ÁLTALÁNOS CÉLÚ NEURÁLIS HÁLÓZAT TESZTELÉSE KÜLÖNBÖZŐ ADATBÁZISOKON

Benkő-Kiss Árpád – Fabulya Zoltán – Hampel György

Absztrakt: Két rejtett réteget tartalmazó, előrecsatolt, teljes összeköttetésű neurális hálózatot teszteltünk több adatbázison. A teszt célja, hogy az adott rendszer milyen hibaszázalékkal képes az input adatokból a várható értékeket előre jelezni, és alkalmas-e gyakorlati feladatok előrejelzésére. A Python scriptnyelven fejlesztett grafikus felülettel rendelkező rendszer rugalmasan paraméterezhető, mind a hálózat mérete, struktúrája változtatható, tehát adaptálható különböző méretű adatbázisok vizsgálatára.

Abstract: We tested a feedforward, fully connected neural network with two hidden layers on several databases. The purpose of the test is to determine with what percentage of errors the given system is able to predict the expected values from the input data, and whether it is suitable for predicting practical tasks. The system with a graphical interface developed in the Python script language can be flexibly parameterized, and the size and structure of the network can be changed, so it can be adapted to the examination of databases of different sizes.

Kulcsszavak: neurális hálózat, lineáris algebra, back-propagation, Python, adatelemzés, aktivációs függvények

Keywords: neural network, linear algebra, back-propagation, Python, data analysis, activation functions

1. Bevezetés

A számítógépes alkalmazások között a mesterséges neurális hálózatok (Neural Network, röviden NN) számos válfaja használatos, illetve a jövőben egyre fontosabbá válik (Zsótér, 2007, Zsótér–Kaliczka, 2014, Zsótér, 2017) a gazdaság szinte minden területén. Természetesen más-más neurális hálózat típus és algoritmusok használatosak a kép és hangfeldolgozásban, és mások az egyszerű adatbányászatban vagy összefüggés keresésben, adatelemzésekben. Ahogy a társadalmi és gazdasági hálózatok is az egyes aktorok, illetve entitások közötti kapcsolatok és kölcsönhatások révén épülnek fel (Kis, 2006; Kis, 2013), úgy a neurális hálózatok is az elemek, neuronok összekapcsolt rendszeréből állnak.

Egy egyszerű, de rugalmasan paraméterezhető, előrecsatolt, egynél több rejtett réteggel rendelkező neurális hálózat viszonylag sok feladatot képes jó hatásfokkal megoldani.

A legegyszerűbb összefüggés-vizsgálatok (pl. kétváltozós regressziók, korrelációk) megoldáshoz megvannak a matematikai módszerek, ilyen esetben nem szükséges a NN használata, és az összefüggés jól ábrázolható grafikusan is.

A három, vagy annál több változót tartalmazó esetben a vizuális ábrázolás egyszerre csak minimum 3 dimenzióban lehetséges. A gyakorlatban azonban egy objektumnak/mintának gyakran háromnál több paramétere, attribútuma, jellemzője lehet. Annak eldöntésére, hogy az egyes jellemzők milyen mértékben és arányban határozzák meg az elvárt kimenetet, bonyolult és hosszadalmas számításokat kell használni.

ISSN 1788-7593 (Nyomtatott) DOI: https://doi.org/10.14232/jtgf.2022.3-4.37-46
ISSN 2676-9867 (Online)

A Neurális hálózatok a többváltozós, komplexebb adatbázisok elemzéséhez nyújthatnak segítséget, ahol a változók nagyobb száma miatt problémásabb lehet a matematikai módszerek használata.

A neurális hálózatok működésének matematikai alapjait a lineáris algebra szabályai és módszerei alkalmazásával lehet kialakítani, mátrixműveletek segítségével (Scharnitzky, 1993).

