Gépi tanulás vizsgálata alacsony mintaszámú tanítóhalmazok esetén

Ábrahám Domonkos Péter
Molekuláris bionika mérnöki BSc

2019

Témavezető: Dr. Horváth András

Tartalom

Kivonat

Az elmúlt években számos területen áttörést hozott a gépi tanulás, azon belül a neurális hálózatok alkalmazása. Ezen programozási módszer lényege, hogy tanítómintákat adunk a rendszernek és valamilyen feladatra tanítjuk. E módszer akkor működik jól, ha nagy számú tanítóminta áll rendelkezésre. Kis mintaszám esetén a pontosság nagymértékben lecsökken. Ezt dolgozatomban kísérletileg is megmutatom. Az emberi tanulás azonban azon esetekben is képes absztrakt jellemzők és tulajdonságok szintetizálására, mikor csupán néhány minta áll rendelkezésre egy lehetséges halmazból.

A természetben az emberi tudást messze túlszárnyaló effektivitású rendszerek találhatók. Talán a legjobban az agy képességei ejtik ámulatba az embert. Nem csoda, hogy a gépi tanulás is az e szerv működéséből inspirálódik, nagy részben ennek köszönheti fejlődését. Kutatásom során mélyen tanulmányoztam az emberi agy és a mesterséges neurális hálózatok összefüggéseit, ezt dolgozatomban be is mutatom.

Gyakorlati kutatásom célkitűzése olyan módszer kutatása volt, mely javítja a neurális hálózat pontosságát alacsony mintaszám esetén. Ezt két módon is elértem. Az egyik az volt, hogy a mesterséges neuronok általánosan használt ReLU aktivációs függvénye helyett egy új függvényt konstruáltam. Ezzel 5,11%-kal sikerült a háló pontosságát növelnem a referencia ReLU aktivációs függvényt használó futtatáshoz képest.

Kutatásom másik részeként a Matching Network hálózatarchitektúrával foglalkoztam. Kimutattam, hogy a neurális háló kimenete nagy mértékben függ a support halmaz választásától. Ebből következik, hogy a *support* halmaz elemeinek véletlenszerű választása helyett található egy olyan optimális *support* halmaz, mellyel a klasszifikáció lényegesen jobb lesz a véletlenszerűen generáltnál. A pontosság javulásának mértéke átlagosan 20% körüli.

Abstract

In recent years adoption of neural networks for machine learning succeeded in many area of science. This programming method is efficient, in cases when have lots of training samples; however, if the number of the samples decreases, the accuracy will fall down significantly. I am showing empirical evidence tho this phenomenon in the thesis. In contrast to this, however, human learning can solve such tasks easily, so people do not need thouands of samples to synthetize abstrakt features from a set.

There are a lot of processes in nature, which have so high efficiency, that humans can not even approximate in practice. Maybe the most amazing is the capabality of the human brain. Not surpisingly, that machine learning draw inspiratin from this organ too. This art of science developed mostly due to the brain research. During my work I studied deeply the correspondence of the human brain and the artificial neural networks. I also demonstrate this in my thesis.

During my practical research my aim was to find methods, that improve the accuracy of neural networks in case of low number of traning samples. I achieved this by two ways. Firstly I adopted a new type of activation function instead of the widly used ReLU function. This way tha accuracy increased by 5% compared to the referency running with the ReLU function.

The second part of my research refered to the Matching Network architetcure. I demonsrtated, that the output of this neural network highly deoendent from the chosen support set. From this statement comes that instead of choosing the support set elements randomly, we can find an optimal set. By using this set the classification will be significantly better than by using the randomly generated. I did prove it. The difference in accuracy is around 20%.

1 Bevezetés

1. A FELADAT ISMERTETÉSE

A mesterséges intelligencia és a gépi tanulás fejlődése a 21. században exponenciális növekedést mutat. Jogosan merül fel a kérdés: minek köszönhető ez a növekedés? Az ember mindig fejlődni igyekszik, az aktuálisan kihívást jelentő problémák megoldásán dolgozik. Ennek köszönhető a kezdeti időkben a mezőgazdaság, később a természettudományok vagy az orvostudomány fejlődése. A legfejlettebb technikai vívmányok a gépek, melyek az ipari forradalom következményeként rengeteg fizikai munkát levettek az ember válláról. Hasonló folyamat zajlik napjainkban a mesterséges intelligencia fejlesztésével, csak most a szellemi feladatok terén. A gépi tanulással betanított gépek teljesítménye több területen megközelíti, sőt túlhaladja az emberét. Ez nagy lehetőséget nyit fejlettebb, alkalmazható technológiák fejlesztésére mind az orvostudományban, mind az iparban, mind a kényelmi cikkek fejlesztésében.

A gépi tanulás működésének lényege, hogy az emberi tanulástól az agyfelépítéstől és agyműködéstől ihletetten, jól megválasztott tanítóminták és egy tanítóalgoritmus segítségével a gépet valamilyen feladatra betanítjuk. Ez optimalizációs algoritmusokkal történik. A rendszert független tesztmintákon teszteljük, melyekkel a tanítás során nem találkozott, és a válaszadások pontosságát mérjük.

Napjaink algoritmusai akkor tudnak jó eredményt elérni, ha kellően nagyszámú tanítóminta áll rendelkezésre. Ha a mintaszám lecsökken, a válaszadás pontossága a független teszthalmazon nagymértékben lecsökken. Ez az állítás az emberi agy képességeiről nem mondható el. Mivel az emberi agyat is erősen leegyszerűsítve a gépi tanulásnál használt neurális hálózatokkal modellezhetjük, azt feltételezzük, hogy ezzel a módszerrel jobb tanulási teljesítmény érhető el kevés minta esetén is. Erre irányult a kutatásom. A kevés mintából való tanulási algoritmus fejlesztésére.

A feladatot két különböző megközelítéssel is megkíséreltem ellátni. Az első megközelítésem a mesterséges neurális hálózat alap építő kövének, a mesterséges neuron belső reprezentációjában szereplő, széles körben használt ReLU aktivációs függvénynek továbbfejlesztése volt. Hipotézisem az volt, hogy a válaszadás pontossága növekedni fog.

Kutatásom második felében a Matching Network [1] architektúrát vizsgáltam, melyet kifejezetten *one shot learning*-re fejlesztettek ki. A célkitűzés ezen architektúra teljesítményének növelése a benne szereplő *support* halmaz speciális választásával.

Dolgozatomat a következőképpen építettem fel. Először általánosságban a gépi tanulást mutatom be, annak fejlődését a történelem során, majd a gépi tanulás elterjedt megközelítését, a bayesi megközelítést prezentálom. Ezek után a gépi tanulás eszközét, a neurális hálózatokat mutatom be, azoknak biológiai ihletettségét. Ezen áttekintés után térek rá a fent említett módszerek kipróbálásának, fejlesztésének leírására. Először az aktivációs függvény vizsgálatát írom le, majd a Matching Network architektúrával végzett munkámat mutatom be.

2 A gépi tanulás

2.1 A GÉPI TANULÁS MOTIVÁCIÓI

Mai felgyorsult világunkban rengeteg feladattal kell szembenéznünk, melyeket sokszor monoton, unalmas munkával tudunk elvégezni. Általában az ilyen gépies munkák teszik ki feladataink legnagyobb részét, ahelyett, hogy az igazi problémával foglalkozhatnánk. Ezt a problémát részben megoldotta az elektronika fejlődése, aminek során sok gépet sikerült elektronikával ellátni. Ez sokat csökkentett az emberek által végzendő munkák monotonitásán. Azonban így is sok olyan feladat maradt, ami egy ember számára nem jelent nagy kihívást, egy gépet viszont majdhogynem lehetetlenség algoritmikusan (tanító algoritmus használata nélkül) beprogramozni úgy, hogy képes legyen ellátni azt. Ilyen feladat lehet például a forgalomszámlálás egy üzlet, vagy bármilyen tömegkiszolgálási rendszer esetében. További példa az orvosok számára sok energiát igénylő feladat, amikor egy szövettani mintát kell tüzetesen átvizsgálniuk elváltozást keresve benne.

További kihívást jelent, hogy belefutunk olyan problémákba, amit emberi erővel képtelenség megoldani az adathalmaz nagy mérete miatt. Vannak olyan bonyolult, sokváltozós problémák, hogy leprogramoznunk is elég nehézkes lenne, ha nem egyenesen lehetetlen. Erre jó példa a dinamikusan fejlődő agykutatásban felmerülő probléma: az agyi jelek mintázatainak felismerése, osztályozása. Másik példa a gazdasági folyamatok elemzése, előrejelzések készítése.

Ezen problémákra megoldást tud nyújtani a gépi tanulás (vagy angolul *machine learning*). E programozási módszer lényege, hogy statisztikai módszerekkel a szoftver tanulni képes. Ez azt jelenti, hogy nem mi írjuk meg a program pontos működését. A programozó a problémát írja le a szoftvernek és az megtanulja, hogy hogyan tudja a legoptimálisabban megoldani azt. Ehhez szükség van valamilyen tanuló algoritmus implementálására, illetve egy adathalmazra, amelyen a rendszer rátanul a feladat elvégzésére.

2.2 TÖRTÉNETI ÁTTEKINTÉS

Dolgozatomat a mesterséges intelligencia történeti áttekintésével kezdem. Az ebben a részben leírt információk Russel és Norvig „Mesterséges Intelligencia” című könyvének bevezetőjén alapulnak. [2]

Az emberiség története során a földi élettér kiterjesztése, a természet erőinek hasznosítása és a mindennapi élet körülményeinek javítása érdekében egyre fejlettebb eszközöket, majd később gépeket használt. Régóta felmerült vágy a gondolkodás, az emberi agy működésének segítése olyan eszközökkel, melyek hatékonyabbá tehetik az emberi tevékenységet.

