Magyar Tudomány 185 (2024) 7, 906–919
DOI: 10.1556/2065.185.2024.7.8

A MESTERSÉGES INTELLIGENCIA ALKALMAZÁSI TERÜLETEI A SZÁMVITELBEN

APPLICATIONS OF ARTIFICIAL INTELLIGENCE IN ACCOUNTING

Polyák Imre
egyetemi docens, Budapesti Corvinus Egyetem Számvitel Tanszék, Budapest
imre.polyak@uni-corvinus.hu

ÖSSZEFOGLALÁS

A mesterséges intelligencia mint hívószó egyre gyakrabban jelenik meg az élet minden területén. Leggyakrabban abban a kontextusban találkozunk vele, hogy a mesterséges intelligencia megszüntethet-e egyes szakmákat. Jelen tanulmányunkban más oldalról vizsgáljuk meg a témát, és a fókuszba egyrészt a mesterséges intelligencia használatához szükséges *üzleti oldali paradigmaváltást* állítjuk, másrészt megvizsgáljuk a mesterséges intelligencia – ennél pontosabb szóhasználattal a gépi tanulás (machine learning, ML) és a mélytanulás (deep learning, DL) – egyes fontosabb területeinek helyét és szerepét a számviteli munkában. Ez a megközelítés azért is fontos, mert a mesterséges intelligencia alkalmazása nem (csak) arról szól, hogy magát a számviteli rendszert tudja-e működtetni autonóm gépi logika. Ezek az eszközök a számvitel egyes részterületein kisegítőként igen hatékony eszközt kínálnak: a bizonylatok előrögzítésére, a becslések és az auditmunka támogatására is hatékony megoldást nyújtanak. Amikor üzleti oldali paradigmaváltásról beszélünk, akkor alapvetően a „hagyományos" algoritmusalapú gondolkodásmód háttérbe szorítására gondolunk. A tanulmányban a tanulási folyamatot és az algoritmusalapú gondolkodással való szembeállítást is megtaláljuk. Mivel a témakör szerteágazó, ezért – a terjedelmi korlátokra figyelemmel – nem lehet cél a teljességre való törekvés, inkább csak az egyes eszközök nagy vonalakban való bemutatása a kapcsolódó számviteli terület azonosítása mellett.

ABSTRACT

Artificial Intelligence (AI) is a buzzword that is increasingly appearing in all walks of life. It is most often used in the context of whether AI could eliminate certain jobs. In this article, we will look at the issue from a different angle, focusing on the paradigm shift required for the use of AI in business, and on the place and role of some of the more important areas of AI – more specifically, machine learning (ML) and deep learning (DL) – in accounting. This approach is also important because the application of AI is not (only) about whether the accounting system itself can be operated by autonomous machine logic. These tools offer a very powerful tool as a back-up in certain areas of accounting: they can be used to pre-record receipts, support estimates, and audit work. When we talk about a paradigm shift on the business side, we are basically talking about a shift away from the "traditional" algorithm-based way of thinking. In the article, we will also discuss the learning

process and contrast it with algorithm-based thinking. Since this is a very broad topic, it is not possible to be exhaustive, given the limitations of space. Rather, only a generous presentation of each tool, with identification of the related accounting area, can be provided.

Kulcsszavak: számvitel, mesterséges intelligencia, gépi tanulás, mélytanulás, neurális háló

Keywords: accounting, artificial intelligence, machine learning, deep learning, neural network

A GÉPI TANULÁS ELMÉLETI ALAPJAI

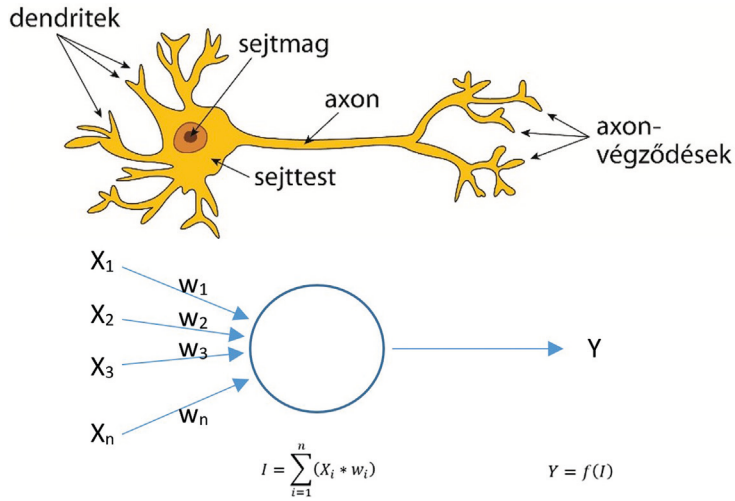
A gépi tanulás (ML) egy általános fogalom, amelynél valamely változó értéket bizonyos, akár sok adatpontos és sokváltozós logika mentén a gép határoz meg. A mélytanulás (deep learning) a gépi tanulási modellek alcsoportja, és olyan mesterséges neurális hálózattal való tanulást jelent, amely több rejtett réteget (esetleg elágazást) tartalmaz.

A gépi tanulás esetében kétféle alaptípust különböztetünk meg. A *felügyelt* (supervised) tanulás esetén adott egy *N* elemű sokaság, amelynél *ismerjük* a kimenetet – mint ún. függő változót. A rendszertől azt várjuk, hogy határozza meg azokat a paramétereket, amelyek mentén – támaszkodva a meglévő észlelésekre – előáll egy olyan modell, amely az újonnan érkező megfigyelésekkel „helyes", megalapozott becslést tud adni a függő változó értékére.