A hálózat által kiszámított eredményt összevetjük az elvárt eredménnyel, és ha az eltérés nagyobb az elvártnál, akkor a hiba visszaterjesztés (backpropagation) alkalmazásával, több száz vagy több ezer iterációs lépésben finomítjuk a hálózat elemei közötti kapcsolatokat a megfelelő eredmény eléréséig. Tehát jó esetben a hálózat megtanulja, hogy az adott inputokhoz milyen outputnak kell tartoznia.

A mátrixszorzások esetén a kimenetet szabályozni kell (egy neuron vagy tüzel, vagy nem), tehát ha egy neuron a rejtett rétegben pl. 10 másiktól kap inputot, akkor a 10 neuron által küldött jelek összeadódnak. A kapott értéket egy aktivációs függvény 0 és 1 közé szabályozza, és ez az érték halad tovább a következő réteghez. Az aktivációs függvény kimenete lehet 0 vagy 1, illetve a kettő érték közötti érték az aktivációs függvénytől függően (forward propagation).

A folyamat végén az output réteg neuronjának értékét és a várható célértéket kell összehasonlítani, majd a hibát (ha az egy elfogadható szint felett van) visszaterjeszteni egy iterációs eljárás során (backpropagation). A folyamat matematikai leírása számos Neurális Hálózatokkal foglalkozó forrásmunkában megtalálható (Chollet, 2021; Fazekas, 2013; Freedman, 2019; Raschka, 2016; Tóth, 2017).

2. Anyag és módszer

2.1. Neurális hálózat

A biológiában egy neuron a beérkező adatokat több száz, vagy több ezer *dendrit*-en keresztül fogadja, majd ezektől függő passzív vagy aktív állapotba kerül. Állapotát az *axon*-on továbbítja, általában egy igen vagy nem jelként (0/1). Az igen jelenti azt, hogy a neuron *tüzel*, tehát impulzust továbbít. Ha a bemenő adatokból származó inputok összege egy bizonyos limitet nem ér el, a neuron nem továbbít impulzust. A mesterséges neuronok lényegében ezt a folyamatot szimulálják, és így többféle neuronhálózat építhető fel.

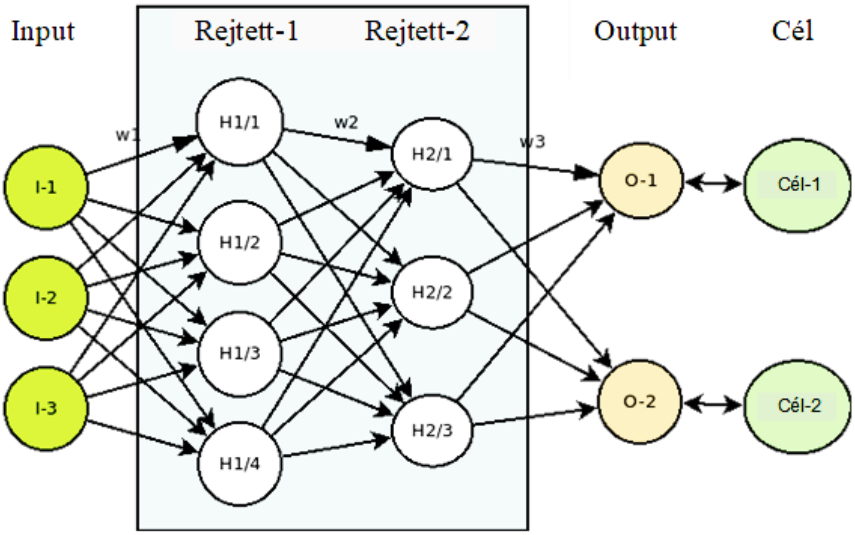
A Neurális hálózat szerkezete lényegében egy irányított, súlyozott gráf (*1. ábra*), melynek csúcspontjai a rétegekbe rendeződött neuronok, míg a rétegek közötti összeköttetés (a csúcspontok közötti élek) mátrixokkal valósítható meg (Altrichter et al., 2007).

