

Megerősítéses tanulás

Az élőlények tanulhatnak tanító segítségével, illetve a környezetükkel való interakció révén, ilyenkor közvetlen szenzomotoros kapcsolatban állnak a környezetükkel. Vagyis amikor valamilyen tevékenységet végeznek, a cselekedeteik következményeiről a környezetéből választ kapnak, ami lehet pozitív (jutalmazó) vagy negatív (büntetető).

A megerősítéses tanulás (reinforcement learning – RL) elsősorban az interakcióból származó célirányos tanulást jelenti, amely magába foglalja annak megtanulását, hogy mit kell tenni és hogyan kell az adott helyzetben lereagálni, a jutalom maximalizálása érdekében. A tanulás során a hálózatnak fel kell fedeznie, mely műveletek hozzák a legtöbb jutalmat. A kihívást az jelenti, hogy a műveletek nemcsak az azonnali jutalmat, hanem a következő helyzetet, és ezen keresztül az összes későbbi jutalmat is befolyásolhatják (Sutton & Barto, 2015).

A tanulót (hálót) és a döntéshozót ágensnek nevezzük, azt amivel kölcsönhatásba lép, környezetnek hívjuk. Ezek folyamatosan kölcsönhatásban vannak, az ágens cselekvéseket választ, a környezet pedig ezekre a cselekvésekre reagál: jutalmaz vagy büntet.

Vegyünk egy konkrét (egyszerű) robotvezérlési példát: egy kéttagú robotkar pozicionálása. Cél: a robot végpontja (end-effector) érje el a céltárgyat. Ágens az adott pozícióból elindítja a robotkart (kiadja a motorvezérlőknek a parancsot). A környezet

szenzorok segítségével érzékeli a kar helyzetét. Jutalom számítása: közelítés a tárgyhoz jutalompont, távolodás, vagy túl nagy sebesség büntetőpont, ha elérte a tárgyat, akkor egy magas jutalompont. (Egy bonyolult robot esetén több különböző ágens működhet egyszerre.)

RL-hálózat lényegében egy függvény-approximátor[8], feladata, hogy egy adott állapotból megjósolja a legjobb cselekvést vagy az abból származó várható jutalmat. A bemeneti réteg az aktuális állapot reprezentációját fogadja. (Ez lehet egy szenzoros adat vagy egy kép pixelmátrixa. A rejtett rétegekben történik a jellemzők kinyerése. Képi input esetén gyakran konvolúciós neurális hálózatot (CNN) használnak. Ha időbeli összefüggések is számítanak, akkor LSTM vagy más visszacsatolt rétegeket is tartalmaz a hálózat. A kimeneti réteg határozza meg a hálózat típusát, amely lehet lineáris vagy valószínűségi eloszlású.

A megerősítéses tanulást (RL) akkor használjuk, amikor a probléma nem egy egyszerű kérdés-válasz jellegű feladat (pl. képfelismerés), hanem ha nincs előre gyártott adatbázis, nem tudjuk megmondani minden helyzetben, mi a tökéletes válasz (lépés), a pillanatnyi döntés befolyásolja a jövőbeli lehetőségeket, illetve, ha a környezet változik, az ágensnek alkalmazkodnia kell.

A neurális hálózatok típusai és felhasználási területe

A következő típusok jellemzően felügyelt tanulású hálózatok, de az architektúra önmagában nem zár ki más tanulási paradigmákat.

CNN (Convolutional Neural Network – konvolúciós neurális háló)

Olyan neurális háló, amely elsősorban képfeldolgozásra alkalmas. Konvolúciós rétegeket használ, amelyek automatikusan tanulják meg a képekben lévő mintázatokat (élek, formák, objektumok).

Tipikus felhasználás: szegmentálás, detektálás, képfelismerés, képosztályozás, arcfelismerés, orvosi diagnosztika (röntgen-, MRI- vagy CT-felvételeken elváltozások azonosítása).

[8] A függvény-approximátor egy olyan modell, amely ismert bemeneti-kimeneti adatpárok alapján megtanulja egy ismeretlen, komplex összefüggés (függvény) közelítő alakját.

LSTM (Long Short-Term Memory – hosszú rövid távú memória)

Az RNN egy speciális típusa, amely jobban kezeli a hosszú távú függőségeket, és csökkenti az ún. eltűnő gradiens problémát.

Tipikus felhasználás: szövegelemzés, nyelvi modellezés, fordítás, idősorok elemzése.

MLP (Multilayer Perceptron – többrétegű perceptron)

A legegyszerűbb, teljesen összekapcsolt neurális háló több rejtett réteggel. Egy két rejtett réteget tartalmazó teljes hálót szemléltet a 9. ábra.

Tipikus felhasználás: klasszikus osztályozási és regressziós feladatok, behatolásjelzés, rosszindulatú kódok felismerése.