A *nem felügyelt* (unsupervised) tanulásnál nem beszélünk a bemeneti halmaz esetén „helyes kimenetekről", azaz nincs az alapsokaságnak függő változója. Észleléseink ilyenkor is vannak, de ezek csak a modell „bemeneti értékeit" mint független változókat tartalmazzák. (Ezeket az angol terminológia *feature* néven nevesíti.) A géptől várjuk, hogy a függő változót megképezze. Erre példa a klaszterezés. Annyit várunk a géptől, hogy a meglévő észleléseket – akár *n* dimenzió mentén – valahány, minél függetlenebb csoportba rendezze.

A mélytanulás „általános" modelljét a mesterséges neurális háló képezi (artificial neural network, ANN). A mélytanulás elmélete egyáltalán nem új: a modell alapját képező mesterséges neuron működési logikája 1943-ig vezethető vissza, az „alapmodellt" Warren McCullogh és Walter Pitts alkották meg, és a nevük alapján McCulloch–Pitts-neuronként hivatkozhatunk rá.

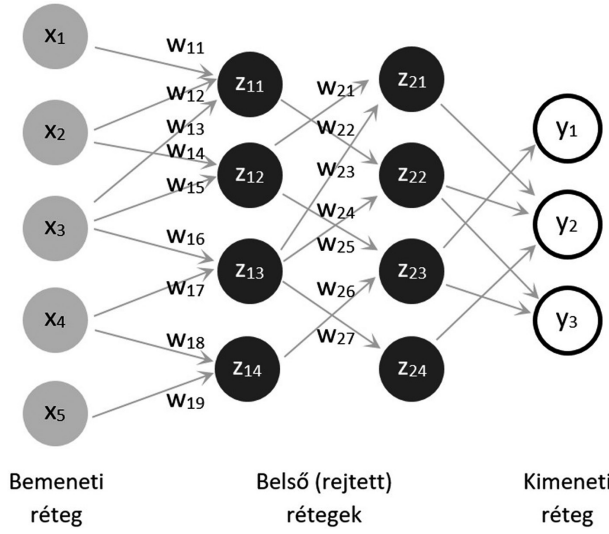
A neuron egy olyan matematikai függvény, amelyet a biológiai idegsejt mintájára alkottak meg *(1. ábra).* Ennek a függvénynek – mint „mesterséges idegsejtnek" – van valahány (*n* db) bemenő paramétere, és van egy kimenő paramétere. Az idegsejten belül nevesítünk egy ún. aktivációs függvényt. Ez a függvény határozza meg, hogy az idegsejt „továbbadja-e az ingerületet", azaz a bemeneti értékek súlyozott összege eléri-e azt a küszöböt, amelynek következtében az idegsejt „aktiválódik", valamint a függvény azt is megadja, hogy milyen értéket továbbítson a követő idegsejtek felé.



1. ábra. A természetes idegsejt felépítése és annak mesterséges leképezése

Forrás: saját szerkesztés

A *2. ábra* egy „tipikus” neurális hálót modellez. A neurális hálónak egy bemeneti, egy vagy több rejtett és egyetlen *output* rétege van. Ez az output réteg állhat egyetlen vagy több idegsejtből, attól függően, hogy milyen probléma megoldására hozzuk létre a neurális hálót.



2. ábra. A neurális háló felépítése klasszifikációs problémára (háromféle kimeneti csoportra)

Forrás: saját szerkesztés

Tipikus regressziós problémák esetén például egyetlen kimeneti idegsejtben összegződik a modell által számított eredmény. Ha klasszifikációs problémára jön létre a neurális háló, akkor a kimeneti réteg annyi idegsejtből áll, ahány kimeneti csoportról beszélünk.

A *2. ábrán* a nyilak mutatják, hogy mely idegsejtek mely idegsejtekkel állnak kapcsolatban, azaz a nyilak jelképezik a szinapszisokat. A működés egy megfelelően felépített és paraméterezett neurális háló esetén meglehetősen egyszerű: az egyes neuronokra a megelőző idegsejtekből érkező értékeket szorozzuk a nyílhoz kapcsolódó (w_{xy}) súlyokkal, és az így kapott értékeket összesítjük. Ezt a kapott értéket az aktivációs függvény átkonvertálja az idegsejt output értékévé, amely így bemenetét képezi az adott idegsejtet „követő", kapcsolódó idegsejteknek.

Ahhoz, hogy a mélytanulás működőképes legyen, a tanulási folyamat leírása is szükséges volt. A felügyelt tanulási folyamat alapját az ún. *backpropagation* eljárása képezi: az a logika, amely mentén a tanuló modell minden egyes bemenetre kiszámítja a kimenetet, és ezt összeveti az elvárt értékkel, majd igazítja a súlyokat. (Itt tehát felügyelt tanulásról van szó, ahol a tanításhoz rendelkezésre áll az elvárt kimenet mint függő változó.)

A modell által számított kimenet ($\hat{y}$) és az elvárt kimenet (y) különbsége adja a hiba értékét. A hibát – ún. költségfüggvényt – a tanító algoritmus a számítási folyamat elejére visszavetíti, és a belső paramétereket (w súlyokat) ennek ismeretében igazítja. A „backpropagation" során a cél a költségfüggvény – tehát az eltérések – minimalizálása.

A GÉPI ELŐRÖGZÍTÉSI MUNKA TÁMOGATÁSA NEURÁLIS HÁLÓK SEGÍTSÉGÉVEL

Mindig erős volt a motiváció, hogy az adatrögzítési feladatokat minél inkább a számítógépre bízzák. Az „áttörést" az képezné, ha a neurális háló „megtanulná" önállóan, autonóm módon könyvelni a bizonylatokat.