A hálózat rétegei:

- Input réteg: ez felel meg az érzékelő neuronoknak, ami egy egyed vagy objektumból származó adatokat fogad. Objektumonként minimum két attribútumnak (adatnak) már van értelme, de a hálózat igazi ereje akkor mutatkozik meg, amikor egy egyed több, akár több tucat adatából kell eredményt meghatározni.

- Output réteg: ez egy vagy több kimenetű adatot jelent. Ez lesz összehasonlítva az elvárt kimenettel, és minél kisebb a különbség, annál pontosabban jósol a hálózat.
- Rejtett rétegek: az input és az output rétegek között helyezkednek el.

1. ábra: **Neurális hálózat két rejtett réteggel**



Forrás: saját kutatás adatai alapján a szerzők szerkesztése.

2.2. A neurális hálózat matematikai modellje

Az egyes rétegek neuronjai kapcsolatának szorosságát mátrixok tárolják, mint súly értékeket. Így egy réteg neuronjaiban tárolt értékekből a mátrix szorzás műveletével kapjuk meg a kapcsolódó réteg neuronjaiban tárolandó értékeket. A neurális hálózat egy rétege egy vektor, melynek minden eleme egy neuron értékét (állapotát) tárolja. A súlyokat tároló mátrix ($A = [w_{ij}]$) n sorból és m oszlopból áll, amikor a hálózatban egy n elemű réteget ($\bar{x} = [x_i]$) egy m elemű réteg ($\bar{y} = [y_j]$) követ. A számítás az (1) képlet szerint történik.

$$\bar{y}^T = \bar{x}^T \cdot A \quad \Rightarrow \quad y_j = \sum_{i=1}^{n} x_i \cdot w_{ij} \quad (j = 1, 2, \dots, m) \tag{1}$$

ahol:

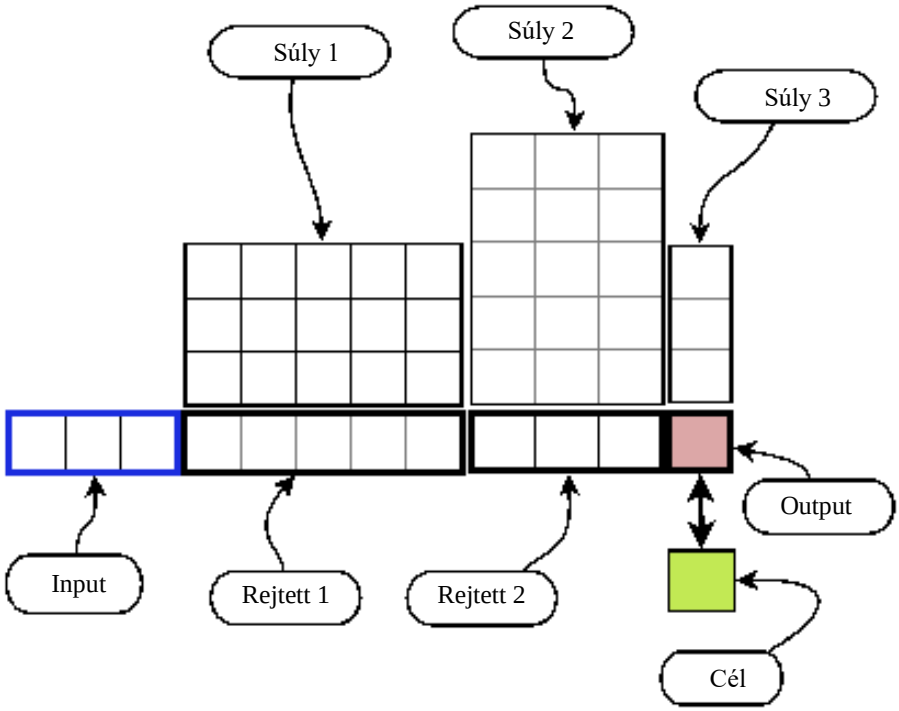
$\bar{x} = [x_1\ x_2\ \dots\ x_n]^T$ és $\bar{y} = [y_1\ y_2\ \dots\ y_m]^T$ a szomszédos rétegek vektorai