RNN (Recurrent Neural Network – rekurzív (visszacsatolt) neurális háló)

Olyan neurális háló, amely képes idősoros vagy szekvenciális adatok kezelésére, mivel az előző állapotok információját továbbviszi.

Tipikus felhasználás: szöveg- és beszédfelismerés, szövegfeldolgozás, érzelemalapú szövegelemzés, gépi fordítás, idősor-előrejelzés.

SNN (Spiking Neural Networks – impulzus vezérelt neurális háló)

Az SNN hálókban elhelyezkedő neuronok nem folytonos, hanem impulzusos működésűek, ezért rendkívül alacsony az energiafelhasználásuk. Így a hardveres megoldásokban kiemelkedő szerepet játszik.

Tipikus felhasználás: orvosi diagnosztika, képfeldolgozás, valós idejű szenzoradatok feldolgozása (pl. önvezető autók kamerái), illetve, ahol a működéshez nem áll rendelkezésre nagymennyiségű energia, de szükséges a hosszútávú üzemidő (pl. orvosi implantátumok, lenyelhető diagnosztikai eszközök, okosotthon, robotika).

SSD (Single Shot MultiBox Detector – egylépéses többdobozos detektor)

Gyors objektumdetektáló módszer, amely egyetlen előrecsatolt hálóval azonosítja és lokalizálja az objektumokat több méretskálán.

Tipikus felhasználás: objektumdetektálás, valós idejű képfeldolgozás, beágyazott rendszerek

Transformer

Egy modern neurális háló architektúra, amely önfigyelésen (self-attention) alapul, és hatékonyan kezeli a hosszú szekvenciákat párhuzamos feldolgozással. Gyakran

először önellenőrzötten tanul (pl. nyelvi modellek esetén), majd felügyelt pontosítás következik.

Tipikus felhasználás: nyelvi modellek (pl. GPT, BERT), fordítás, osztályozás, regresszió, kódgenerálás.

YOLO (You Only Look Once – „csak egyszer nézel rá”)
Valós idejű objektumdetektáló algoritmus, amely egyetlen neurális háló futtatással végzi el az objektumok felismerését és lokalizálását.

Tipikus felhasználás: Objektumdetektálás (gyalogosok, táblák felismerése), videóelemzés, autonóm járművek, megfigyelőrendszerek.

Összegzés

A neurális hálózatok fejlődése jól mutatja, hogyan közelít a mesterséges intelligencia egyre inkább a biológiai információfeldolgozás komplexitásához. Az első generáció egyszerű logikai modelljeitől a mélytanuláson át a tüskés és neuromorfikus rendszerekig tartó fejlődés nem pusztán technológiai előrelépés, hanem szemléletváltást is jelent. A hangsúly a merev algoritmikus szabályokról az adaptív, tanuló rendszerek irányába tolódott.

A modern neurális hálózatok már nem csupán mintázatfelismerő eszközök, hanem komplex döntéshozó és reprezentációtanuló rendszerek, amelyek képesek nagyméretű, strukturálatlan adatokból magas szintű absztrakciókat létrehozni. Ugyanakkor a nagy modellek megjelenésével új kihívások is előtérbe kerültek: az energiahatékonyság, az értelmezhetőség, a megbízhatóság és az etikai kérdések.

A jövő valószínűleg a különböző paradigmák integrációja lesz, vagyis a mély tanuló (deep learning) neurális hálózatok, a neuromorfikus architektúrák és a szimbolikus következtetés ötvözése. Ez olyan rendszerekhez vezethet, amelyek nemcsak tanulnak, hanem strukturált módon érvelni is képesek.

A neurális hálózatok kutatása így nem lezárt fejezet, hanem egy dinamikusan fejlődő, interdiszciplináris tudományterület, a biológia, a matematika, az informatika és a mérnöktudomány határán.

Evolúciós algoritmusok

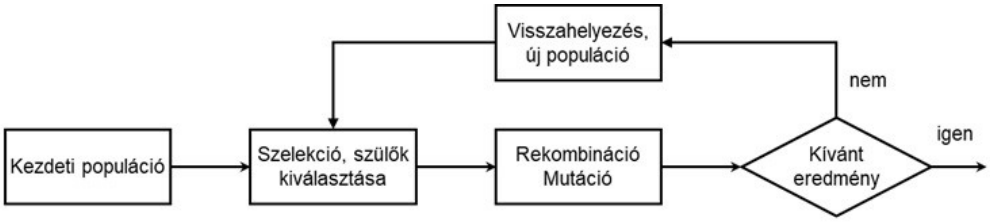
A biológiai fejlődésben döntő szerepet játszott az evolúció, a természetes kiválasztódás és a mutáció. Ennek során az élőlények „megtanultak” alkalmazkodni a változó környezethez. Ez nem sikerült mindegyiknek, így amelyik nem tudott alkalmazkodni, az kihalt.