Nézzük meg, hogy ez milyen folyamat mentén képzelhető el. A könyvelés (kontírozás) folyamatában az a kihívás, hogy egy adott beérkező számláról, bizonylatról el kell dönteni, hogy melyik vagyonrészt érinti, illetve hogyan hat az eredményre. A kontírtétel többi eleme (dátum, szöveg, összeg) jellemzően jól definiálható. A megfelelő főkönyvi számla meghatározása így standard csoportosítási (klasszifikációs) problémaként jelenik meg, melynek kezelése „felügyelt" tanulási problémát képez, azaz a neurális háló „tanításához" megfelelő mennyiségű, jól kontírozott számlát kell felhasználnunk.

A neurális hálónak lesz egy bemeneti (input) rétege – amely az egyes bizonylatok adatait írja le –, valahány belső (rejtett vagy *hidden*) rétege, és egy *k* elemű kimeneti rétege, amely lényegében az egyes adatpontokhoz a modell által rendelt

főkönyvi számlát azonosítja. Három alapvető tényező szükséges ahhoz, hogy egy könyvelési folyamatot neurális hálóval támogassunk:

1. a felépített neurális háló;
2. megfelelő számítási kapacitás, figyelemmel a folyamat számításigényére;
3. megfelelő mennyiségű „adatpont" (bemeneti adat) a neurális háló tanításához.

A neurális háló megalkotása alapvetően rutinjellegű feladat. Több eszköz áll rendelkezésre ehhez. Az egyik leggyakoribb megoldás a Python nyelv scikit-learn könyvtára (URL1). Ebben az esetben a neurális háló felépítésekor csak nagyon alapvető döntéseket kell meghozni, amelyek a programon belül egy-egy paraméter beállítását jelentik: hány belső réteget hozunk létre, az hány idegsejtből álljon, milyen aktivációs függvénnyel jöjjenek létre a neuronok stb.

A számítási kapacitás meglétének azért van jelentősége, mert leginkább ennek hiánya hátráltatta a gépi tanulás elterjedését az elmúlt évtizedekben. A legtöbb számítási kapacitást a modell tanítási folyamata igényli. Erre a feladatra ma már olyan *online* eszközök állnak rendelkezésre, amelyek megfelelő erőforrást kínálnak. A tanítási folyamat a neurális háló méretétől (rejtett rétegek és szinapszisok száma), a tanítási halmaz méretétől és a tanítási ciklusok számától függően akár sok órát is igénybe vehet. A tanítási folyamat lefutását követően az egyes „új észlelések" – esetünkben új bizonylatok – letesztelése (kontírozása) másodpercek kérdése. A folyamat során a legnagyobb dilemmát a forrásadatok rendelkezésre állása képezi. Ebben a tekintetben kétféle minőségű adathalmazra támaszkodhatunk.

(1) Strukturált adatforrást veszünk igénybe, amely általában megfelelően egzakt adathalmazt eredményez, de ilyen formában a gazdasági események kisebb halmaza áll csak rendelkezésre. Erre a fajta adatforrásra jó példa az, amikor a Nemzeti Adó- és Vámhivatal (NAV) számlaadat-szolgáltatását vesszük igénybe. Ebben az esetben strukturált formában, XML-ben (Extensible Markup Language) jut el hozzánk az adathalmaz. (Sajnos a NAV számlaadat-szolgáltatás segítségével gyűjtött adatok használata az adatok minősége miatt sokszor erős kihívást jelent, részben az adathibák miatt, ami időnként a NAV számára is feladja a leckét.[1]) A másik problémát az jelenti, hogy a NAV számlaadat-szolgáltatás struktúrájában elég sok adatkör van, amely nem kötelező, így rendszerint üresen is marad, miközben szükséges lenne a tétel megfelelő azonosítására.

(2) Nem strukturált adatforrást képez a papíralapú, illetve egyéb, például PDF *(portable document format)* formátumú e-számla feldolgozása. Ilyen esetben optikai szövegfelismerés (*optical character recognition,* OCR) és annak értelmezése

[1] Czinege Attila (2019) NAV ellenőrzési szakfőigazgató előadása: *A digitalizáció alkalmazása a NAV ellenőrzések során*.

útján jutunk el a számlaadatokhoz. A megoldás előnye, hogy a bizonylati elv alapján a dokumentum mindig rendelkezésre áll. Hátránya, hogy az eredeti dokumentum minősége vagy a szkennelés torzíthatja a felismert és értelmezett adat minőségét. Ugyanakkor elkerülhetetlen ez a technológia is, hiszen sok esetben más adatforrás az eseményekről nem áll rendelkezésre, mert például külföldről érkezik a számla.

AZ ADATOK „GÉPI" FELDOLGOZÁSA: ALGORITMUS VS. NEURÁLIS HÁLÓ

Az algoritmust úgy definiálhatjuk mint előre meghatározott lépések sorozatát valamilyen kívánt eredmény elérése érdekében. Ez a gondolkodásmód azt a fajta megközelítést jelenti, amikor az üzleti probléma megoldása során a számviteli szakember pontosan definiálja, hogy az egyes adatelemek egyes értékállapotai hatására mit és hogyan kell a programnak tennie. A mesterségesintelligencia-alapú megközelítés ettől alapjaiban eltérő. A gépi tanulás során a rendelkezésre álló adathalmaz alapján maga a tanulási logika fogja megtalálni az összefüggéseket az egyes bemeneti jellemzők között, nem kell, sőt inkább a hatékonyságot rontja, ha a felhasználó ezt megpróbálja befolyásolni.