2.2.1 Előzmények a történelmi időkből

A gondolkodás folyamatának megértése már az ókor emberét is foglalkoztatta és számos tudományterület fejlődését ösztönözte: filozófia, matematika, gazdaságtan, neurális tudományok és

napjainkban a számítástechnika. Néhány említésre méltó tudományos mérföldkő a filozófia területén: Arisztotelész (i.e. 384-322) szillogizmus elmélete, René Descartes (1596-1650) elemzése az elme, az anyag és a szabad akarat összefüggéseiről, a John Locke által fémjelzett empiricista (*empiricist*) mozgalom, ami alapja a későbbi logikai pozitivizmus (*logical positivism*) doktrínájának (Ludwig Wittgenstein és Bertrand Russell munkájára alapozva a Rudolf Carnap (1891–1970) által vezetett Bécsi kör alkotja meg).

A matematika terén a formális logika elmélete az ókori görögökig nyúlik vissza, a mai elméletet George Boole (1815–1864) munkássága alapozza meg (ítélet- vagy Boole-logika). Ehhez járul hozzá a számításelmélet és a valószínűség-elmélet kifejlődése, a Thomas Bayes (1702–1761) által felállított Bayes szabály és az ezen alapuló Bayes-analízis, ami ma is egyik fontos alapköve a mesterséges intelligencia rendszerekben a bizonytalan következtetések kezelésének.

A gazdaságtudományok terén is találunk érdemi előzményeket: Adam Smith skót filozófus (1723–1790) a közgazdaságtant, mint tudományt megalapozó, első rendszerbe szedett közgazdasági művében írja le, hogy az egyéni érdek és közérdek konfliktusait, a munka-érték hasznosulása és a munkamegosztás folyamataiban a szabad kereskedelem mechanizmusa tartja egyensúlyban és osztja be a nemzet erőforrásait. Ennek hatására előtérbe kerül a hasznosság elmélete, aminek matematikai vonatkozásait Léon Walras (1834–1910) vizsgálta. Ennek nyomán alakul ki a döntéselmélet és a játékelmélet, az utóbbit Neumann János és Oskar Morgenstern fejleszti ki 1944-ben.

Elvitathatatlan szerep jut a mesterséges intelligencia kifejlődésének megalapozásában a neurális tudományoknak, ami az idegrendszer és elsősorban az agy működését tanulmányozza. Paul Broca (1824–1880) beszédzavar kutatása majd Camillo Golgi (1843–1926) által kifejlesztett festési módszer, ami által az agyban az egyedi neuronok is megfigyelhetőkké váltak, adott kezdő lökést a tudományágnak. Az agy aktivitásának mérésére Hans Berger dolgozta ki 1929-ben az elektroencefalográfia (EEG) elvét. 1990 körül került kifejlesztésre Ogawa és társai által a funkcionális mágneses rezonancia (fMRI) módszere, ami eddig nem látott részletességgel képes feltárni az agy aktivitását.

2.2.2 Fordulópont a XX. század közepén

Az igazi nagy áttörést azonban a mesterséges intelligencia kifejlődésében – szomorú, de mint oly sokszor az emberiség történetében - a haditechnika fejlődése hozta, pontosabban annak egyik „mellékága" a hírszerzés a II. világháború idején. A német hadigépezet a német fejlesztésű Enigma - üzenetek titkosítására és visszafejtésére alkalmazott - elektromechanikus gépet használta a rejtjeles hadászati üzenetek továbbítására. Az első Enigmát Arthur Scherbius német mérnök fejlesztette ki az első világháború végén. Ennek változatait az 1920-as évektől kereskedelmi céllal kezdték el alkalmazni. Használták a spanyol polgárháborúban is, majd a második világháborúban a Harmadik Birodalom. Az Enigmát használói abszolút biztonságosnak tartották, mivel az ezzel kódolt szövegek hagyományos módon megfejthetetlenek voltak. Nem számoltak viszont azzal, hogy a gép által generált titkos szöveget egy másik géppel meg lehet fejteni. Az Enigma feltörhetetlennek tartott kódolását először a lengyel Marian Rejewski vezetésével egy kriptográfusokból és matematikusokból álló csoport törte fel 1932-ben. A II. világháború

alatt a német hadvezetés folyamatosan fejlesztette a gépet a korábbi 3 tárcsás gépből 8, majd 12 tárcsás lett, amiből 4 volt választható. A világháború végére kifejlesztett gépbe lyukválasztó tárcsát is beépítettek, ami lehetővé tette, hogy tetszőleges számban és helyen blokkolják a továbbkapcsolást. A tárcsák lehetséges kombinációinak száma 23760 volt.

A Szövetségesek a rejtjeles üzenetek megfejtésére 1939. augusztusától az Angliai Bletchley Parkban állítottak fel munkacsoportot, melynek vezetője Alan Turing volt. Az intézmény neve Kormányzati Kód és Rejtjel Iskola (Government Code and Cypher School), a kódfejtő rendszer fedőneve ULTRA volt. A munka a lengyelek korábbi tapasztalatai alapján folytatódott először elektromechanikus, később elektronikus számítógépekkel (Mark 2 Colossus számítógép). 1940-től a világháború végéig a legtöbb német üzenetet meg tudták fejteni. Tevékenysége csúcspontján a GC&CS-ben naponta 4000 üzenetet fejtettek meg. A legnehezebbnek a Német Haditengerészet üzeneteinek megfejtése bizonyult. Amikor 1942 februárjában a németek bevezették a négytárcsás Enigma készüléket az Atlanti-óceánon hajózó tengeralattjárókkal történő rádióforgalmazáshoz, ezek megfejtése tíz hónapra lehetetlenné vált. A kódok megfejtésének nagy szerepe volt a tengereken óriási fölényben levő német tengeralattjáró flotta megsemmisítésében is, ami talán túlzás nélkül állíthatjuk, hogy a II. világháború végkimenetelét is eldöntötte.

2.2.3 A II. világháború után

Alan Turing a háború után Londonban, majd 1948-tól a Manchester University-n folytatta kutatói tevékenységét. 1950-ben jelent meg tudományos cikke Computing Machinery and Intelligence („Számítógépek és intelligencia") címmel, amiben elsőként írja le a Turing-tesztet, amivel el lehet dönteni, hogy egy gép gondolkozik-e vagy sem. Ugyanitt vezeti be a „gépi tanulás" és a „megerősítéses tanulás" fogalmakat is.

Az első neurális számítógépet a Princeton Egyetem matematika tanszékén két végzős hallgató – Marvin Minsky és Dean Edmonds - építette meg 1951-ben. A gépben 3000 elektroncső és a B-24-es bombázó automatapilóta mechanizmusa egy 40 neuronból álló hálózatot szimulált.

Princetonban végzett az a John McCarthy is, aki - Minsky-vel együttműködve - az automata elméletben, a neurális hálókban és az intelligencia kutatásban érdekelt angol és amerikai kutatókat 1956 nyarán Dartmouth College-ba hívta egy 2 hónapos munkatalálkozóra. A találkozón többek között az IBM és a Massachusetts Institute of Technology kutatói is részt vettek. A találkozón elfogadták a tudományterület McCarthy által kreált új nevét a mesterséges intelligenciát (MI) (*artificial intelligence* (AI)). A MI a kezdetektől fogva céljának tekintette az olyan emberi képességek duplikálását, mint a kreativitás, az önfejlesztés és a nyelv használata.

John McCarthy 1958-ban Dartmouth-ból a Massachusetts Institute of Technology-hoz ment át és itt definiálta először a későbbiekben elsődleges MI programozási nyelvvé váló Lips-et. Ugyanebben az évben „Program with Common Sense" című cikkében írja le az „Advice Taker"-t, egy hipotetikus programot, ami az első teljes MI rendszernek tekinthető.

2.2.4 A 60-as évektől a 21. századig

1963-ban McCarthy a Stanford University-n megalakítja a MI labort. Kutatásai elsősorban a logikai következtetés általános módszereire irányultak, aminek lökést adott a J. A. Robinson által felfedezett rezolúció (az elsőrendű logika teljes bizonyítási eljárása).

Ezzel párhuzamosan a neurális hálón alapuló kutatás is folytatódott. Frank Rosenblatt bevezette a perceptronokat és bebizonyította a perceptron konvergencia tételét (perceptron convergence theorem). Később Minsky és Papert „Perceptrons" című könyvében (1969) a perceptronok „taníthatóságának" korlátaira hívta fel a figyelmet. Ennek hatására a neurális hálók kutatásának finanszírozása drasztikusan visszaesett annak ellenére, hogy a többrétegű neurális hálók hiba visszaterjesztéses tanuló algoritmusát (error backpropagation) éppen ekkor fedezi fel Bryson és Ho (1969).

A Stanford University kutatói (Buchanan és társai) 1969-ben megalkotják a „Dendral" nevű programot, ami az úgynevezett „nehéz problémák" megoldására területspecifikus tudást használt fel. Néhány évvel később Fegenbaum és Buchanan belefogtak egy heurisztikus programozási módszerbe, hogy megvizsgálják a szakérő rendszerek új módszertana milyen módon alkalmazható más szakmai területeken. Az első komoly eredmény az orvosi szakterületen valósult meg, a vérrel kapcsolatos fertőzések diagnosztizálásásra fejlesztették ki a „Mycin" rendszert. A másik jelentős szakterület a nyelvfelismerés volt, Winograd fejleszti ki a „Shrdlu" nyelvfelismerő rendszert. Az első üzletileg sikeres szakértőrendszert a Digital Equipment Corporation-nél alkalmazták az 1980-as évek elejétől. A szakértőrendszer az új számítógépes rendszerek megrendeléseit segítette konfigurálni.