E folyamat számítógépes modelljei az evolúciós algoritmusok. Az evolúciós algoritmus (EA) a mesterséges intelligencia egyik kiemelt jelentőségű részterülete, amely az evolúció elvén, azaz a legalkalmasabb túlélésén alapul. Ezen algoritmusokon alapuló evolúciós programozási technikák számos olyan optimalizálási feladatra használhatók, ahol a hagyományos matematikai módszerek túl lassúak vagy nem alkalmazhatóak: mint például a lineáris és nemlineáris korlátokkal rendelkező függvények optimalizálása, az utazó ügynök problémája, valamint az ütemezés, a particionálás és a vezérlés problémái (Michalewicz, 1996). Az algoritmus alapja az iteráció, amely során a lehetséges megoldások halmazát (populációját) fokozatosan javítjuk.

Ahhoz, hogy megértsük az EA működését, definiálnunk kell a biológiai analógián alapuló fogalmakat (Borgulya, 2004):

- Egyed (individuum): a populáció egyetlen eleme, amely egy lehetséges megoldást reprezentál. Formai szempontból lehet: sztring, vektor, illetve bármilyen összetett struktúra.
- Populáció: az egyedek egy adott generációban létező halmaza.
- Generáció: azon egyedek összessége, amelyek egy adott időpillanatban (ciklusban) együtt léteznek a populációban.
- Fitneszfüggvény: megadja egy adott egyed "jóságát" vagy életképességét. (Optimalizálási feladatnál lehet célfüggvény is.) A fitneszfüggvény kiértékelésével dől el, mely egyedek maradnak fenn. Vagyis szerepet játszik a szülők és az életképes utódok kiválasztásában.
- Szelekció: az egyedek kiválasztása a populációból. (Vonatkozhat a szülőkre, de az utódokra is.) Figyelembe veszi a fitneszfüggvény értékét, így a rátermettebb egyedek nagyobb súllyal (valószínűséggel) kerülnek kiválasztásra.
- Rekombináció (keresztezés): az utódok előállítása egy vagy több szülő felhasználásával, az információ tartalmuk kombinálásával.

- Mutáció: az utódok egy-egy jellemzőjét véletlenszerűen megváltoztatja.
- Visszahelyezés: az utódok létrehozása után az új populáció kialakítása. Az új populáció kialakítása többféle képen történhet: minden egyedet lecserélünk az előző generációban, csak bizonyos arányban cserélünk, vagy a fitneszfüggvény alapján tartunk meg, illetve cserélünk. Legyen az utódképzési ráta (U), jelentése az utódok száma hányszorosa a populációnak. Továbbá legyen a visszahelyezési ráta (V), a populáció egyedeinek hányad részét cseréljük le az utódokra. Ekkor, ha:
    - V=1: teljesen új populáció jön létre,
    - V<1: adott számú egyed cserélődik,
    - U<=V: minden utód visszakerül a populációba,
    - U>V: az utódoknak csak egy része kerül vissza a populációba.
- Generációs ciklus: az a folyamat, amelynek során a műveletek (szelekció, rekombináció, mutáció) elvégzésével egy új generáció jön létre (10. ábra).



10. ábra. Generációs ciklus

Az algoritmus megállásának feltételei (Borgulya, 2004):

- maximális generációszám elérése,
- maximális futásidő elérése,
- adott idő vagy adott számú ciklus után nem javul a megoldás minősége,
- nagyon hasonlóak az egyedek (a rekombináció hatástalan),
- előre adott érték megközelítése vagy elérése,
- a populáció minősége megfelelő.

Az evolúciós algoritmus típusai

Az evolúciós algoritmus egy gyűjtőfogalom, amely a populációalapú sztochasztikus direkt keresési algoritmusok leírására szolgál (Bartz-Beielstein, Branke, Mehne, & Mersmann, 2014). Ez egy keretrendszer, amely igazítható egy adott feladathoz. A különböző paraméterek megválasztása, a tulajdonságok reprezentációja, a műveletek megválasztása, és maga a feladat is befolyásolják az alkalmazott módszer kialakítását

(Borgulya, 2004). Így az evolúciós algoritmusok több alapvető kategóriára bonthatók, amelyek eltérő reprezentációt és operátorokat használnak:

- Genetikus algoritmusok (GA): a legismertebb és legelterjedtebb EA-típus, ahol a megoldásokat tipikusan kromoszóma-szerű bit- vagy vektorsorozatok reprezentálják, a populáció keresztezés, mutáció, szelekció kombinációjával fejlődik.
- Evolúciós stratégiák (ES): főként folytonos problémákra optimalizált változat, a hangsúly a mutáción és a rekombináción van.
- Evolúciós programozás (EP): hasonló az ES-hez, de a konkrét műveletekben jelentős eltérést mutatnak. Eredetileg automaták vagy programok evolúciójára lett kifejlesztve, később optimalizációs változatokkal is bővült.
- Genetikus programozás (GP): a GA-ból származtatott módszer, amely programok vagy kifejezések automatikus fejlesztését valósítja meg. Minden egyed egy-egy komplett programot ír le.
- Differenciális evolúció (DE): az ES távoli rokonának tekinthető, egy hatékony, vektor-alapú keresőalgoritmus, amely az egyedek közötti különbséget használja fel a mutációhoz (különbségi mutáció).