Az üzleti felhasználók az alapadatok feldolgozására alapvetően az algoritmus-alapú megközelítést tartják elfogadhatónak. Azaz ők akarják definiálni, hogy mely számlák azonosíthatók egyértelműen, és „választhatók le" a sokaságról mint jól meghatározott, kontírozható elemek. Sokszor merül fel például, hogy egy adott partnertől a vállalat mindig egyféle terméket vásárol/szolgáltatást vesz igénybe, azaz a definiált elvárás, hogy a program az adott partner számláit adott költségként könyvelje le. (Például a közüzemi szolgáltatók számlái tipikusan egyféle költségnemet keletkeztetnek.) Állításunk szerint részben éppen ez a felfogás blokkolja a neurális hálók térhódítását: az üzleti felhasználónak meg kellene tanulni „elengedni" a tanulási folyamatot, ne akarjon „kézivezérelni". Ehhez azonban paradigmaváltásra van szükség. Miben más a mélytanuláson alapuló megközelítés?

A kérdés megválaszolásához első lépésként azonosítani kell a feldolgozandó adatok szintjét. Tipikusan tételszintű bontásból célszerű kiindulni, azaz abból a legszélesebb halmazból, amelyen belül további főkönyvi szintű bontást nem várunk.[2] Azonosítani kell egy különálló adatpontot: ez a bizonylattétel lesz, amelyhez rendelkezésre állnak a kapcsolódó bizonylat és partner adatai is.

A következő lépés az adattisztítás, amikor a nyilvánvalóan érvénytelen adatokat kezelni kell: a hiányzó adatot pótolni, a helytelen/téves adatot javítani,

[2] Természetesen itt is felmerülhet további bontási igény, lásd költséghelyek/költségviselők szerinti elkülönítés, de ezzel most nem foglalkozunk.

eltávolítani, majd az adatokat át kell alakítani. A neurális háló ugyanis kizárólag numerikus adatokat fogad el, ebből adódóan az adathalmazt megfelelő konverziókkal át kell alakítani ilyen formára. A kategóriaadatokat különböző eljárásokkal át kell kódolni. Például az adathalmazban a partnerek városa (például: Budapest, Szeged, Debrecen) kategóriaadatot képez, ezt „átkódoljuk".

A kategória típusú adatok átkódolásának legegyszerűbb megoldása, ha az egyes kategóriákat besorszámozzuk. Például Budapest = 1, Szeged = 2, Debrecen = 3, és ilyenformán numerikussá alakítjuk: az eredeti kategóriaadatot az új, átkódolt numerikus értékre cseréljük ki. Ez a megoldás egyszerűsége miatt ugyan jónak tűnhet, de erős torzulást ad a modellnek, mivel a neurális háló értelmezése szerint a debreceni (3) adatpont háromszorosa lesz a budapesti (1) adatpontnak, azaz az érték valódi mennyiségként jelenik meg a modellen belül a követő feldolgozás során. Gyakran használt megoldás ehelyett, ha az egyetlen kategória oszlop helyett annyi oszlopot veszünk fel, ahányféle elem szerepel a sokaságban, és 1, 0, 0 – 0, 1, 0 stb. kódolással küszöböljük ki a kategóriaadatot. Ez a bemeneti oszlopok számát ugyan jelentősen megnöveli, de a modell szempontjából ez kevésbé jelent problémát. Végül a folyamat eredményeként előáll egy olyan adathalmaz (ún. értékvektorhalmaz), amely kizárólag numerikus adatokat tartalmaz.

A következő lépésben a vektor elemeit standardizáljuk, amivel az egyes bemeneti adatelemek várható értéke nulla, szórása pedig egy lesz. E lépés nélkül az egyes inputadatok az értéktartomány szélességéhez mérten jelentős túlsúlyt képviselnének a modellen belül, elrontva ezzel a modell használhatóságát.

A NEURÁLIS HÁLÓ TANÍTÁSI FOLYAMATA

A neurális háló tanításánál a szabványos folyamat szerint a tanításra rendelkezésre álló adathalmazt két részre bontjuk: egy tanítási halmazra (training set) és egy tesztelési halmazra (test set). Szokvány szerint a teljes halmaz tetszőleges 75–80%-át azonosítjuk tanítási halmazként, a fennmaradó 20–25% lesz a tesztelési halmaz. A tanítási halmaz elemei szolgálnak a neurális háló tényleges tanítására, azaz a tanításhoz ezt a halmazt és a halmaz bizonylatainak valós besorolását (azaz esetünkben a helyes eszköz/költség/ráfordítás számlaszámot) használjuk fel, amelyekkel a neurális háló beállítja a saját belső paramétereit – azaz a *2. ábra* jelölései szerinti w_{xy} súlyokat.

Első lépésben „inicializáljuk" a neurális hálót, azaz az egyes kapcsolatokhoz rendelt súlyokat kicsi (nullához közeli, de nullánál nagyobb) értékre állítjuk be. Következő lépésben a tanítási halmaz első elemének adatait bedobjuk a bemeneti (input) rétegbe, végigfuttatjuk a hálón, és a kimenetet összehasonlítjuk a helyes besorolással. Ha eltér, akkor a hálót arra utasítjuk, hogy korrigálja a kapcsolatok

w_{xy} súlyozását. Ugyanezt elvégezzük a tanítási halmaz minden elemére, mindig egyre korrigálva a súlyokat. A tanítási folyamatot annyiszor futtatjuk végig, ameddig a besorolási eltéréseket képező költségfüggvény megfelelő érték alá nem süllyed. Ezt a folyamatot említi tehát az angol terminológia *backpropagation* néven. Ekkor mondhatjuk azt, hogy a tanulási folyamat lezajlott. Következő lépésben a korábban leválasztott tesztelési halmaz bizonylatain kipróbáljuk a modellt, azaz megnézzük a modell szempontjából „új" bizonylatokon, hogy azokat a neurális hálónk hogyan kontírozná. Végül ezt összevetjük a tesztelési halmaz tényleges (helyes) kontírozásával. Ezzel a modellünk validációja is megtörténik: meghatározható, hogy milyen pontossággal képes az új adatpontok esetén a kontírozásra.