A legutóbbi évtizedek a MI kutatások módszertanában és tartalmában is lényeges változásokat hoztak. Az új elméletek helyett a létező elméletekre építés, a komoly kísérleti bizonyítékokra és a bizonyított tételekre alapozás került előtérbe. Ennek szemléltetésére álljon itt egy David McAllester amerikai MI kutatótól származó idézet 1998-ból:

> *„Fel kell ismerni, hogy a gépi tanulást nem szabad elszigetelni az információelmélettől, hogy a bizonytalanság melletti következtetést nem szabad elszigetelni a sztochasztikus modellezéstől, hogy a keresést nem szabad elszigetelni a klasszikus optimalizálástól és szabályozástól, és hogy az automatikus következtetést nem szabad elszigetelni a formális módszerektől és a statikus elemzéstől."*

2.3 A GÉPI TANULÁS BAYESI MEGKÖZELÍTÉSE

A bevezetőben láthattuk, hogy milyen új lehetőségeket nyit elénk a gépi tanulás. Azonban minden jó ötlet akkor használható a mérnök gyakorlatban, ha egy jó matematikai modellt tudunk definiálni hozzá. Ezt a modellt tudjuk majd implementálni a számítógép nyelvén, mely a nagy számításigényű feladatot hatékonyan el tudja végezni. Jogosan merül fel tehát a kérdés, hogy hogyan tudjuk a gépi tanulást - első lépésben matematikailag - megközelíteni?

A gépi tanulás formalizálásának elterjedt, bevált módja a bayesi statisztikai megközelítés. A modell megértéséhez kitérőt teszünk a bayesi statisztika alapjaihoz. A bayesi gondolkodásmód a statisztikában egy más megközelítési, szemléleti módot igényel, mint ami a klasszikus statisztikában megszokott. Igaz sok közös pontot is tartalmaznak, bár ezek elsősorban módszertani hasonlóságok. Az alap hozzáállás tehát más.

2.3.1 A bayesi gondolkodásmód

Az ebben a részben leírtakhoz Hunyadi László „A bayesi gondolkodás a statisztikában" című cikkét használtam forrásként [3].

A Bayes-tétel és jelentősége

A bayesi statisztika Thomas Bayes 18. században megalkotott tételéből indul ki, mely a következőképpen szól. Ha A és B egymástól nem független események (tehát az az eset, hogy mindkét esemény bekövetkezik nem egyenlő a két esemény valószínűségének szorzatával), akkor az együttes előfordulásukra a feltételes valószínűségek definícióját felhasználva felírhatjuk a következő összefüggéseket:

$$\Pr(A \text{ és } B) = \Pr(A|B)\Pr(B)$$

$$illetve$$

$$\Pr(B \text{ és } A) = \Pr(B|A)\Pr(A)$$

Mivel $\Pr(A \text{ és } B) = \Pr(B \text{ és } A)$, az egyenletek jobb oldalai egyenlővé tehetők. Az így kapott egyenletből egyszerű leosztással adódik a Bayes-tétel közismert alakja:

$$\Pr(B|A) = \frac{Pr(A|B)\Pr(B)}{\Pr(A)} \qquad (1)$$

Ami ebben az egyenletben nagy lehetőséget nyit az az, hogy Bayes ezzel az összefüggéssel kapcsolatot teremt a két feltételes valószínűség között, melynek a jelentőségét akkor láthatjuk, ha az A, B eseményeknek ok-okozati értelmezést adunk:

> *„Amennyiben a feltételes valószínűségeknek ok-okozati interpretációt adunk, akkor az okok és okozatok egymással való kapcsolatát, az okozati láncon való kétirányú mozgást írja le […]. Mindenképpen a fordított irányú gondolkodás (inverz-valószínűségek számítása) az az elem, ami itt lényeges, és ez volt az a mozzanat, ami a klasszikus alkalmazásokat jellemezte" [3, p. 1152]*

A bayesi gondolkodás a statisztikában

Az előző részben láttuk a Bayes-tételt és annak közvetlen következményét. Hogyan következik ebből a bayesi gondolkodás, mint általános módszertani keret? Ennek megértéséhez úgy juthatunk el, ha a következőkképpen értelmezzük az (1) egyenletben lévő *A* és *B* eseményt.

Legyen a H egy tetszőleges hipotézis, E pedig egy tapasztalati bizonyítéka H-nak. Avagy H az „ok", E az „okozat". Definiálnunk kell továbbá a *Pr(H)*, *Pr(E)*, a *Pr(E|H)* és a *Pr(H|E)* valószínűségeket. A *Pr(H)* legyen a *H* hipotézis tapasztalás előtti (priori) valószínűsége *Pr(E)* a bizonyíték bekövetkezésének valószínűsége. A *Pr(E|H)* annak a valószínűsége, hogy a bizonyíték bekövetkezik, amennyiben *H* teljesül. *Pr(H|E)* pedig arra a kérdésre ad választ, hogy mennyi a valószínűsége annak, hogy E bekövetkezése esetén a *H* hipotézis igaz.

Ebben a felosztásban a Bayes-tétel így írható fel:

$$\Pr(H|E) = \frac{Pr(E|H)\Pr(H)}{\Pr(E)} \qquad (2)$$

A gyakorlatban általában a *H*-t nem csak egy eseményként, hanem H_1, H_2, …, H_K eseményrendszerként értelmezzük. Ha ezt is felhasználjuk az egyenlet konstruálásakor, illetve felhasználjuk továbbá a teljes valószínűség tételét, akkor a következő összefüggést kapjuk:

$$\Pr\left(H_j\middle|E\right) = \frac{Pr\left(E\middle|H_j\right)\Pr(H_j)}{\sum_{i=1}^{K}\Pr(E|H_i)\Pr(H_i)}\ , j = 1,2,\dots,K \qquad (3)$$

Ebben az összefüggésben a *Pr(H)*-t prior valószínűségnek, a *Pr(H|E)*-t posterior valószínűségnek nevezzük.

Hunyadi rávilágít arra, hogy ez az összefüggés jól modellezi, lényegében alapját képzi a mai tudományos gondolkodásunknak. A világról van egy előzetes tudásunk, melyet egy valószínűségben fejezünk ki. Ez a prior valószínűség (*Pr(H)*). Ez a valószínűség alapulhat mérések sokaságán, de lehet teljesen szubjektív is (sőt legtöbbször a szubjektív elemek nem zárhatók ki teljes mértékben, nem tudunk teljesen objektívek lenni a modellalkotásunk során, de ez a szubjektivitás legtöbbször elhanyagolható). Ezt fogalmazza meg Hunyadi is cikkében:

> *„A klasszikus módszertan (beleértve természetesen a statisztikát is) következtetéseit tapasztalatokból (megfigyelések, mérések stb.) vonja le, hangsúlyozva azok objektív (az elemzést végző személytől független) jellegét. Ezzel szemben a bayesi gondolkodás modellje általánosabb, hiszen azt mondja ki, hogy létezhetnek (sőt szinte mindig léteznek) mintán kívüli információk, amelyek egy része lehet szubjektív is (hiszen az elemzőnek lehetnek előzetes ismeretei a tárgyról), és amelyek az egész tudományos következtetést vagy döntést befolyásoló lényeges tényezők lehetnek." [3, p. 1155]*

Ezek után mintavételezést hajtunk végre és az empirikus következményt figyeljük meg, ami feltételezésünk szerint *Pr(E|H)* valószínűséggel a hipotézisünk következménye.

Végül a posterior valószínűséget kiszámoljuk a (2) vagy a (3) egyenlet segítségével. Ez az érték arról ad nekünk egy valószínűséget, vagy valószínűség-eloszlást, hogy a hipotézisünket milyen mértékben támasztja alá az adott mintavétel.

A bayesi gondolkodás egyfajta oknyomozó gondolkodást igényel. mivel az okozatot figyeljük meg és abból következtetünk az okok valószínűség-eloszlására.

2.3.2 A gépi tanulás bayesi modellje

A gépi tanulás modellje bayesi megközelítésének lényege, hogy egy prior tudásunk van. Ezt a tudást lényegében az adatbázis reprezentálja, amit tanítóhalmazként használunk. Azonban tudatosítanunk kell, ez tudás lehet különböző mértékben szubjektív, ami nagyban befolyásolhatja a mesterséges intelligenciánk teljesítményét.

Álljon itt egy kokrét példa, hogy egyértelműbb legyen, hogyan kell ezt értelmezni:

Példa

Prior tudás – szubjektivitás a rendszerben

Legyen a tanítóhalmazunk az MNIST [4] adatbázisból vett elemek. Emlékeztetőül: az MNIST adatbázis kézzel írott egyjegyű számokat tartalmaz, melyek egyesével fel vannak címkézve, így kiválóan alkalmasak felügyelt tanuláshoz. (A címkék azt jelölik, hogy az adott képek milyen számjegyet ábrázolnak)

A felcímkézett adathalmaz egy előzetes tudást reprezentál (prior). Több szempontból is láthatjuk ezt. A legnyílvánvalóbb - ami a célunk is, hogy ezt tanulja meg a gép – a számok alakja. Tehát hogy a kilences úgy néz ki, hogy felül egy kör, jobb oldalából lefele húzott vonallal. Azt gondolhatnánk, hogy ez nem hoz szubjektivitást a rendszerbe, hiszen ez nyílvánvaló objetív tény mindenki számára. Azonban ha például a hetes számjegyet vesszük, akkor máris problémába ütközhetünk e téren. Hogy miért? Azért, mert az MNIST amerikai emberek kézzel írott számait tartalmazza, ahol a hetest máshogy írják mint Magyarországon (1. ábra).