Ezeken kívül a szakirodalmak több hibrid vagy továbbfejlesztett EA-típust is megemlítenek.

Genetikus Algoritmusok (Genetic Algorithms - GA)

Ez a legismertebb és legelterjedtebb típus, amely a biológiai DNS-ábécéhez hasonlóan eredetileg karakterlánc (bitsztring) reprezentációkra épült, de alternatív kódolásokat is használtak, mint például a valós számokkal történő kódolást.

Egy *k* hosszúságú bitsztring megfelel az egyed genotípusának[9]. Az egyed fenotípusát[10] az objektumparaméterekre való leképezéssel, az úgynevezett genotípus-fenotípus leképezéssel[11] valósítják meg.

A következő példában egy egyszerű egyváltozós optimalizálási feladatot mutatunk be. Nézzünk egy szemléltetés szakirodalmi példát (Goldberg, 1989): legyen egy 4

[9] A genotípus egy élőlény egyedi genetikai felépítése, a DNS-ben kódolt összes öröklött tulajdonság összessége.

[10] A fenotípus az élő szervezetek megfigyelhető, mérhető külső és belső tulajdonságainak (morfológiai, élettani, viselkedési) összessége

[11] A genotípus-fenotípus leképezés a genetika egyik alapfogalma, amely azt írja le, hogyan alakulnak át az élőlény örökletes információi (genotípus) a ténylegesen megfigyelhető fizikai vagy funkcionális tulajdonságokká (fenotípus).

egyedből álló kezdeti populációnk, ahol az egyedeket egy-egy 5 elemű bitsorozat reprezentálja, a fitneszfüggvény legyen $f(x)=x^2$:

Bináris kromoszóma	Decimális x	$f(x)=x^2$
01101	13	169
11000	24	576
01000	8	64
10011	19	361

(Megjegyezzük, hogy a fitneszfüggvényeknél (rátermettség mérésénél), gyakran használjuk a négyzetre emelést mert felnagyítja a különbséget a jó és a kevésbé jó megoldások között.)

A szelekciónál cél, hogy a nagyobb fitneszű egyedek nagyobb valószínűséggel váljanak szülővé. Esetünkben a 11000 és a 10011 kóddal reprezentált egyed.

Következő lépés a keresztezés, amely a következőképpen valósítható meg:

- **Egypontos** keresztezés: véletlenszerűen kiválasztunk minkét bitsorozatnál egy metszési pontot, az egyik utód az első szülő metszéstől balra lévő, míg a másik szülőtől a jobbra lévő biteket (géneket) örökli, a másik utódnál fordítva.
- **Többpontos** keresztezés: az egypontos kereszteződéstől eltérően, itt több vágási pontot határozunk meg, majd minden bitcsoportnál külön-külön döntjük el, hogy azt melyik utód fogja ürükölni.
- **Uniform** keresztezés: ebben az esetben minden elemről (bitről) külön-külön véletlenszerűen döntjük el, hogy azt melyik szülőtől választjuk. Az utód minden bit (gén) pozíciójában egy véletlen bináris maszk határozza meg az öröklést. (Pl. ha a bitmaszk adott bitje 1, akkor az első szülőtől, ha nulla, akkor a második szülőtől vesszük a bit értékét.)
- **Keverő** keresztezés: kiküszöböli a két vagy többpontos keresztezésnek azt a hibáját, miszerint a kromoszóma (bitsorozat) két szélén lévő gének ritkán maradnak együtt. Véletlenszerűen összekeverjük a gének sorrendjét mindkét szülőnél (ugyanazt a keverési mintát használva mindkettőnél). Elvégzünk egy hagyományos (pl. egypontos) keresztezést, majd az utódban a géneket visszatesszük az eredeti pozíciójukba.

Keresztezés típusa	1. szülő	2. szülő	1. utód	2. utód
Egypontos	110|00	100|11	11011	10000
Többpontos	1|100|0	1|001|1	10010	11001
Uniform	11000	10011	10010	11001
Keverő	11000	10011	10001	10110

(Az uniformnál a bináris maszk: 10101, a keverőnél a sorrend 2, 0, 4, 1, 3.)

A továbbiakban vegyük az egypontos keresztezést.