A folyamat tehát teljes mértékben függetleníthető az üzleti felhasználók előzetes várakozásaitól: nincs szükség „előre kitalálni", hogy mely bemeneti adatok milyen szempontból számítanak fontosnak vagy kevésbé fontosnak. Az univerzális neurális hálóba beleöntjük a tanító halmaz „megfigyeléseit" – a számlák adatait mint független változókat –, és megadjuk az egyes megfigyelésekhez kapcsolódó kimenetet, azaz a számlák eredeti, helyes kontírozását. A tanuló logika teljesen autonóm módon határozza meg, hogy a bemeneti adatok közül melyek fontosak, és milyen súlyozás szerint, mely bemeneti paraméterek lesznek érdektelenek a kimenet szempontjából.

A MODELL ÉRTÉKELÉSE

A gépi tanuláson alapuló modelltől soha nem várjuk el, hogy teljes precizitással dolgozzon, ugyanakkor egyre növekvő valószínűséggel fogja megtalálni a helyes kimenetet. A modell gyakorlati felhasználhatóságához nem is szükséges az, hogy tökéletesen működjön. Elég, ha az emberi munkaerő sztochasztikus hibázási mintázatát meg tudja haladni a modell „kiszámítható bizonyosságú" teljesítménye.

A tanulási folyamat során a használt eszköztől függően megjeleníthető a költségfüggvény értéke. A tanulási körök iteratív futtatásakor a program megmutatja, hogy még mekkora az eltérés a modell által számított érték és a tényleges érték között. Nagy a kísértés, hogy a tanítás erőltetésével a költségfüggvényt nulla körüli értékre csökkentsük a tanítási folyamat többszörös, sokszor ismételt elvégzésével, azaz így a tanulási halmaz (training set) alapján tökéletessé tehetjük a modellünket. Ez a csapda az ún. overfitting (egyfajta „túlillesztés"), amelynek során a tanulási folyamat kiterjesztésével a modell egy ponton túl már nem *általában,* az egyes bemeneti értékmezők közötti összefüggéseket tanulja meg, hanem specifikusan, a tanítási halmaz adatpontjai közötti összefüggéseket képezi le. A felhasználó ilyenkor azzal szembesül, hogy amely modell „tökéletesnek"

tűnik a tanítási halmazra, az egyre rosszabb minőségben dolgozik a teszthalmaz elemeivel, és elromlik a modell használhatósága.

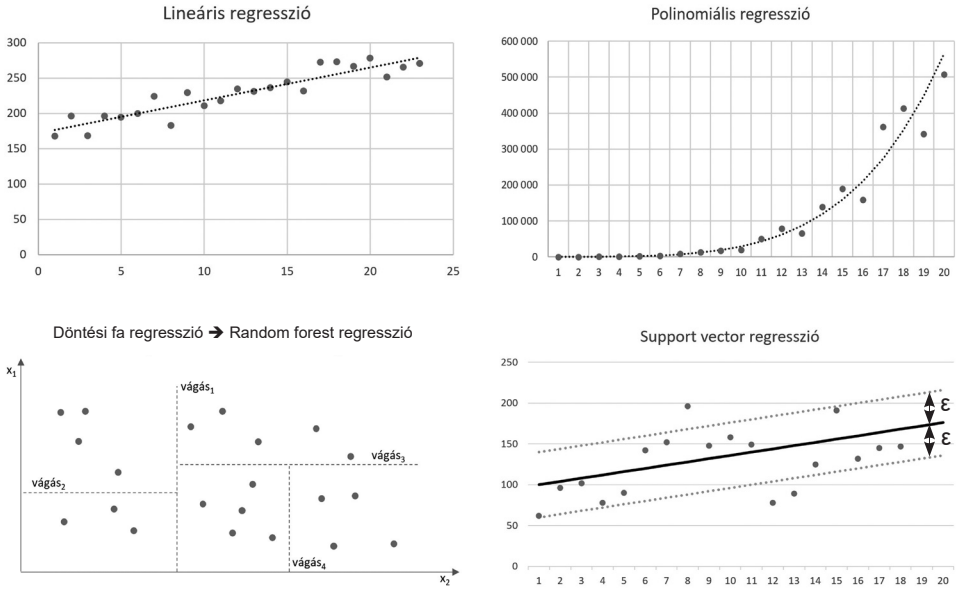
Mekkora a szükséges halmaz, amely elegendő a neurális háló tanításához? Ezt a kérdést nem lehet egzakt módon megválaszolni. A helyes válasz az lenne, hogy „minél több", de igazából sok tényező függvényeként határozható meg, attól függően, hogy hány figyelembe vehető jellemzővel dolgozunk, hány csoportot (főkönyvi számlát) definiáltunk. Ha kicsi a tanulási halmaz, akkor a költségfüggvényünk nehezen közelít a nullához. Ha a tanítási körök számát növeljük, akkor a költségfüggvény csökkenni fog, de már egyre erősebben fog jelentkezni az overfitting jelensége.