1. ábra: Amerikában (bal oldali) és Magyarországon (jobb oldali) a kilences írásmódja különbözik

Ebből klasszifikációs hibák adódhatnak. Ki is próbáltam egy számfelismerő programmal [5] a jelenséget, amit az alábbi ábra mutat.

2. ábra: A 9-es szám eltérő írásmódja miatt az A. W. Harley klasszifikátora [2] rosszul osztályozza a „magyarosan" írt kilencest (bal oldal). Ha az alsó ívet letöröljük, akkor már helyes a klasszifikáció (jobb oldal).

Jól látható, hogy az eltérő írásmód miatt az általam írt kilences alsó íves részét a szoftver a hármas illetve a nyolcas számok jelegzetes jegyének ismerte fel. Ha letöröltem az ívet, akkor helyreállt a klasszifikáció. Megjegyzem, nem az első próbálkozásra sikerült átverni a hálót, illetve szükség volt a szóban forgó also ív kicsit túlzott megjelenítésére.

Azonban ez nem minden, a prior tudásnak ebben a pédában vannak más dimenziói is. Prior tudás lehet az is, hogy ha a mintában a kilencesek a szám írójának írásképéből adódóan például mindig jobbra dőlnek. Ha ez lenne a helyzet, akkor egy balra dőlő írásképű tesztalany kilenceseit nem biztos, hogy felismerné a rendszer.

Az előbbikeből láthatjuk, milyen fontos a jó tanítóhalmaz meghatározása, megalkotása. Ez nagy kihívást jelent a tervezőnek, és mondani sem kell, tökéletes tanítóhalmaz nem létezik, azonban törekedni kell a diverzitásra.

2.4 NEURÁLIS HÁLÓZATOK

Az előző fejezetben a gépi tanulás statisztikai megközelítését mutattam be, ami megalapozza a tudományághoz való matematikai hozzáállását napjainkban. Ez emléteti síkon közelítette a problémát. Hogyan lehet a gyakorlatban megvalósítani egy tanulásra képes gépet? Erre ad választ ez a fejezet.

A gépi tanulás modellezéséhez a modellező eszköznek alapvetően a következő képességekkel kell rendelkeznie. rendelkeznie kell egy vagy több bemenettel, ahol digitalizált mintákat fogadnia kell tudnia. A belső reprezentációja kellően komplex kell legyen ahhoz, hogy komplex probléma esetén jó döntést tudjon hozni (prediction). Végül a döntését melyet a kimeneten reprzentálnia kell tudnia. Tanítható kell legyen,

tehát egy általunk megadott tanítóminta halmaz segítségével optimalizációt kell, hogy tudjunk végrehajtani a paraméterein.

Ezen tulajdonságok mindegyikével bírnak az úgynevezett mesterséges neurális hálózatok. Ez az architektúra rendkívül jól teljesít gépi tanulási feladatok megvalósítójaként, jól tanítható, gyors. Napjainkban szinte minden mesterséges inteligencia alapú rendszer neurális hálózatokat használ, legyen szó arcfelismerő, beszédfelismerő, vagy személyre szabott hirdetésajánló rendszerről.

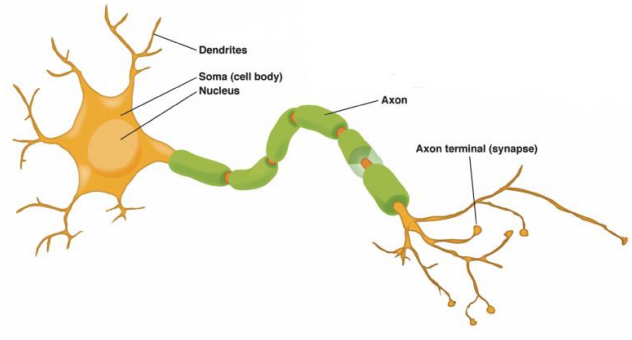
A neurális hálózatok alapjait McCulloch és Pitts fektették le 1943-ban [6]. Modellükben még nem szerepelt bias, illetve az aktivációs függvényük (akkor még nem nevezték így) bináris (0,1) kimenetű volt. Később 1950-ben Roosenblatt fejlesztette tovább a ma használt formára [7].

2.4.1 A neurális hálózatok ihletője: az emberi agy

A neurális hálózatokat az agyi idegsejt hálózatok ihlették. Az emberi agy rendkívül nagy komplexitású feladatokat képes nagyon rövid idő alatt megoldani. Példaként álljon itt az emberi agy beszédmegértő képessége. Ehhez az agynak a fül által közvetített tüzelési mintázatot - mely akár erősen zajos is lehet – kell jelentéssel összekapcsolnia valós időben. Ezt a feladatot rendkívül hatékonyan el tudja végezni akár egy egyészen kis gyermek agya is. [8]

Az agy pontos működéséről még nagyon keveset tudunk. Azonban a mai tudomány már felfedezte, hogy az agyban nagyon nagyszámú (megközelítőleg 200 milliárd) idegsejt (neuron) található. Ezek egymáshoz kapcsolódnak és egy rendkívül összetett hálózatot alkotnak. Ez az idegsejthálózat valósítja meg a hihetetlenül komplex és szerteágazó agyműködés funkcióit. [8] Ami igazán figyelemre méltó ebben az az, hogy a neuronok különálló egységekként funkcionálnak. Tehát működésük csak a velük kapcsolatban lévő neuronoktól függ és ők is csak az axonjukkal kapcsolódó neuronokra vannak hatással. Tehát nincs központi rendezőerő.

Az idegsejtek három fő alkotó részből állnak: a dendritekből, a sejttestből és az axonból. A szerteágazó dendritek begyűjtik az úgynevezett posztszinaptikus potenciálokat, melyek a hozzá kapcsolódó neuronokon érkező akciós potenciálok hatására keletkeznek. Ezek ionáramként folynak be a sejttestbe. A sejttestben ezek a potenciálok összegződnek. Ha a potenciál mértéke megüt egy bizonyos küszöbértéket, a sejt axonján akciós potenciál indul meg. A sejt axonja szintén kapcsolódik más neuron dendritjéhez. Ennek köszönhetően posztszinaptikus potenciálókat generál, hasonló folyamatot indítva meg bennük is. Ez a működés játszódik le az agyban minden egyes másodpercben a 200 milliárd agyi idegsejt között. És ami az igazán nagy csoda, - jelenlegi tudásunk szerint – ez az egyszerű működés teszi lehetővé a rendkívül bonyolult és sokféle agyi funkció ellátását.



3. ábra: A mesterséges neuron ihletője: a biológiai idegsejt. kép: [9]

Az agy azonban önmagában „csak" egy jelfeldolgozó. Tehát, hogy értelme legyen, kellenek jelek, amiket feldolgoz és kimenetek, melyekkel valamilyen válaszreakciót valósít meg. Az agy az érző idegeken bejövő jeleket dolgozza fel. A lánc elején tehát az érző idegsejtek vannak. Ők érzékelik a külvilágból érkező, az adott idegsejt által érzékelhető hatást. A jeleket az agy feldolgozza, majd valamilyen folyamatot indít.

Talán jobban megértjük a folyamatot egy triviális példa segítségével. Legyen az a példa, hogy ha az ember meglát egy kóbor kutyát, aminek habzik a szája, akkor távolságot tart tőle, mert feltételezi, hogy veszett. Ebben az esetben leegyszerűsítve a következő lépések történnek. Az előttünk lévő képet a fotoreceptor idegsejteken keresztül a látóideg közvetíti az agyba. Az idegen keresztül az agyba már egy digitalizált adat érkezik, mivel a receptorok a rájuk eső fényt egy tüzelési mintázattá alakítják. Ezt az agy rendkívül effektíven és gyorsan értelmezni képes a 200 milliárd neuronhálózatával és információkat nyer ki a képből. Például felismeri, hogy van valamilyen objektum a képen. Aztán az objektumot osztályozza, felismeri, hogy az egy kutya. És azt is felismeri, hogy habzik a szája. Ez már elég információ, hogy arra következtessen, hogy nagy valószínűséggel veszett. Az információ feldolgozás itt lényegében véget ér. Az utolsó lépésben válaszreakciót generál az agy. Ezt úgy teszi, hogy különböző parancsokat ad ki. Például irányváltoztatási parancsot, mellyel a kutyát elkerüli az ember.

Ha az agy működésére egy nagyon egyszerű modellt akarunk alkotni az előző részben leírtak alapján, akkor közelíthetjük a következő módon: az agy egy neuronhálózat, amelynek vannak bemenetei a receptorok, illetve vannak kimenetei is: a lehetséges viselkedések parancsai. A bemenetek és a kimenetek között pedig egy nagyon komplex, gyors döntésre képes neurális hálózat van, melyben a neuronok a fent leírt módon kommunikálnak egymással.

Ezen a ponton jutunk el a neurális hálózatokig, hiszen e modell matematikai leírásával kapjuk meg őket. A modell leírásához először a mesterséges neuront kell definiálnunk.

2.4.2 A mesterséges neuron

Aktiváció kiszámítása

A mesterséges neuronok a neurális hálózatok alap építő kövei. Ahogy az agyi idegsejteknek vannak bemeneti (dendritek), úgy a mesterséges neuron is rendelkezik bemenetekkel. A bemeneteken számértékek jönnek (x_i), melyek megszorzódnak egy úgynevezett szinaptikus súllyal (w_i). Ezek a szorzatok összeadódnak, tehát matematikailag a bemenetek lineáris kombinációját képezik. Emlékezzünk, a neuronok kimenetén akkor generálódott jel, ha a bemeneten érkező potenciál egy bizonyos küszöbértéket megütött. Ezt matematikailag úgy reprezentálhatjuk, hogy az összegből levonunk egy konstans bias értéket (b). Ezt az egészet hívjuk a neuron aktivációjának (a). Tehát az aktiváció a következő képlettel számítható:

$$a = \sum_i w_i x_i \quad - b$$

Aktivációs függvény

Egy elem hiányzik még a mesterséges neuron definiálásához: az aktivációs függvény. Az aktivációs függvénynek is található megfelelő az agyi idegsejtben. Ez a neuron belső felépítésével vonható párhuzamba.