A következő lépés a mutáció, ahol kis valószínűséggel (pl. 1%) a bitek ellentétjükre váltanak. (Pl. a 2. utód 0. bitje mutálódik, vagyis 10001 lesz.)

Ezután következik a visszahelyezés, vagyis az utódok hozzáadása az eredeti populációhoz. Ez többféle keppén valósítható meg:

- minden utóddal bővül a populáció,
- csak az adott fitneszértéket elérő utódokkal bővül a populáció,
- az eredeti populáció egyedeit megtartjuk,
- az eredeti populációban a fitneszérték alapján kiszelektáljuk a kevésbé rátermetteket,
- a teljes generációt lecseréljük az utódokkal.

Evolúciós Stratégiák (Evolutionary Strategies - ES)

Az 1960-as években Németországban dolgozták ki, kezdetben mérnöki optimalizálási feladatokra, többek között aerodinamikai formatervezésre (Rechenberg, The Evolution Strategy. A Mathematical Model of Darwinian Evolution, 1973). Az ES a feladat végrehajtása során a mutációt helyezi előtérbe, annak mértékét az optimalizálás során dinamikusan változtatja. Az algoritmus menet közben tanulja meg, mekkora mértékű mutáció az ideális.

Egyedek leírásánál, míg a GA a bitekre fókuszál, az ES inkább valós számokkal (lebegőpontos vektorokkal) dolgozik. Minden egyednek egy *n* elemű vektor felel meg, így az *m* darab egyedből álló populáció egy *nxm* dimenziós vektorral írható le. E mellett minden egyed tartalmaz egy ún. mutációs lépésközt (ez szintén evolválódik), amely egyedenként eltérő lehet (Borgulya, 2004).

Szelekció a fitness-függvény alapján, bármely egyedet többször is választhatjuk szülőnek.

Mutációnak több változata lehetséges attól függően. hogy hogyan képezzük az egyes változók szórását (mutációs lépésközt):

- Minden változónál (vektorelemnél) azonos értékű a szórás, ekkor a változókat egy normál eloszlású véletlenszámmal módosítjuk.
- Minden változóhoz egyedi, eltérő értékű szórás tartozik.

Visszahelyezés művelet teljesen determinisztikus, ahol a μ számú szülőből és a λ számú utódból a következő módon választhatunk:

- A (μ, λ) szelekciónál csak az utódok versenyeznek: λ számú utódból választjuk ki a μ számú legjobb egyedet, amelyek az új populációt alkotják. Nincs garancia arra, hogy a legjobb egyedek kerülnek be az új populációba, minden egyed csak egy populációt él. Előnye az erősebb szelekciós nyomás és a szélsőértékek elhagyása, mivel a legjobb megoldások nem maradnak folyamatosan a populációban.
- A $(\mu + \lambda)$ szelekciónál a szülők és az utódok egyesített halmazából választjuk ki a μ számú legjobb egyedet, amelyek az új populációt alkotják. Előnye a stabilabb konvergencia és az erősebb elitizmus (a legjobbak maradnak).

Tipikus alkalmazási területek:

- mérnöki tervezési optimalizálás,
- robotika,
- hiperparaméter-hangolás (a gépi tanulási folyamat megkezdése előtt beállított paraméterek),
- neurális hálók tanítása (pl. gradiens nélküli RL),
- evolúciós neurális architektúra keresés.

Evolúciós programozás (Evolutionary Programming - EP):

Az EP az egyedek megfigyelhető viselkedésére (fenotípus) és annak adaptációjára összpontosít, hasonló az ES-hez, de a konkrét műveletekben jelentősen eltérnek, nincs keresztezés, csak mutáció. Eredetileg a mesterséges intelligencia viselkedésének modellezése, véges állapotú automaták, illetve programok evolúciójára fejlesztették ki, később optimalizációs változatokkal bővítették.

Az egyedek itt is valós vektorok, de azok szórás értéket nem tartalmaznak, helyettük $2n$ darab (minden egyedhez két-két darab) ún. stratégiai paramétert használunk a mutációnál. A mutációs lépéshez standard normál eloszlást alkalmazunk.

Kezdőértékként egy véletlenszerű megoldáshalmazt (pl. véges állapotú automatát vagy számsort) adunk meg.

Mivel a klasszikus EP nem használ keresztezést (crossover), a populáció diverzitását és az új megoldások létrehozását szinte teljes egészében a mutáció biztosítja (Fogel, Owens, & Walsh, 1966).

Visszahelyettesítésnél μ számú szülő és μ számú utód közötti verseny dönti el, hogy mely egyedek fogják alkotni az új populációt. Az EP-ben gyakran verseny alapú szelekciót (tournament selection) alkalmaznak.

Az evolúciós programozás különösen olyan területeken használható, ahol a keresési tér nagy és nemlineáris, nincs deriválható célfüggvény, illetve sok lokális optimum van.