A MESTERSÉGES INTELLIGENCIA ALKALMAZÁSÁNAK TOVÁBBI CSAPDÁI

A kontírozás számviteli helyessége sokszor a bizonylati adatokon túlmenő, egyéb információhalmaztól is függ. Elég arra gondolni, hogy egy adott tétel bizonyos körülmények között költségként, más esetekben beruházásként értelmezhető. Ugyanaz az eszköz értelmezhető lehet anyagként, áruként, meglévő eszköz alkatrészeként vagy önálló beruházásként is. Mindez a bizonylat/bizonylattétel adataiból nem is vezethető le. A gépi modellektől ezért tipikusan előrögzítést várhatunk el, azaz egyelőre nem fogadjuk el, hogy teljesen autonóm módon döntsenek a számviteli besorolásról, szükségesnek tekintjük annak utóellenőrzését. Ez is azt indokolja, hogy az algoritmusalapú megoldást erősebbnek tekintjük. A számviteli besorolást valakinek valahol meg kell adnia, vagy valamely adatokból egzaktan levezethetőnek kell lennie. Ellenkező esetben, még mindig elkerülhetetlennek tűnik annak humán validációja. Ez tekinthető talán a legfontosabb gátló tényezőnek, amely a mesterséges intelligencia térhódítását akadályozza a kontírozási folyamatban.

A SZÁMVITELI BECSLÉSEK TÁMOGATÁSA

A számviteli becslések a beszámoló adatok jelentős hányadát befolyásolják. Szerepük lehet a tárgyi eszközök értékcsökkenés-paramétereinek megképzésénél, a követelések értékelésénél, a céltartalékok meghatározásánál stb. A számviteli becslések támogatása gépi tanuló algoritmusok által szintén fontos paradigmaváltással jár, és a számviteli információ pontosságát is javítani tudja. Nézzük meg három becslési probléma kezelési lehetőségeit azokon a területeken, amelyekkel a gépi tanulási modellek minél szélesebb palettáját be tudjuk mutatni. A gyakorlatban természetesen sokkal szélesebb kört érintenek a becslések.



3. ábra. Fontosabb regressziós modellek a gépi tanulásban

Forrás: saját szerkesztés

A maradványérték becslése

A maradványérték meghatározása egy tipikus becslési feladat a számvitel területén. Tárgyi eszköz aktiválásakor meghatározandó, hogy (1) várhatóan hány évig használja a vállalkozás az adott eszközt (hasznos élettartam), (2) a használati évek leteltekor várhatóan mennyit fog érni az eszköz, azaz a használat során mekkora értékig íródik le, (3) milyen módszerrel osztja szét az elhasználódás értékét a vonatkozó üzleti évek között.[3] A maradványérték meghatározása jelentősen befolyásolja az egyes időszakok értékcsökkenését, és így az eredmény alakulását is. Ha egy eszköz megfelelően kiterjedt használt piaccal rendelkezik, akkor a maradványérték meghatározásához a becslési bizonytalanság jelentősen csökkenthető regressziós modellekkel. A maradványérték becsléséhez megfelelő mintát kell gyűjteni a hasonló eszközökből, a pontos jellemzőik és piaci használtértékük megjelölésével. Az így képződött adatpontok alapján fogjuk az ML regressziós modellünket tanítani és tesztelni. Az elérhető termékek jellemzői képezik a modellünk független változóit (features), a piaci ára pedig a függő változót (y).

Döntési pontot jelent, hogy mely modellt használjuk a regresszió végrehajtására (lásd *3. ábra*). Tipikus választás lehet a lineáris regresszió, amikor lineáris kapcsolatot látunk a tulajdonságvektorok és a függő változó értéke között.

[3] 2000. évi C. törvény 52. §.

Polinomiális modellt alkalmazunk, ha exponenciális kapcsolatot tapasztalunk. Ismerünk ezen túlmenően egyéb, hatékony modelleket is, így például a döntési fa regresszióból a vágások véletlenszerű végrehajtásával származtatott ún. *random forest* regressziós modellt vagy a *support vector* regressziót. Ezeket a modelleket a támogató eszközök felhasználásával teljesen analóg eljárás szerint lehet használni: a korábban leírt tanítási halmaz és teszthalmaz szétbontását követően a tanítási halmazzal illeszthető a sokaságra a gépi modell, amelyet ezt követően a teszthalmaz elemeivel ki lehet próbálni. A kipróbálás részeként statisztikailag mérhető a modell teljesítménye, azaz hogy mennyire „sikeres" a becslés során. Erre különböző mutatószámok használatosak. (Például a Python r2 *score* pontozási eszköze.)

GARANCIÁLIS KÖTELEZETTSÉGEKRE KÉPZETT CÉLTARTALÉK

A garanciális kötelezettségekre képzett céltartalék lényege, hogy ha egy termelő vállalat adott évben értékesít valahány terméket, akkor a tárgyévi értékesítéshez kapcsolódóan a garanciális idő alatt a meghibásodások miatti helytállási kötelezettségből adódóan kiáramlás fog keletkezni. Ennek a várható értéke becslés alapján határozható meg. A problémakör specialitása, hogy az időtényezőnek jelentős, ámde erősen specifikus szerepe van, ami kezelhető lenne a korábban tárgyalt regressziós modellekkel, de ennél jobb eszköz is rendelkezésre áll. Egyrészt a neurális hálók regressziós alkalmazása, illetve egy másik, még ennél is erősebb eszköz, amelyet „rövid memóriával rendelkező" neurális hálóként, azaz RNN-ként (recurrent neural network) azonosítunk. A modellt több éven át gyűjtött tapasztalati adatok alapján tudjuk felépíteni, amelyben így a legelső évek észlelései is benne lesznek. Ezek az észlelések azonban sokkal kevésbé kell, hogy befolyásolják az aktuális év eseményeit, bár eltekinteni nem lehet tőlük. A megoldást az RNN-modell jelenti, amely az emberi agy rövid távú memóriájának mintája szerint a felejtés hatásának paraméteres beépítésével „lehalkítja" a korábbi időszakok hatását.