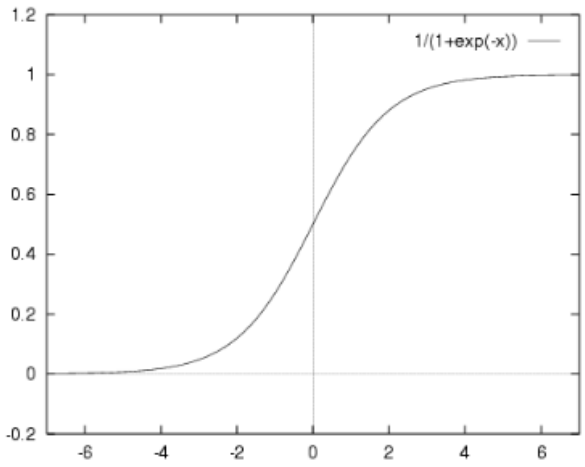
Az aktivációs függvény (f) az aktivációt kapja bemenetként és egy számot rendel hozzá. Ez le a szám lesz a neuron kimenete (y). Képlettel így formalizálhatjuk tehát a mesterséges neuron kimenetét:

$$y = f\left(\sum_i w_i x_i \quad - b\right)$$

Aktivációs függvénynek többféle függvényt választhatunk. Az aktivizációs függvény szinte mindig egy nemlinearitás. (Ez alól csak ritka kivételek vannak, mint például a Blum Li konstrukció [10] utolsó neuronjának a függvénye.) Azért fontos a nem lineáris tulajdonság, mert ha lineáris volna a függvény, akkor neuronok hálózata esetén az egész rendszert le lehetne egyszerűsíteni egy lineáris mátrixtranszformációra. Így a magasfokú komplexitás nem valósulna meg, ami pont a célunk lett volna. A nemlineáris aktivációs függvényeknek köszönhetően a neurális hálózatok úgynevezett unierzális aproximátorok. Ez azt jelenti, hogy bármilyen függvény reprezentálható velük, illetve rá tudnak tanulni tetszőleges függvényalakokra. [11] [12]

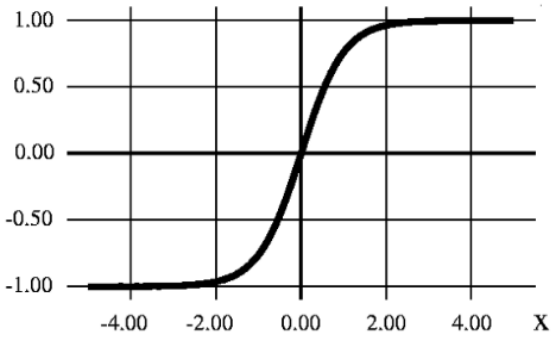
Mint említettem, aktivációs függvénynek többfajta függvényt is használhatunk. A következőkben leírom a legszélesebb körben használtakat, és rövid jellemzésüket. [11]

Aktivációs függvényként használható a sigmoid függvény. Képlete a következő: *f(x) = 1 / (1 + exp(-x))*. Előnye, hogy könnyű alkalmazni. Viszont van néhány hátránya, ami matt a manapság már kisebb népszerűségnek örvend. Egyik ilyen hátránya, hogy lassan konvergál a tanítás során. Az sem szerencsés, hogy értékkészletük nem 0-központú. Ez a tulajdonsága az optimalizációt nehezíti.



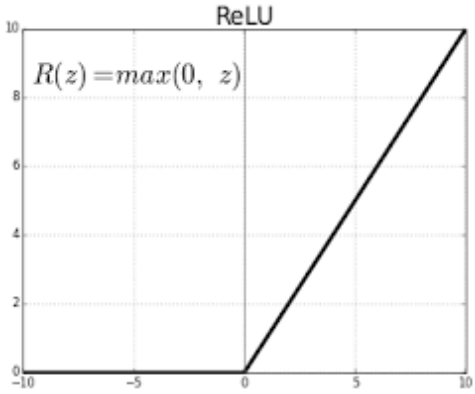
4. ábra: Sigmoid függvényre

Ezt a problémát (mármint a nem 0 központúságot) megoldja a tangens hiperbolikus függvény: *f(x) = 1—exp(-2x) / 1 + exp(-2x)*. Ennek a kimenete már nulla központú, hiszen az értékkészlete -1 és 1 közötti. Így az optimalizáció már könnyebb ezzel a függvénnyel. Azonban a másik probléma megmaradt. A konvergációja ennek a függvénynek is lassú.



5. ábra: Tangens hiperbolikus függvény

A harmadik aktivációs függvényként használt függvény, amit bemutatok a ReLU (*Rectified Linear units*). Ez a függvény kifejezetten fontos a szakdolgozatom során, mert a kutatásom első felében ezt a függvényt vettem alapul, fejlesztettem tovább. A ReLU képlete a következő: *R(x) = max(0,x).* Ez a függvény nagyon széles körben használt több igen kedvező tulajdonsága miatt. [11]



6. ábra: ReLU függvény

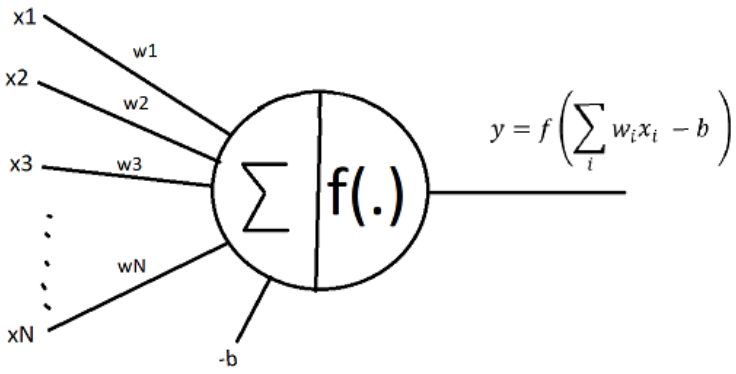
Ami a legszembetűnőbb, hogy a függvény deriváltja nagyon egyszerű (0, ha x kisebb, mint 0 és 1, ha x nagyobb, mint nulla). Ez azért fontos, mert a tanítás során az *error backpropagation* algoritmushasználata során minden neuronnak ki kell számolni a lokális hibáját, amihez szükségünk van az aktivációs függvény deriváltjára (ld. következő rész). Ez meglehetősen számításigényes, ha bonyolultabb a deriválás és sok neuron van a hálózatban.

Ezzel a függvénnyel a tanítás már gyorsan konvergál, hiszen a deriváltja (x nagyobb, mint nulla esetén) mindig 1.

Azonban hátrányai is vannak a ReLU függvénynek, ezeket fejlesztettem tovább a kutatásom első felében. Ezeket, illetve lehetséges továbbfejlesztésüket az „Aktivációs függvény vizsgálata" részben fejtem ki.

Megjegyzendő: a következő részben taglalt *error backpopogation* tanítóalgoritmusnak kritériuma, hogy a függvény deriválható legyen, ugyanis az egyes lokális hibák kiszámításához szükség van a függvény deriváltjára. Így, ha ezt a tanítóalgoritmust használjuk, akkor mindenképpen deriválható függvényt kell választanunk aktivizációs függvénynek.

> *Amikor 1943-ban McCulloch és Pitts* [6] *megalkották modelljüket abban egy egyszerű lépcsős függvényt használtak, melynek értéke nulla, ha az aktiváció negatív, és egy abban az esetben, ha az aktiváció pozitív. További különbség, hogy még nem volt a modelljükben bias, hanem maga a lépcsős függvény reprezentálta a trasholdot.*
>
> *Ez a konstrukció kiválóan alkalmas volt logikai függvények reprezentálására. Azonban bináris kimenete miatt a lehetőségek korlátozottak bonyolultabb problémák reprezentálása esetén. Ezt megoldandó fejlesztette tovább a modellt Roosenblatt 1950-ben bevezetve a biast, mely így már funkcionált egyfajta trashold-ként, és utat nyitott a folytonos, nem bináris aktivizációs függvényeknek.* [7]



7. ábra: A mesterséges neuron sematikus ábrája és a kimenet kiszámítási módja

2.4.3 Neurális hálózat

Előállt tehát a mesterséges neuron modellünk. A neurális hálózat (háló) úgy áll össze a neuronokból, hogy egyes neuronok kimeneteit bekötjük más neuronok bemeneteire.

A könnyű hivatkozás érdekében a neurális hálózatokban különböző rétegeket különböztetünk meg. A neuronok első rétegét bemeneti rétegnek (*input layer*) nevezzük. Az utolsó réteg a kimeneti réteg (*output layer*). A közbülső rétegeket – ha vannak – rejtett rétegeknek (*hidden layers*) nevezzük. Az olyan gépi tanulási megvalósítást, mely olyan neurális hálózattal történik, mely tartalmaz rejtett rétegeket, mélytanulásnak (*deep laerning*) nevezzük.