Tipikus alkalmazási területek:

- optimalizáció (mérnöki tervezés),
- vezérlőrendszerek,
- jel- és képfeldolgozás,
- gépi tanulás,
- robotika,
- bioinformatika.

Genetikus Programozás (Genetic Programming - GP)

A számítógépes programok az ember által alkotott egyik legösszetettebb és legbonyolultabb struktúrák közé tartoznak. A számítógépes programok elkészítése általában nehéz feladat, amit a programozók sorról sorra írnak meg egy adott feladat megoldására. Már az 1950-es években felmerült a kérdés, hogy hogyan tanulhatnak meg a számítógépek problémákat megoldani anélkül, hogy explicit módon programoznák azokat. Más szóval, hogyan lehet rávenni a számítógépeket arra, hogy azt tegyék, amit meg kell tenniük, anélkül, hogy pontosan megmondanák nekik, hogyan kell azt csinálni (Koza, 1992).

A genetikus programozás célja olyan programok automatikus létrehozása, amelyek egy adott feladatot megoldanak anélkül, hogy azt explicit módon leprogramoznánk. Ehelyett hagyjuk, hogy a gép maga készítse el a megoldáshoz szükséges algoritmust, illetve a programot. Vagyis itt nem adatokkal, hanem a számítógépes programokkal (vagy matematikai formulákkal) végezzük el az evolúciós műveleteket.

Az egyedek ebben az esetben programok vagy programfák (pl. AST[12]-fa). A kezdő populációban nagy számú egyedet (programot) hozunk létre véletlen módon.

A szelekciós művelet a rekombinációhoz kapcsolódva kerül végrehajtásra. Annak megfelelően választ egyedet, hogy a rekombinációhoz egy vagy két szülő szükséges. Leggyakrabban a fitnesz arányos vagy versengő szelekciót alkalmazzuk. A rekombináció leggyakrabban egypontos művelet, ahol például a kijelölt részfát a két szülőnél kicseréljük (Borgulya, 2004).

A GP-nél számos mutáció eljárás ismert, amelyek közül a GP többféle változatot szimultán alkalmaz (Borgulya, 2004):

- Részfa mutáció esetében véletlenszerűen kiválasztunk egy részfát, amit egy véletlen kialakított részfával lecseréljük.
- Pont mutáció esetében egy csomópont tartalmát véletlenszerűen megváltoztatjuk.
- Részfa képzés esetén véletlenszerűen kiválasztunk egy részfát, ami a későbbiekben önálló utód lesz.
- Rövidítés esetén egy véletlenszerűen kiválasztott részfából végpont (levél) lesz.
- Véletlen konstans mutáció, amikor minden konstanst egy normál eloszlású véletlenszámmal megváltoztatunk.

Minden programot egy fitness (alkalmassági) függvény értékel, vagyis futtatja a programot és a kapott eredmények alapján értékel. A jobb programok nagyobb valószínűséggel szaporodnak.

A GP egyik gyakori problémája a bloat, azaz a programok indokolatlan növekedése.

Bár a GP-t programok fejlesztésére dolgozták ki, az sokkal szélesebb körben alkalmazható:

- automatizált programgenerálás, programrészletek evolúciója,
- ahol nem ismerjük a pontos matematikai összefüggést az adatok között, olyan képlet keresése, ami legjobban illeszkedik az adatokra,
- elektronikus áramkörök tervezés,
- robotika,

[12] Absztrakt Szintaxis Fa (Abstract Syntax Tree - AST) a számítógéptudományban a forráskód (program) szerkezetének fa alapú reprezentációja.

- szabályrendszerek, stratégiák kidolgozása,
- gépi tanulási modellek kidolgozása,
- videók, zenék készítése.

Differenciális evolúció (Differential Evolution – DE)

A DE egy stochasztikus, populációalapú globális optimalizáló algoritmus, amely különösen hatékony folytonos paramétertereken végzett optimalizációra. Az egyik leghatékonyabb globális keresőalgoritmus, elsősorban akkor, ha a probléma nem lineáris és nem deriválható. A populáció egyedei közötti távolságot és irányt használja fel az új megoldások kereséséhez. Nem véletlenszerűen mutál, hanem a meglévő egyedek különbségét (differenciáját) adja hozzá egy másik egyedhez. A módszer egyszerűsége, kevés paramétere és robusztussága, valamint párhuzamosíthatósága miatt széles körben használják mérnöki optimalizációs feladatoknál, gépi tanulásnál és numerikus optimalizálásnál. (Storn & Price, 1997).