$A = \begin{bmatrix} w_{11} & \cdots & w_{1m} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nm} \end{bmatrix}$ a kapcsolat súlyainak mátrixa

A képlet alapján egy réteg ($\bar{y}$) egy neuronjának értékét (y_j) úgy kapjuk meg, hogy az előző réteg ($\bar{x}$) neuronjainak súlyozott összegét képezzük. Amikor az input és az output rétegek között rejtett rétegek is vannak, akkor nem elegendő egyetlen

mátrix a számításokhoz, mivel minden további réteg miatt újabb mátrixok szükségesek. Ekkor a rétegek közötti mátrixokkal szorzások sorozata adja az eredményt az output rétegben. Az ábrán (*2. ábra*) jól értelmezhetjük a számítás menetét. Az egymást követő rétegeknek megfelelő sorvektorok felett látható az a mátrix, mely a réteg neuronjainak értékét kiszámítja az előző réteggel vett szorzat eredményeként.

2. ábra: **Neurális hálózat rétegei sorvektorokként a súlymátrixokkal**



Forrás: saját kutatás adatai alapján a szerzők szerkesztése.

Több rejtett réteg esetén a számítás az első rejtett réteg kiszámításával kezdődik (2), majd az egymás utáni rejtett rétegekkel folytatódik (3). Végül az output réteg határozható meg az utolsó rejtett rétegből (4). Amikor a neuron hálózat tanulási folyamata befejeződik, akkor a rétegek közötti súlymátrixok szorzataként egyetlen mátrix adódik (5), mellyel közvetlenül kiszámítható az output réteg vektorának értéke az input rétegből (6).

$$\bar{h}_1{}^T = \bar{x}^T \cdot A_0 \tag{2}$$

$$\bar{h}_{i+1}{}^T = \bar{h}_i{}^T \cdot A_i \qquad (i = 1, 2, \dots, k-1) \tag{3}$$

$$\bar{y}^T = \bar{h}_k{}^T \cdot A_k \tag{4}$$

$$A = A_0 \cdot A_1 \cdot \dots \cdot A_k \tag{5}$$

$$\bar{y}^T = \bar{x}^T \cdot A \tag{6}$$

ahol:

$\bar{x}$ az input réteg vektora

$\bar{h}_i$ a rejtett rétegek vektorai $(i = 1, \dots, k)$
$\bar{y}$ az output réteg vektora
k a rejtett rétegek száma
A_i a rétegek közötti súlymátrix $(i = 0, \dots, k)$
A az input és output rétegek közötti mátrix

2.3. A Python scriptnyelv

A kísérleti adatfeldolgozó, bemutató, illetve oktatási célú szoftverhez a Python scriptnyelvet, valamint a hozzá tartozó modulokat használtuk. Ezek segítségével a megfelelő mátrixműveletek elvégezhetők, a grafikus megjelenítés könnyű, és számos forrás áll rendelkezésre (Smola–Vishwanathan,, 2008; Chen, 2016; Fazekas, 2013).

Ugyanakkor az alkalmazás kidolgozásához nem használtuk a kifejezetten Machine Learning-re kifejlesztett modulokat (SciPy, Keras, Tensorflow stb.), csak a Numpy és a Matplotlib modulokat, ami a lineáris algebrai és a mátrixműveletek elvégzése, valamint a grafikus megjelenítés miatt szükséges. A könnyebb kezelhetőség érdekében a szoftvert grafikus környezettel, menürendszerrel láttuk el, ezért jól áttekinthető, kezelhető, rugalmasan beállítható. Emiatt gyorsan adaptálható az adatbázis paramétereihez.

A vizsgált adatbázisokból 20 elemű mintát elemeztünk, majd a legjobb eredményt adó beállítással 10-szer ismételtük (epoch) a futtatást. Ezt követően az adatbázis másik 20 elemén is lefuttattuk a programot.