VEVŐKÖVETELÉSEK ÉRTÉKELÉSE

A gyakorlatban a szabályozási környezet a számításokat a költség-haszon összevetés elvének megfelelően, bizonyos egyszerűsítésekkel engedi elvégezni. Ilyen egyszerűsítést jelent a vevők csoportos értékelése, amely szerint – bár főszabály szerint a vevőket egyedileg értékeljük a rendelkezésre álló információk alapján – bizonyos „kisösszegű követelések" csoportosan is értékelhetők. A gépi tanulás felhasználásával az egyszerűsítésre nincs szükség, ami jelentősen pontosabbá teheti a számított értékvesztés értékét.

A csődelőrejelzés mutatószámaival elég sok tanulmány foglalkozik, így jelen értekezés keretében a kapcsolódó gépi tanulási logikát vázoljuk csak fel. A csődelőrejelzés a gépi tanulás keretében többféle szemléletben kezelhető. A vevőkövetelés értékeléséhez a csődvalószínűség meghatározása szükséges. A becslés pontosságának javítására lehetőség van a korábban vázolt sokféle eszköz kombinálására is. Ezt hívja a szakirodalom „ensemble" metódusnak, azaz amikor többféle eszköz egymásra épülésével állítjuk össze a modellt. Ha a csődvalószínűség minden partnerre rendelkezésre áll, akkor a kitettséggel (nyitott vevőköveteléssel) szorozva megkapjuk a veszteség várható értékét, amely alapján az értékvesztés megképezhető.

A SZÁMVITELI AUDIT TÁMOGATÁSA

A számviteli beszámolók, pénzügyi kimutatások könyvvizsgálata során is szerephez juthatnak a becsléseknél alkalmazott megoldások a becslések helyességének megállapításakor, amikor a könyvvizsgáló újraszámítással szeretné ellenőrizni a becslés észszerűségét. Az auditmunkában a „nem felügyelt" tanulás is szerephez juthat. Helyes megközelítésként a modellbe beleönthetjük az észleléseket, és azt várjuk a modelltől, hogy azonosítsa a sokaságból minél jobban „kilógó" elemeket. Ezek az ún. *outlier*ek, amelyek lehetnek akár teljesen helyes elemek is, de lehetnek adathalmazon belüli hibák (ún. anomáliák), amelyek akár helytelen rögzítés, akár csalás következményeképpen is bekerülhetnek az adathalmazba. Az érték szerinti outlierek egyébként is jelentős szerepet kapnak a vizsgálat során. A könyvvizsgáló az érték szerint „kilógó" elemeket a módszertan szerint leválasztja, és tételesen vizsgálja. A gépi modelltől azonban ennél többet várunk. Széles körben alkalmazott technika a „gyanús" tételek kiválasztására a Benford-törvény, amely az egyes tételösszegek kezdő számjegye(i) alapján jelöli ki a normális eloszlásból kilógó, tehát a gyanú szerint mesterségesen kreált tételösszegeket.

A mélytanulás (deep learning) világában tudunk olyan modellt építeni, amely az outlierek meghatározását *n* dimenzió mentén, az adathalmazon belül feltérképezett mintázat segítségével hajtja végre. Az *n* dimenziós outlier-elemzésre megfelelően erős eszközt kínál a gépi tanulás egyik típusa, az önszerveződő térkép (self organized map, SOM), melynek alapváltozatát Teuvo Kohonen alkotta meg, mintegy negyven évvel ezelőtt. Ez nem felügyelt (unsupervised) tanulást jelent, azaz nem tudjuk előre megmondani a sokasági elemekről, hogy outliernek számítanak-e, illetve milyen módon csoportosíthatók. (Ez lenne a sokaság függő változója.) Ami ilyenkor a rendelkezésre áll, az egy *n* elemű bemeneti halmaz, amely a bizonylatokat tartalmazza. Minden bizonylat *m* jellemzőből áll. Ez az *m* jellemző (gazdasági esemény időpontja, fizetési mód, rögzítő, rögzítés időpontja stb.) adja a modell független változóit. A gépi tanulás során az önszerveződő térkép segít-

ségével meg tudjuk találni az adathalmazon belüli mintázatot. Ezenfelül azt is várjuk, hogy az egyértelműen „kilógó" elemek láthatóvá váljanak.

Tegyük fel, hogy rendelkezésünkre áll ezer darab bizonylat, és minden bizonylatnak tíz jellemzőjét azonosítjuk, amelyek numerikussá alakíthatók. A numerikus jellemzőket jelölje: x_1, x_2, … x_{10}. Bár a neurális hálóval ellentétben ez a modell nem igényli a sztenderdizálást, a számítási kapacitás kímélése miatt szokás legalább egy normalizálást végrehajtani. (Azaz legalább a 0–1 intervallumon belülre szorítani az egyes jellemzők értékét.) Ezt követően tekintünk egy $n \times n$-es rácsot, amelyen a bizonylatok mintázatának lenyomatát szeretnénk megkapni. Ez esetünkben lehet például 40 × 40-es. Ez annyit tesz, hogy adott 40 * 40 = 1600 pontunk. Minden pontnak ugyanannyi jellemzője lesz, mint az egyes bizonylatoknak. Egy d. rácspont jellemzői: u_{d1}, u_{d2}, … u_{d10}. A következő lépésben „inicializáljuk" a rácsot, azaz az 1600 ponthoz rendelünk egy-egy véletlen vektort, azaz minden pontnak véletlenszerűen feltöltjük a tíz bizonylati jellemzőjét. Ezután minden bizonylathoz megkeressük azt az elemet a rácsból, amely a tíz jellemzőjével a legközelebb áll az adott bizonylathoz. Ezt egyszerűen, az ún. euklideszi távolság meghatározásával tesszük meg. Ezt a „legközelebbi" rácselemet „legjobban egyező elemként" azonosítjuk (best matching unit, BMU). Az euklideszi távolság számítása a *d.* rácspontra vonatkozóan:

$$D = \sqrt{(x_1 - u_{d1})^2 + (x_2 - u_{d2})^2 + \cdots + (x_{10} - u_{d10})^2}$$