Az egyedet egy D dimenziós vektor írja le, így az N elemű populáció egyede:

$$\boldsymbol{x}_i = (x_{i,1}, x_{i,2}, x_{i,3}, \cdots x_{i,D}),\ i = 1,2,3, \cdots N$$

Kiindulás egy véletlenszerű egyedekből (vektorokból) álló populáció a keresési téren belül:

$$x_{i,j} = x_j^{min} + rand(0,1)(x_j^{max} - x_j^{min})$$

Mutáció a DE legfontosabb lépése. Minden egyedhez ($\boldsymbol{x}_i$) generálunk egy mutáns vektort ($\boldsymbol{v}_{i,G}$) három véletlenszerűen választott, különböző egyedből.

$$v_{i,G+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G})$$

ahol *F* skálázási tényező, általában [0,1] közötti érték. A különbségvektor $\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3}$ adja a keresés irányát.

Rekombinációnál (crossover) a mutáns vektor elemeit valamilyen valószínűséggel összekeverjük az aktuális (szülő) vektor elemeivel:

$$u_{ij} = \begin{cases} v_{ij} & rand(0,1) \le CR \\ x_{ij} & \text{egyébként} \end{cases}$$

ahol *CR* a crossover ráta.

Szelekció: ha a mutáns vektor jobb eredményt (fitnesz értéket) ad, mint az eredeti szülő, akkor átveszi annak helyét a következő generációban. Ha nem, a szülő marad.

$$x_i^{t+1} = \begin{cases} u_i & f(u_i) \leq f(x_i) \\ x_i & \text{különben} \end{cases}$$

ahol *t+1* a következő generáció.

A DE-t elsősorban olyan feladatoknál alkalmazzák, ahol a keresési tér folytonos és nagy dimenziós, a célfüggvény nemlineáris, nem konvex vagy zajos, a gradiens nem ismert vagy nehezen számolható. Ezek az alkalmazási területek:

- mérnöki optimalizáció (pl. minimális tömeg, maximális szilárdság, szerkezeti optimalizálás, aerodinamikai alakoptimalizálás),
- vezérlőrendszerek hangolása,
- modellparaméterek optimalizálása,
- gépi tanulás és mesterséges intelligencia például neurális hálók súlyainak tanítása (neuroevolution),
- jel- és képfeldolgozás,
- zajszűrés,
- energetikai rendszerek optimalizálása,
- robotika és irányítástechnika,
- gazdasági stratégiák kidolgozása,
- bioinformatika és élettudományok,
- gyártási folyamatok optimalizálása.

Összegzés

Az evolúciós algoritmusok a természetes szelekció elveit felhasználva hatékony eszközt kínálnak komplex optimalizálási problémák megoldására. Rugalmasságuk és általános alkalmazhatóságuk miatt számos tudományterületen és ipari alkalmazásban használatosak, különösen ott, ahol a hagyományos módszerek nem hatékonyak. Bár működésük gyakran számításigényes, folyamatos fejlődésük és az új hibrid megközelítések révén egyre fontosabb szerepet töltenek be a modern mesterséges intelligencia és optimalizálás területén.

Hivatkozások

Altrichter, M., Horváth, G., Pataki, B., Strausz, G., Takács, G., & Valyon, J. (2006). *Neurális hálózatok.* Budapest: Panem.

Bäck, T., & Schwefel, H.-P. (1993). An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary Computation*, 1-23.

Bäck, T., Fogel, D., & Michalewicz, Z. (1997). *Handbook of Evolutionary Computation.* Oxford, England: Oxford University Press.

Bartz-Beielstein, T., Branke, J., Mehne, J., & Mersmann, O. (2014). Evolutionary Algorithms. *WIREs Data Mining and Knowledge Discovery*, 178-195.

Boden, M. A. (2016). *Ai: Its Nature and Future.* United Kingdom: Oxford University Press.

Borgulya, I. (2004). *Evolúciós algoritmusok.* Budapest: Dialóg Campus.

Chapelle, O., Scholkopf, B., & Zien, A. (2009). *Semi-Supervised Learning.* Cambridge, Massachusetts, USA: MIT Press.

Coll, P. (1966). *Már az ókorban is tudták.* Budapest: Gondolat kiadó.

Crevier, D. (1993). *AI: The Tumultuous Search for Artificial Intelligence.* New York: BasicBooks.

Csapó, B. (1992). *Kognitív Pedagógia.* Budapest: Akadémiai kiadó.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* , 182-197.

Domingos, P. (2015). *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World.* London: Penguin Books Limited.

Dreyfus, H. L. (1992). *What Computers Still Can't Do: A Critique of Artificial Reason (átdolgozott kiadás).* Cambridge, Massachusetts, USA: The MIT Press.

Floridi, L. (2019). *The Logic of Information - A Theory of Philosophy as Conceptual Design.* United Kingdom: Oxford University Press.

Fogel, D. B. (2006). Historic perspective - Nils Barricelli-artificial life, coevolution, self-adaptation. *IEEE Computational Intelligence Magazine*, 41–45.