2.4. Adatbázisok

A bemeneti adatbázisok (*1. táblázat*) két forrásból származnak. Az egyik csoport az interneten meglévő, valós adatokat tartalmazó UC Irvine Machine Learning Repository (Dua–Graff, 1987), a másik csoport saját, generált adatokat tartalmaz.

1. táblázat: **Adatbázisok és jellemzőik**

Adatbázis	Rekordok száma	Adatok száma rekordomként	Elvárt kimenet	Forrás	Eredet
Iris	150	4	3 faj közé besorolás	Külső	Valós
Wines	4899+1599	11	0-10 közötti érték	Külső	Valós
Cancer	569	32	Igen/Nem	Külső	Valós
Klaszter	80	3	4 csoport közötti választás	Saját	Generált
Adat-4	1000	7	egyedi értékre illesztés	Saját	Generált

Forrás: saját kutatás adatai alapján a szerzők szerkesztése.

2.4.1. Az adatbázisok jellemzői:

Iris: Ez az egyik legrégebbi és kedvenc többváltozós adatbázis, 1988-tól használják neurális hálózatok tesztelésére. Lényegében három Iris faj (Iris Setosa, Iris Versicolour, Iris Virginica) csészelevél és sziromlevél hosszúság és szélesség méretadatai az inputok, az elvárt output pedig maga a faj meghatározása. Az adatbázis 150 egyed adatait tartalmazza.

Wine: Bor minősítési adatok. A 2009-es Portugál „Vino verde" fehér és vörösborok 11 jellemző adatát tartalmazza, melyek a borok kémiai összetevői, outputként pedig a bor szubjektív minőségi osztályozását tartalmazza egy likert-skála adataként, 0-10 pont között. Hatalmas adatbázis, 4899 fehérborminta, és 1599 vörösborminta adatait lehet elemezni. A jelenlegi munkában a fehérbor adatait használtuk fel.

Cancer: Emlődaganat adatok: 1995-ös Wisconsini emlődaganat adatbázis, mely 596 páciens 32 adatát tartalmazza. Az elvárt output az utolsó, 33. adat, ami a diagnózis maga, mely szerint a 0=Negatív, 1=Pozitív eredményt jelez. Ennél az adatbázisnál a kulcsfontosságú adatok is meghatározhatók, tehát akár egyesével valamely tetszőleges adat és az eredmény közötti összefüggés is tesztelhető.

Klaszter: A vizsgálathoz 4 db 20-20 elemből álló 3D klasztert hoztunk létre. Minden elem egy 3D vektor, aminek az értékeit bizonyos értéksávban véletlen szám generátorral (LibreOffice-Calc 'RANDBETWEEN' függvénye) állítottuk elő, megcímkéztük, majd a vektorcsoportokat összekevertük. Klaszter címkeként az elvárt -a csoportot jellemző- értéknek a klaszterben szereplő számok átlagát adtuk meg. 80 db 3 elemű vektort kellett 4 klaszterbe szervezni a Neurális Hálózatnak. Ez lényegében csoportosítási feladat, melyben azt kell 'megjósolni', hogy egy adott 3D vektor melyik klaszterbe sorolható a legnagyobb valószínűséggel, tehát 4 lehetséges válasz egyikét eredményezi a hálózat.

Adat-4: Generált adatbázis, egyedenként 7 input és egy elvárt output adattal. A matematikai összefüggést a neurális hálózat természetesen nem tudja megadni, de azt, hogy a bemeneti adatok és a kimenet között kapcsolat van-e, azt igen.

3. Eredmények és értékelésük

Neurális hálózat tanítására és használatára alkalmas programot fejlesztettünk Python scriptnyelven.

3.1. A kísérleti hálózat jellemzői

A program által használt adatbázis rekordjai tetszőleges számú input adatot (neuronok száma) tartalmazhatnak. Paraméterként megadható a feldolgozandó rekordok száma. Általában tanítópéldának 15-50 rekord már megfelelő, és rekordonként minimum egy adatpár (input és elvárt eredmény) a követelmény. A rekord utolsó adata az elvárt eredmény.