Ennek a BMU-rácselemnek a jellemzőit egy adott paraméterrel „közelebb húzzuk" a bizonylati jellemzőhöz, vagyis végzünk a rácselem jellemzőiben egy igazítást. Ugyanezt tesszük a BMU adott *r* sugár szerinti környezetében is úgy, hogy minél közelebb áll egy elem a BMU-hoz, annál erősebben igazodjon a bizonylati jellemzőkhöz, majd továbblépünk a következő bizonylatra. Miután a teljes bizonylati halmazra, azaz – a példa szerinti – ezer bizonylatra végrehajtottuk a fenti igazítást, azután indítunk *t* darab új kört, körönként lecsökkentve az *r* sugár értékét, azaz ezt követően a BMU-elem egyre szűkebb környezetét fogják érinteni a korrekciók. Ezzel az eljárással a bizonylathalmaz belső mintázata „lenyomatot képez" a rácson. A „hasonló" bizonylatcsoportok egy-egy egymás melletti rácselemet fognak „lefedni". Ezzel lényegében egy klaszterezést hajtottunk végre. Ezt követően az egyes rácselemekhez meghatározható, hogy mekkora távolságra vannak a szomszédos elemektől. Ha egy rácselem távolsága a környező elemektől nagy, akkor abból következik, hogy egyik „jellemző csoporthoz" sem tartozik. Ez és a hasonlóan „kilógó" rácselemek lesznek az outlierek. Következő lépés annak a meghatározása, hogy mely bizonylatoknak voltak az outlier rácselemek a BMU-i.

Így a bizonylati halmazból kiemelhetővé váltak a mintázatokból kilógó elemek. (Ha normalizált értékekkel dolgoztunk, akkor a normalizálás visszafordításával tudunk hozzájutni az eredeti bizonylatjellemzőkhöz.) Az önszerveződő

térkép ilyen módon felfogható egy radikális dimenziócsökkentési eljárásnak is, amelynek során egy akárhány dimenziós halmaz belső összefüggéseit kétdimenziós rácsra képezzük le.

IRODALOM

Amidi, Afshine – Amidi, Shervine (n. d.): *Recurrent Neural Networks Cheatsheet.* https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Bauer Péter – Endrész Mariann (2016): Modelling Bankruptcy Using Hungarian Firm-Level Data. (*MNB Occasional Papers* 122) Budapest: Magyar Nemzeti Bank, https://www.mnb.hu/letoltes/mnb-op-122-final.pdf

Czinege Attila (2019): Czinege Attila NAV ellenőrzési szakfőigazgató előadása: *A digitalizáció alkalmazása a NAV ellenőrzések során*. In: MKVK XXVII. Országos Könyvvizsgálói Konferencia, 2019. 09. 26.

Fisher, James (n. d.): What Is a McCulloch-Pitts Neuron? (Facts on File Math Library) https://jameshfisher.com/2019/05/27/what-is-a-mcculloch-pitts-neuron/

Kohonen, Teuvo (1982): Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics*, 43, 59–69. DOI: 10.1007/BF00337288, https://www.cnbc.cmu.edu/~tai/nc19journalclubs/Kohonen1982_Article_Self-organizedFormationOfTopol.pdf

Kumar, Yashwant (2018): Credit Card Fraud Detection using Self Organizing FeatureMaps. *Medium,* 17 Sep 2018. https://medium.com/@yashwant2451/credit-card-fraud-detection-using-self-organizing-featuremaps-f6e8bca707bd

McCulloch, Warren S. – Pitts, Walter H. (1943): A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics,* 5, 115–133. Reprint: *Bulletin of Mathematical Biology,* 1990. 52, 1–2, 99–115. https://www.cs.cmu.edu/~./epxing/Class/10715/reading/McCulloch.and.Pitts.pdf

Olszewski, Dominik (2014): Fraud Detection Using Self-Organizing Map Visualizing the User Profiles. *Knowledge-Based Systems,* 70, 324–334. DOI: 10.1016/j.knosys.2014.07.008, https://www.researchgate.net/publication/266320924_Fraud_detection_using_self-organizing_map_visualizing_the_user_profiles

Pimbley, Joe M. (2014): *Benford's Law as a Logarithmic Transformation.* http://www.maxwell-consulting.com/Benford_Logarithmic_Transformation.pdf

Qu, Yi – Quan, Pei – Lei, Minglong et al. (2019): Review of Bankruptcy Prediction Using Machine Learning and Deep Learning Techniques. 7th International Conference on Information Technology and Quantitative Management. *Procedia Computer Science,* 162, 4, 895–899. DOI: 10.1016/j.procs.2019.12.065, https://tinyurl.com/yj6a483j

Rumelhart, David E. – Hinton, Geoffrey E. – Williams, Ronald J. (1986): Learning Representations by Back-Propagating Errors. *Nature,* 323, 533–536. DOI: 10.1038/323533a0, https://www.cs.utoronto.ca/~hinton/absps/naturebp.pdf

Tabak, John (2014): *Geometry: The Language of Space and Form*. Infobase Publishing

URL1: scikit-learn. *Machine Learning in Python*. https://scikit-learn.org/stable/