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial Intelligence Through Simulated Evolution.* New York City, United States: Wiley.

Garcıa-Lopez, P., Garcıa-Marın, V., & Freire, M. (2006). Three-Dimensional Reconstruction and Quantitative Study. *Journal of Neuroscience*, 11249 – 11252 .

Gerstner, W., & Kistler, W. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity.* Cambridge, United Kingdom: University Press.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Tuscaloosa, USA: The University of Alabama.

Gui, J., Chen, T., Zhang, J., Cao, Q., Sun, Z., Luo, H., & Tao, D. (2024). A Survey on Self-supervised Learning: Algorithms, Applications, and Future Trends. *IEEE*, 9052-9071.

Hebb, D. O. (1949). *The Organization of Behavior: A Neuropsychological Theory.* New York: Wiley.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* Cambridge, Massachusetts, USA: The MIT Press.

Kandel, E. R., Koester, J. D., Mack, S. H., & Siegelbaum, S. A. (2021). *Principles of Neural Science.* Columbus, Ohio, Egyesült Államok: McGraw-Hill.

Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, Massachusetts, Egyesült Államok: MIT Press.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998. november). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 1-46.

Lighthill, J. (1972. July). *Artificial Intelligence: A General Survey". UK: Science Research Council.* (T. U. Edinburgh, Producer) Letöltés dátuma: 2025. november 25, forrás: Artificial Intelligence Applications Institute: https://rodsmith.nz/wp-content/uploads/Lighthill_1973_Report.pdf

Luciano, F., Josh, C., Monica, B., Raja, C., Patrice, C., Virginia, D., . . . Effy, V. (2018). AI4People—An Ethical Framework for a Good AI Society: Opportunities, Risks, Principles, and Recommendations. *Minds and Machines*, 689–707.

Markowska-Kaczmar, U., & Koldowski, M. (2015). Spiking neural network vs multilayer perceptron: who is the winner in the racing car computer game. *Soft Comput*, 3465–3478.

McCarthy, J., Minsky, M., Newell, A., & Simon, H. (1956. június 18). *Proposal for the Dartmouth summer research project on artificial intelligence*. Letöltés dátuma: 2025. november 18, forrás: Artificial Intelligence Coined at Dartmouth: https://home.dartmouth.edu/about/artificial-intelligence-ai-coined-dartmouth

McCulloch, W., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 115–133.

McCulloch, W., & Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 115–133.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs.* Berlin: Springer Nature.

Minsky , M. L., & Papert, S. A. (1969). *Perceptrons: An Introduction to Computational Geometry.* Cambridge: MIT Press Direct.

Neuron Structure. (2014. otóber 11). Letöltés dátuma: 2026. január 6, forrás: SlideServe: https://www.slideserve.com/kumiko/neuron-structure

Newell, A., & Simon, H. A. (1956). The logic theory machine (A complex information processing system.). *Institute of Radio Engineers, Transactions on information theory*, 61-79.

Rechenberg, I. (1973). *The Evolution Strategy. A Mathematical Model of Darwinian Evolution.* Berlin: Technische Universität Berlin, Fachgebiet Bionik und Evolutionstechnik.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 386–408.

Roy, K., Jaiswal, A., & Pand, P. (2019). Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 607-617.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 533–536.

Russell, S., & Norvig, P. (2020). *Artificial Intelligence A Modern Approach.* Columbia University, United States: Prentice Hall.

Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models Wiley, Chichester. 1981, 389 pp.* Chichester. Anglia: Wiley.

Simon, H. A., & Newell, A. (1956). The logic theory machine - A complex information processing system. *IRE Transactions on Information Theory*, 61-79.

Simonyi, K. (1981). *A fizika kultúrtörténete.* Budapest: Gondolat kiadó.

Stanley, K. O., & Miikkulainen, R. (2002). Evolving Neural Networks through. *Evolutionary Computation*, 99-127.

Storn, R., & Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 341–359.

Sutton, R., & Barto, A. (2015). *Reinforcement Learning: An Introduction.* Cambridge, Massachusetts, USA: MIT Press.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 230-265.

Turing, A. M. (1950. október). Computing Machinery and Intelligence. *Mind* , 433-460. Forrás: http://www.jstor.org/stable/2251299

Turing, A. M. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 37-72.

Turing, A. M. (2004). Intelligent machinery (1948). In B. s. Copeland, *In The Essential Turing;* (old.: 395–432). Oxford, UK: Oxford Academic.

Weizenbaum, J. (1966. January). *ELIZA - a Computer Program for the Study of Natural Language Communication Between Man and Machine.* Letöltés dátuma: 2025. november 24, forrás: University at Buffalo, New York State: https://cse.buffalo.edu/~rapaport/572/S02/weizenbaum.eliza.1966.pdf