A hálózat 2 rejtett réteget tartalmaz melyek mérete szintén szabadon beállítható (javasolt a rétegenként 2-10 neuron). A programban minden réteghez 5 aktivációs

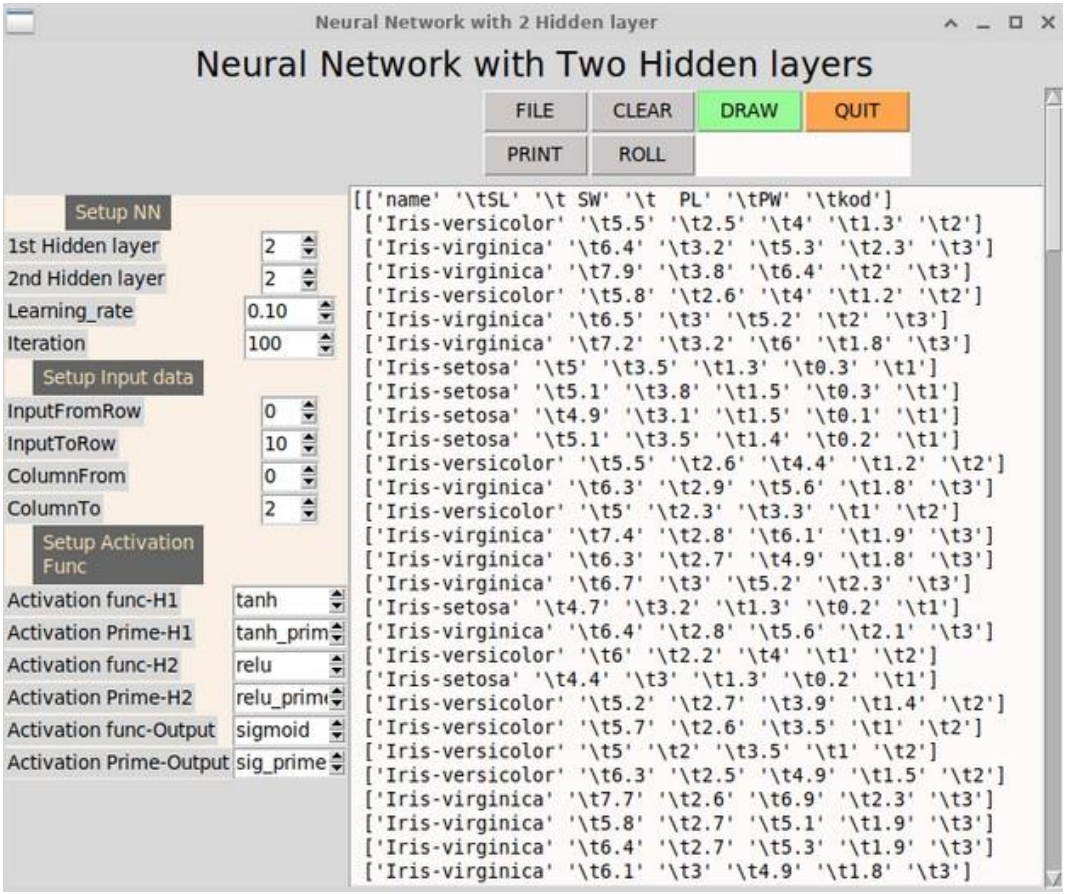
függvényből (TanH, Sigmoid, eLU, ReLU, Leaky-Relu) és azok deriváltjaiból lehet szabadon kiválasztani a megfelelőt (Fazekas, 2013; Chen, 2016).

A bemeneti adatbázis 'csv' formátumú, melyben az adatokat pontosvesszővel kell elválasztani, tizedes tört esetében vesszőt kell használni. A bemeneti file tartalmazhat szöveget is az első oszlopban illetve sorban, de a számításban ezt az oszlopot illetve sort az alkalmazás kihagyja.