A neurális hálózatok csoportosíthatók aszerint, hogy hogyan vannak a neuronok összekapcsolva bennük. Vizsgálataim során úgynevezett előrekapcsolt neurális hálózatokat (*feed forward neural network*) használtam. Ennek az a tulajdonsága, hogy az egyes réteg neuronjainak kimenetei mindig csak a következő réteg neuronjainak bemeneteit képzik. Ha hálóban minden neuron kimenete be van kötve minden következő rétegbeli neuron bemeneteként, akkor *fully connected* hálóról beszélünk.

A neurális hálók, mint említettem jól használhatók döntési feladatok ellátására. Ez a következőképpen történik. A paraméterektől függően valamilyen kimenetet ad. A kimenet lehet jó vagy kevésbé jó. A kimenet milyenségét felügyelt tanulás esetén (ld. a következő részben) úgy tudjuk mérni, hogy definiálunk egy hibafüggvényt (*loss function*). A hibafüggvény bemenetei a kívánt kimenet és a háló kimenete. Ha a hibafüggvény értéke nagy, az azt jelenti, hogy a háló kimenete messze van a kívánttól, tehát a válasza „nem olyan jó”. Ha a hibafüggvény értéke kicsi, az azt jelenti, hogy közel van a jó válaszhoz a háló, ha nulla, akkor a háló válasza tökéletes. A célunk a tanítás során (ld. következő rész) a háló paramétereinek (szinaptikus súlyok és biasok) optimalizálása úgy, hogy a hibafüggvény a bemenetekre átlagosan a legkisebb legyen.

2.4.4 Neurális hálózat tanítása

Bemutattam a neurális hálózat architektúráját. Egy háló azonban még önmagában nem tud elvégezni egy feladatot, pontosabban, ha tudjuk a feladat optimális megoldását adó hálózat összes paraméterét (a szinaptikus súlyokat, illetve a biasokat) és ezeket kézileg beállítjuk, akkor el tudja végezni. Ez azonban szinte lehetetlen komplexebb feladatoknál. Sőt, pont az a célunk, hogy olyan feladatokat oldjunk meg ezzel az architektúrával, amit nem mi programozunk be kézileg, hanem a gép tanul meg. Ezért is gépi tanulás a programozási technika neve.

Az agy tanulási képessége

Emlékezzünk: a gépi tanulás az emberi agy működéséből merít inspirációt. Az emberi agy sem csak beprogramozott működésekre képes. Vannak ilyen funkciók is az idegrendszerben. Ezek a feltétlen reflexek azonban az agy sokkal több ennél. Képes a változó környezethez alkalmazkodni, képes új dolgokat megtanulni. Ez a nagymértékű plaszticitásának köszönhető, mely kisgyermekkorban a legnagyobb, de egészen felnőttkorig megmarad. Ez azt jelenti, hogy a neurális kapcsolatok, útvonalak meg tudnak erősödni, illetve új kapcsolatok jöhetnek létre. [8]

Gondoljunk vissza a veszett kutyás példára! Tehát sétálunk egy elhagyatott úton és egy habzó szájú kóbor kutyát pillantunk meg. Tegyük fel, hogy ez esetben nem tudjuk, hogy veszélyforrást jelent számunkra. Óvatlanságunk eredményeképp a kutya belénk harap. A sebészeten kötünk ki, el kell szenvednünk egy tetanusz oltást is és a kedvenc csapatunk meccséről is lemaradunk. Legközelebb újra találkozunk egy habzó szájú kutyával. Ebben az esetben már messziről elkerüljük. Látható, hogy az agyban valami megváltozott. Hiszen az első esetben a kutya látványa által kiváltott vizuális ingernek, nem volt „óvatossági parancs" a válaszreakciója (kimenete). Míg a második esetben igen. Ha az agyat neuronok hálózataként modellezzük, akkor kijelenthetjük, hogy a háló paraméterei megváltoztak az eset hatására.

A tanítás modellje

Az előző részben bemutatott példa mintájára elkészíthető a tanítás modellje. Mit láttunk a példában? Volt egy helyzet, azaz az agy egy bemenetet kapott: a kutya látványa. A lehetséges kimenetek száma az agy esetében közel végtelen, hiszen az emberi cselekvés körülbelül bármi lehet. Egyszerűsítsük le mégis a kimenetek halmazát mondjuk úgy, hogy óvatos magatartás és bátor magatartás legyenek az elemei. Ebben a felosztásban az első esetben a bemenet a kutya volt, a kimenet pedig a bátor magatartás. A második esetben szintén a kutya volt a bementet, de itt már az óvatos magatartás volt a kimenet. Látjuk tehát, hogy a páciensünk megtanulta a kívánt kimenettől való eltérés következtében, hogy a kutya látványa esetén a kívánt kimenet az óvatos magatartás. Ilyen bemenet-kimenet párokra szükségünk lesz tehát a tanításhoz. Pontosabban, ez egy lehetséges mód a tanításhoz.

Tanítóhalmaz

Definiálhatunk tehát egy tanítóhalmazt, mely *(bemenet, kívánt kimenet)* párokból áll. Ha a tanítóhalmaz ilyen bemenet-kimenet párokból áll, akkor felügyelt tanulásról beszélünk [2]. Kutatásom során felügyelt tanítást alkalmaztam.

Hibafüggvény

Definiálhatunk így egy – az előző részben már említett – hibafüggvényt, vagy veszteségfüggvényt (angolul *loss function*) is a következő módon. Ez egy távolságfüggvény, aminek értéke nagy, ha a háló adott mintára adott kimenete nagyban eltér az elvárt (címke szerinti) kimenettől. A hiba értéke nulla, ha a háló válasza tökéletes (azaz megegyezik az elvárttal).

Tanítóalgoritmus

Szükségünk lesz továbbá egy tanítóalgoritmusra, ami megvalósítja a tanítást. A neurális hálózatok tanító algoritmusa az *errorbackpropagation* (hibavisszaterjesztéses) algoritmus. [13]

Az algoritmus elve az, hogy a kimeneti rétegen kapott – hibafüggvény által megadott – hibát visszaterjeszti az előbbi rétegekre a neuronok közötti éleken lévő súlyok arányában. Így minden neuronhoz hozzárendelhetünk egy lokális hibát.

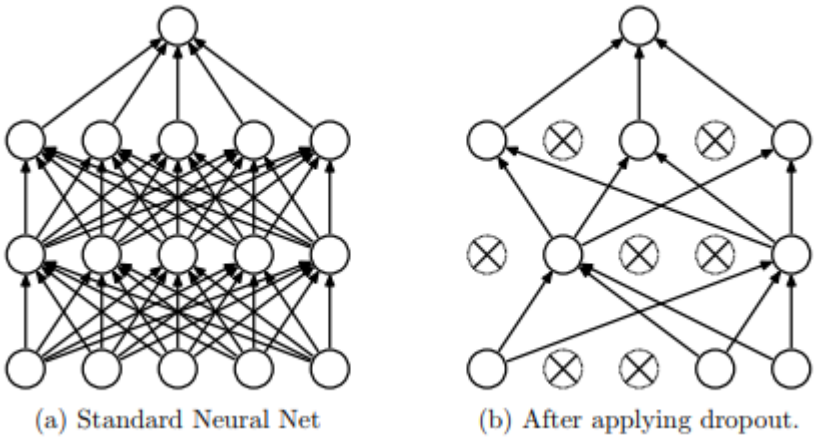
Az algoritmus utolsó lépése a súlyok adjusztálása, azaz a lokális hibájuk mértékében a neuronok súlyainak módosítása. A módosítás mértéke állítható paraméter, ezt *learning rate*-nek hívjuk. A *learning rate* növelésével gyorsíthatjuk a tanítást. Azonban vigyázni kell azzal, hogy ha túl nagyra állítjuk a *learning rate* értékét, lehetséges, hogy átugorjuk a tanítás során az optimális súlymátrixot.

Ez az algoritmus elegendő iterációig futtatva jó közelítéssel megtalálja a neuronoknak egy optimális súlymátrixát, amely a legtöbb tanítóhalmazbeli elemre jó választ ad. Ezt a hibafüggvény minimumának keresésével éri el.

A Backpropagation korlátai

Természetesen vannak határai a backpropagation algoritmusnak, ezeket tudomásul kell vennünk. Ilyen például, hogy az algoritmus nem feltétlen az abszolút minimumot találja meg, hanem egy lokális minimumban marad. Ezen lehetőség előfordulási esélyének csökkentésére vannak módszerek.

Az is előfordulhat, hogy a háló túltanul. Ez azt jelenti, hogy a tanítóhalmaz elemeire szinte tökéletes válaszokat ad, viszont, ha olyan elemeket mutatunk neki, amikkel még nem találkozott, jelentősen csökken a teljesítménye. Ekkor rossz a háló generalizáló – általánosító – képessége, azaz megszerzett „tudása” túlságosan specifikus a tanítóhalmaz elemeire. Ez a *dropout* módszerrel javítható. [14] E módszer lényege, hogy a futtatás során minden iterációban egyes neuronokat kiveszünk a hálóból. Ezt úgy érjük el, hogy p valószínűséggel a kimenetét manuálisan nullára állítjuk. Így olyan, mint ha mindig egy kicsit más mintát látna a háló. A *dropout* mechanizmusát az alábbi ábra mutatja.



8. ábra: A dropout során p valószínűséggel megtartunk egy neuront, (1-p) nullává tesszük a kimenetét. Ezzel gyakorlatilag kiiktatjuk a hálózatból. [14]

2.4.5 Osztályozás

Világunkban rengeteg objektum vesz minket körül. Ezeket mind osztályokba soroljuk. Az agy alapvető funkciója az osztályozás. Ezt jól mutatja az a tény, hogy már egy hatéves kisgyermek is több mint 10 000 objektum kategóriát képes megkülönböztetni. [15] Az osztályozás azért létfontosságú minden napi életünkben, mert ez az alapja a döntéshozatalainknak. Gondoljunk csak vissza a veszett kutyás példánkra. A döntéshozásban kulcsfontosságú volt a kutya helyes osztályozása, hiszen a válaszreakció ettől függött.

Az osztályozás rendkívül összetett, koránt sem triviális feladat. Hagyományos programozási módszerekkel szinte lehetetlen hatékony osztályozó szoftvert írni összetett feladatokra. Ilyen összetett feladatok lehetek a kézzel írott számjegyek felismerése, arcfelismerés, az önvezető autó projekteknél előkerülő alapprobléma: a járművek felismerése különböző helyzetekben, fényviszonyokban, vagy az orvostudományban életbevágó feladat, a tumoros sejtek felismerése. Nem csak képalapú példák vannak. Ilyen a beszédfelismerés, vagy az agykutatásban nagy kihívást jelentő feladat: az agyi hullámok mintázatainak osztályozása

A neurális hálózatok rendkívül jó eredményeket érnek el osztályozási feladatok megoldásában. Például a kézzel írott számok osztályozásában 99,77 % pontosságot is képes elérni egy neurális hálózat. [16]. A neurális hálózatok a következőképpen valósítják meg az osztályozási feladatokat:

Neurális hálózat, mint osztályozó

A működést kézzel írott számfelismerő példáján keresztül mutatom be, mivel én is ilyet valósítottam meg munkám során.

A háló bemenete egy minta. A háló a döntést úgy valósítja meg, hogy annyi kimenete (y) van, ahány lehetséges osztály (N). Ezekre a kimenetekre alkalmazzuk a *softmax* függvényt [17], aminek képlete a következő.

$$S(y_i) = \frac{e^{y_i}}{\sum_{i=1}^{N} e^{y_i}}$$

Így egy eloszlást kapunk. Az $S(y_i)$ megadja, hogy mennyi a valószínűsége annak, hogy az aktuális bemeneti minta az i-edik osztályba tartozik. Azt az osztályt fogjuk választani az osztályozás során a bemeneti minta osztályának, melynek a valószínűsége ($S(y_i)$) a legnagyobb.

A tanításhoz szükség van még a hibafüggvény meghatározására. Erre a *cross entropy* függvényt használják:

$$D(S, L) = -\sum_{i=1}^{N} L_i \log(S_i)$$

A képletben szereplő L vektor a címke vektort jelöli, mely úgynevezett *one hot encoding* kódolással van reprezentálva. Ez azt jelenti, hogy az annyiadik eleme 1, ahányadik osztályba tartozik. A többi eleme 0.

A *cross entropy* függvény egyfajta távolságot definiál a két vektor között. Értelemszerűen az értéke alacsony lesz, ha az eloszlás közelíti a címke vektor *one hot* eloszlását. Az optimalizáció során ezen eloszlás közelítése a cél. Megjegyzendő, hogy a függvény nem szimmetrikus, tehát az implementáció során az elemek sorrendjére oda kell figyelni.

3 Kevés mintából történő tanulás - One shot learning

3.1 A MINTASZÁM CSÖKKENTÉSÉNEK HATÁSA

Általánosságban igaz a gépi tanulásra, hogy minél nagyobb a tanítóhalmaz elemszáma, tehát azon minták mennyisége, melyeket a hálónak „megmutatunk" a tanítás során, annál jobb lesz a tanítás eredménye. Abból az okból kifolyólag, hogy így sokkal többféle mintával találkozik és így jobban meg tudja találni, rá tud tanulni azokra a tulajdonságokra, amelyek a válaszadás szempontjából lényegesek.

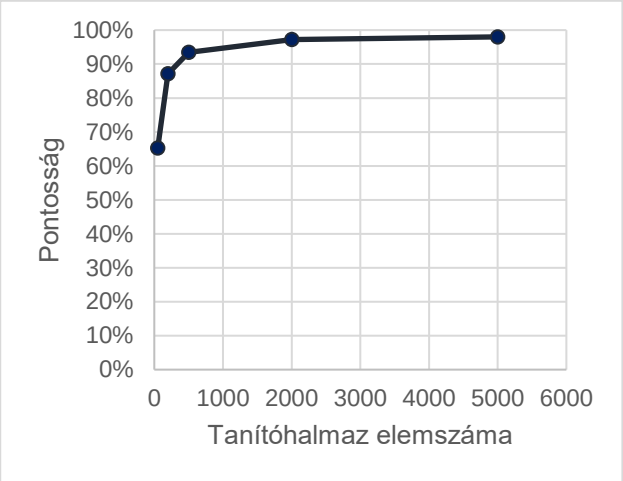
Tehát a mintaszám csökkentésével a független teszthalmazon adott jó válaszok aránya csökken.

Ennek a jelenségnek két oka is lehet. Egyrészt a jellemző tulajdonságokra kevésbé tanul rá a háló, mert az alacsony mintaszám miatt nagy lesz a szórása ezen tulajdonságok előfordulásának. A másik jelenség pedig az, hogy a háló olyan véletlen zajokra tanulhat rá, melyek a tanítóhalmaz elemein éppen egy jellemző tulajdonságnak mutatkoznak, azonban a valóságban zajok.

3.1.1 Kísérlet

Ahogy fentebb említettem a mintaszám csökkentésével független teszthalmazon a jó válaszok aránya jelentősen csökken.

Ezt a jelenséget kísérletileg igazoltam. A vizsgálathoz a fent leírt neurális hálózat-architektúrát használtam. A tanítást 5000 iterációig futtattam. Minden elemszámból 10-10 mérést végeztem, majd ezek átlagát vettem. Ezen átlagokat mutatja a diagramm. Jól látható, hogy a tanítóhalmaz elemszámának csökkentésével a pontosság exponenciálisan lecsökken.



9. ábra: A tanítómintaszám csökkenésével a válaszadás pontossága egy bizonyos küszöb alatt jelentősen lecsökken

3.1.2 A *One shot learning* motivációi

A kevés mintából történő gépi tanításokat *one shot learning*-ként emlegetik. Természetesen sokszor előfordulnak olyan problémák, amikor nem áll rendelkezésünkre sok adat. Ilyenkor kénytelenek vagyunk a kevés adattal is megelégedni, és abból minél jobb eredményt elérni.

Ebben az esetben viszont a tanítás pontossága csökken, ahogy ezt a az előző példában láttuk. Emlékezzünk viszont arra, hogy a gép tanulás ihletője az emberi agy. Ha megfigyeljük az agyműködést, arra a belátásra juthatunk, hogy ez az agynál ilyen drasztikus csökkenés a válaszadás pontosságában nem következik be a mintaszám 50 alá csökkentésével.

Tekintsük az alábbi ábrán látható példát. *Segway*-t mindenki látott már sokat igaz? Nem jelent kihívást a felismerése. A jobb oldali ábrán látható bekeretezett szimbólumból megkeresni a másik példányt már kicsit több időbe telik. Azonban ez a feladat sem jelent nagy kihívást. Egy óvodás is képes megoldani talán a feladatot, még így is, hogy csak egy mintát látott belőle.

10. ábra: Meg tudja találni az azonos osztályú objektumot? [18, p. 2568]

Mivel az agyat is neuronok hálózatával közelítettük, jogos a feltételezés, hogy neurális hálózatok is tudnak még fejlődni a *one shot learning* terén.

A kevés mintából való tanulás teljesítményét úgy tudjuk emelni, hogy a hálózat generalizáló képességét, azaz általánosító képességét próbáljuk növelni. Ez azt jelenti, hogy a hálónak általános tulajdonságokat kell kinyernie a bemenetekből, hogy képes legyen addig nem látott hasonló tesztmintákat helyesen osztályozni. A generalizáló képesség növelése azért is fontos, mert így képes lesz a háló a megtanult tudást más hasonló feladatokba átültetni.

Például egy kutya-macska képek osztályozására előtanított neurális hálózat kutya-ló osztályozására gyorsabban rátanul, ha jó volt a generalizációs képessége.

Jó opció a *one shot* learning javítására az az ötlet, hogy memóriával egészítjük ki a hálózatot. Ez az eljárás nagyon felkapott lett napjainkban. [1] [18] Ez azt jelenti, hogy egy más osztályokon előtanított háló bementére mintákat kötünk be. Az előtanított háló mivel speciálisan o*ne shot learning*-re lett tanítva, ezért hatékonyan tud általános információkat kinyerni a bemenetére érkező kérdéses mintából és asszociálni valamelyik memóriában lévő mintára.

4 Az aktivációs függvény vizsgálata

4.1 MNIST ADATBÁZIS

Vizsgálataim során az MNIST adathalmazt [4] használtam. Ez egy szürkeárnyalatos képeket tartalmazó adatbázis, ami kézzel írott egyjegyű számokat tartalmaz. Mindegyik kép fel van címkézve, hogy hányas számjegyet ábrázolja, így jól használható felügyelt osztályozási tanítási feladat megvalósítására neurális hálózattal.