Az elvárt és a számított kimenetet, illetve az iterációhoz tartozó átlagos hibát a program grafikonon jelzi. A súlymátrixok, illetve az eredmények adatai a tanítási folyamat után szövegesen is kimenthetők, így a hálózat struktúrája és értékei a későbbiekben felhasználhatók.

A fejlesztett alkalmazás felhasználói felülete (*3. ábra*) biztosítja, hogy a jellemzők tág határok között beállíthatók legyenek.

3. ábra: **Az alkalmazás felhasználói felülete, a jellemzők beállítása**



Forrás: saját kutatás adatai alapján a szerzők szerkesztése.

3.2. A program futtatásának eredményei

Minden futtatás után az alkalmazás kimenti a Neurális hálózati környezetet, valamint a kapott eredmények alapstatisztikai eredményeit (átlagos eltérés, szórás, és variancia). Mivel mind a bemeneti, mind az eredményadatok 0 és 1 közé

normalizáltak, a statisztikai adatok százalékban értendők. A *2. táblázatban* adatbázisonként olyan futtatások beállításai és eredményei láthatók, melyek a legjobbnak bizonyultak.

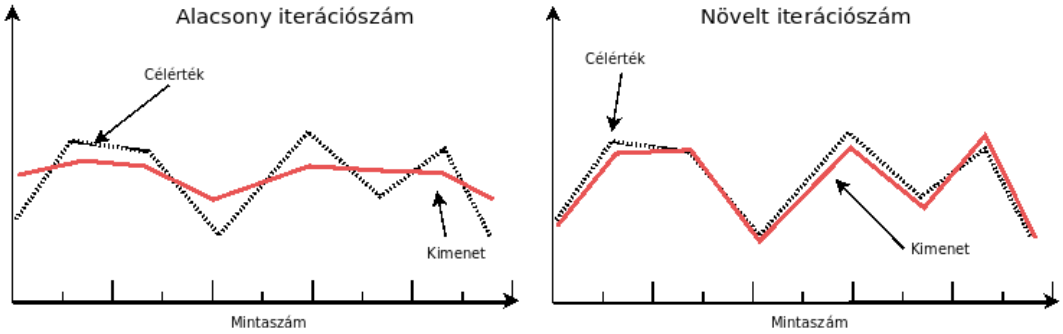
2. táblázat: **A legjobb kimenetet biztosító beállítás adatbázisonként**

Adatszett	Iris	Wine	Cancer	Klaszter	Adat-4
Tanítóminták	Minden dataszett 20-20 egyede				
Futtatások	Az alapvető beállítások után 10-10 futtatás (epoch)				
Legjobb outputok					
Input Neuron (db)	4	10	31	3	6
Rejtett réteg_1	4	3	10	3	3
Rejtett réteg_2	5	3	10	4	4
Iteráció száma	1099	1499	2999	2199	599
Learning rate	0,1	0,1	0.01	0,03	0,1
Akt függvény_1	tanh	tanh	tanh	tanh	tanh
Akt függvény_2	relu	relu	tanh	l_relu	relu
Akt függvény_3	tanh	tanh	tanh	l_relu	tanh
Hibák átlaga (abs)	4,05	3,85	3,22	2,44	1,76
Hibák szórása	5,45	4,43	5,81	2,90	2,44

Forrás: saját kutatás adatai alapján a szerzők szerkesztése.

A hálózat a legtöbb esetben néhány iteráció alatt megtalálja az átlagos értéket (átlagos hiba 0 közeli), de nem erre van szükség, hanem az egyedi értékek kiszámítására. Ez pedig nagyobb iterációszámot igényel. Ezért nem a hibák átlaga, hanem azok szórása és varianciájának értéke a mérvadó, ha nincs grafikus megjelenítése az elvárt és a számított output közötti egyedi eltéréseknek. A *4. ábrán* jól látható, hogy növelt iterációszám esetén a neuronhálózat kimeneti értéke jobban megközelíti a célértéket.

4. ábra: **Az iterációszám változásának hatása a pontosságra**



Forrás: saját kutatás adatai alapján a szerzők szerkesztése.

A hibák esetében a különbségek abszolút értékeit átlagoltuk, hogy a negatív és a pozitív eltérések ne egyenlíthessék ki egymást.

4. Következtetések

A vizsgált adatbázisok Neurális hálózattal történő vizsgálata azt mutatta, hogy a hálózat jó beállítások esetén jó eredményeket hozott a természetes és a mesterségesen generált adatbázisok esetében is.

A viszonylag sok beállítási lehetőség közül, alapvetően a magasabb iterációszámmal, és csökkentett iterációs lépésközzel (learning_rate) lehetett javítani a kiinduló eredményeket, de a rejtett neuron rétegek mérete, illetve a második és a harmadik rejtett réteg aktivációs függvényeinek módosításaival is javítani lehet az eredményt.

Irodalomjegyzék

Altrichter M., Horváth G., Pataki B., Strausz Gy., Takács G., Valyon J. (2007): *Neurális hálózatok.* Panem Könyvkiadó. Budapest.

Chen, G. (2016): A gentle tutorial of recurrent neural network with error backpropagation. *arXiv preprint* arXiv:1610.02583. https://doi.org/10.48550/arXiv.1610.02583

Chollet, F. (2021): Deep learning with Python. Simon and Schuster. <https://www.simonandschuster.com/books/Deep-Learning-with-Python/Francois-Chollet/9781617294433> (2022.10.10.)

Dua, D., Graff, C. (2019): UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. <http://archive.ics.uci.edu/ml> (2022.11.20)

Fazekas I. (2013): *Neurális hálózatok*. Debreceni Egyetem Informatikai Kar. Debrecen.

Kis K. (2006): A társadalmi tőke, mint a társadalmi és gazdasági folyamatokat befolyásoló erőforrás. *Agrártudományi Közlemények (Acta Agraria Debreceniensis)*, (20): 69–73. https://doi.org/10.34101/actaagrar/20/3157

Kis K. (2013): Vidékgazdaság, erőforrások, infrastruktúra rendszerszemléletben. In: Veres L. (szerk.): *Regionális földrajzi tanulmányok: Abonyiné Dr. Palotás Jolán 70. születésnapja tiszteletére*. Egyesület Közép-Európa Kutatására, Szeged. 109–120.

Freedman, R. S. (2019): Visual Backpropagation. *arXiv preprint* arXiv:1906.04011. https://doi.org/10.48550/arXiv.1906.04011

Raschka, S., Mirjalili, V. (2019): Python Machine Learning - Third Edition. Packt Publishing. < https://github.com/packtpublishing/python-machine-learning> (2022.10.05.)

Scharnitzky V. (1993): *Vektorgeometria és lineáris algebra*. Nemzeti Tankönyvkiadó. Budapest.

Smola, A., Vishwanathan, S. V. N. (2008): Introduction to machine learning. Cambridge University Press. UK.

Summerfield, M. (2009): *Python 3 programozás - Átfogó bevezetés a Python nyelvbe*. Kiskapu Kiadó.

Tóth L., Drósz T. (2017): Mesterséges Neuronhálók és alkalmazásaik. Szegedi Tudományegyetem, Szeged. <https://www.inf.u-szeged.hu/~tothl/ann/Neuronhalok-egyben.pdf> (2022.09.15.)

Zsótér Brigitta (2007): A Hotel Nonius szolgáltatásait igénybe vevők földrajzi megoszlása. *Agrár- és Vidékfejlesztési Szemle* 2 (2): 201–206.

Zsótér B. (2017): Financial planning in connection with accomodation development in a sport centre. *Quaestus Multidisciplinary Research Journal*, 4 (11): 172–177.

Zsótér B., Kaliczka Renáta (2014): Examinations carried out in relation to the shopping habits and satisfaction of costumers in the shops of Coop Szeged Ltd. *Review of Faculty of Engineering Analecta Technica Szegedinensia*, 8 (1): 38–41.