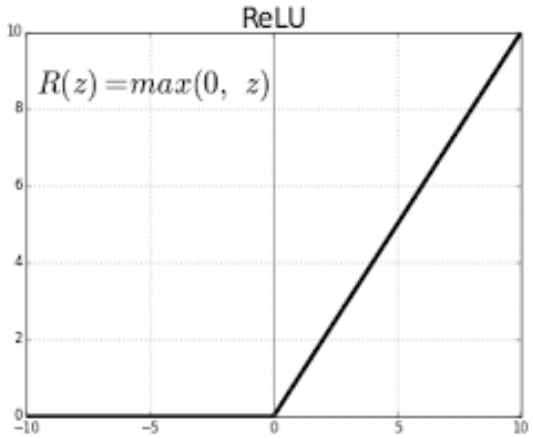
4.2 A VIZSGÁLATOKHOZ HASZNÁLT NEURÁLIS HÁLÓZAT ARCHITEKTÚRÁJA

A neurális hálózat-architektúrát, melyet vizsgálataim során használtam, *tensorflow*-ban implementáltam [19]. A hálózat első rétege konvolúciós. Ez azt jelenti, hogy a bemeneti kép egy 3x3-as ablakának pixelei a neuronok bemenetei. Ezen ablakok átfedésben vannak egymással. Általában élkeresésre tanul rá. A másodok réteg szintén konvolúciós. Ez a *max pooling* algoritmust valósítja meg. Ez azt jelenti, hogy egy adott környezetből a maximális értéket viszi tovább, így csökkenti a dimenziók számát. Ez a két réteg többször ismétlődik. Az utolsó rétegek *fully connected* rétegek. A hálónak tíz kimeneti neuronja van, melyek a tíz számjegyet reprezentálják.

4.3 A GENERALIZÁLÓ KÉPESSÉG JAVÍTÁSA

A *one shot learning* javításához tehát a hálózat generalizálóképességét kell javítanunk. Témavezetőm, Horváth András javaslatára az aktivációs függvényt vettem górcső alá.

A legkedveltebben alkalmazott aktivációs függvény a ReLU, mert a hiba visszaterjesztéshez szükséges deriváltját egyszerű számolni.



11. ábra: A ReLU függvény

A ReLU *tensorflow* implementációja a következő:

```
def ReLU(x):
    x=tf.maximum(0,x)
    return x
```

Ez a függvény nagy számú tanítóminta esetén jól működik. Kevés minta esetén viszont van olyan függvény, amivel jobb lesz a háló generalizáló képessége, így a pontosság is. Ez a sejtés Horváth András témavezetőmtől származott.

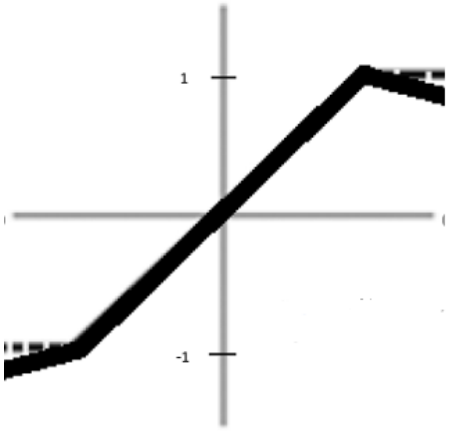
A ReLU esetében a következő jelenség lép fel. Az ilyen függvényt tartalmazó neuron kimenete bármilyen nagy lehet, mivel, ha az aktiváció mértéke nagyon nagy pozitív irányban, akkor a kimenet is nagy lesz. Ez azt eredményezi, hogy kialakulhatnak olyan neuronok, amelyek kimenete a többit elnyomó arányban befolyásolja a háló válaszát. Így, ha például a legtöbb egy osztályba tartozó képen megjelenik valamilyen hasonló zaj (aminek kevés minta esetén nagy a valószínűsége), akkor a háló arra erősen rátanul, és a többi, felismerés szempontjából releváns tulajdonságait a képnek kevésbé veszi számításba. Ennek eredményeként a tanítóhalmaz elemeit jól fel fogja ismerni, viszont a független teszthalmazon gyenge teljesítményt fog nyújtani. Tehát a generalizáló képessége rossz lesz.

Másik probléma a hagyományos ReLU függvénnyel, hogy a 0 alatti intervallumon a deriváltja 0. Így, ha az aktiváció erre az intervallumra esik, akkor nem tud tanulni a neuron, mivel a súlyváltoztatás mértéke mindig 0 lesz. Ezt nevezzük „*dying* ReLU" problémának. Erre megoldást jelent a *leaky* ReLU függvény [20], ami annyiban különbözik az előbbitől, hogy van egy kicsi meredeksége a 0 alatti részének is, ily módon a deriváltja már sehol sem 0.

A ReLU imént említett kedvezőtlen tulajdonságainak kiküszöbölésére Horváth András témavezetőm azt javasolta, hogy limitáljuk a függvény értékkészletét, illetve tegyük bele a *leaky* tulajdonságot. Így a következő függvényt implementáltuk:

```
def limited_leaky_RelLU(x):
    x = tf.maximum(x, -1.0+0.01*(x+1.0))
    x = tf.minimum(x, 1.0-0.01*(x-1.0))
    return x
```

Ily módon a neuron kimenetének lehetséges értékét -1 és 1 közé szorítottuk. Ettől azt vártuk, hogy a háló generalizáló képessége javulni fog, mert a tanítás során „kénytelen lesz" több tulajdonságot is figyelembe venni a döntése meghozásához.



12. ábra: Az általunk konstruált limitált leaky ReLU függvény alakja

4.3.1 Vizsgálat

Azt állítottam fel tehát hipotézisként, hogy kevés minta esetén a fent konstruált limitált ReLU-val jobb pontosság érhető el kevés mintán történő tanítás során, mint a széleskörben használt ReLU függvénnyel.

Első lépésben megállapítottam a mintaszámot, amivel dolgozni fogok. Döntésemben a fent leírt mérés elvégzése segített. Az 50 tanítóhalmaz elemszám mellett döntöttem, mert itt már kellően lecsökken a tanítás pontossága. Alacsonyabb mintaszámnál már bezavar az a tényező, miszerint lehetséges, hogy egy-egy osztályba tartozó képek sokkal kisebb arányban szerepelnek, mint a többi. Ezt a döntést témavezetőm is támogatta.

A pontosság, illetve a futási sebesség javítása érdekében *batch normalizációt* [21] alkalmaztam. Ennek az eljárásnak az a lényege, hogy a *batch*-ek aktivációját az aktivációs függvény nemlineáris szakaszához normalizálja.

A vizsgálathoz a fent leírt neurális hálózat-architektúrát használtam. A tanításokat 10 000 iterációig futtattam, mivel itt a pontosság már nem sokat változik, beáll egy értékre.

Először futtattam egy referencia futást, melyben az 50 mintán a hagyományos ReLU-t használtam aktivációs függvényként. Másodszor ugyanilyen paraméterekkel, csak a limitált *leaky* ReLU-t alkalmazva futtattam a tanítást. Mindkét aktivációs függvénnyel 10 futtatást végeztem, majd ezeket kiátlagoltam. Ezen értékeket az alábbi táblázat mutatja.

ReLU (referencia)	**limitált leaky ReLU**
61.18%	**66.28%**

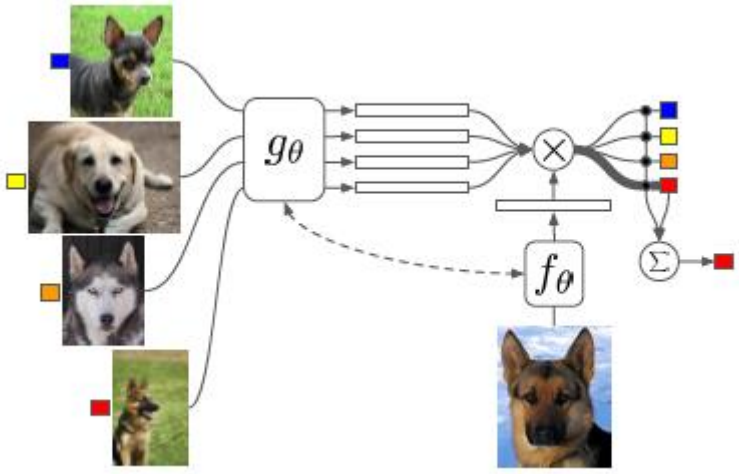
Ahogy az értékekből látszik, az új aktivációs függvény használatával 5,11%-os pontosság növekedést sikerült elérni, mely több mint 8%-os relatív teljesítménynövekedést jelent a referencia futtatáshoz képest. Ezzel sikerült igazolnom a felállított hipotézist.

5 Matching Networks vizsgálata

Az aktivációs függvény vizsgálata után egy másik kutatási területbe fogtam bele a kevés mintából történő tanulás fejlesztése céljából. Egy speciálisan one shot learning-re kifejlesztett neurális hálózat-architektúrát vizsgáltam meg. Ez az architektúra Matching Networks névre hallgat [1]. Az architektúrát rendkívül eredményesen alkalmazták kevés mintából történő felügyelt osztályozási (klasszifikációs) feeladat megoldására.

5.1 A MATCHING NETWORKS MODELLJE

Az hálózat-architektúra döntési folyamata a minták közötti hasonlóság mérésén alapszik. A háló úgy működik, hogy a bementére nem csak a kérdéses minta érkezik, hanem ki van egészítve memóriával, és az is be van kötve a bemenetre. A módszert, hogy memóriával egészítik ki a neurális hálózatot, az utóbbi években sokan mások is sikerrel alkalmazták *one shot learning*-re. [22] [18] [23]A memória a gyakorlatban a Matching Networks esetében egy *support* halmazt jelenti. Ez a halmaz minden osztályból tartalmaz egy vagy több mintát. Az egyszerűség kedvéért vegyük azt az esetet, hogy minden osztályból egy minta képzi a *support* halmazt – tehát annyi a *support* halmaz elemszáma, amennyi osztály van.



13. ábra: A Matching Networks architektúrája [1]

5.1.1 A kimenet generálásának módja

A Matching Networks bemente tehát egy kérdéses (*query*) mintából – melynek nem tudjuk az osztályát (ezt akarjuk osztályozni) – és *N* db. *support* mintából, vagy csak szimplán supportból áll (ahol *N* az osztályok száma). A *support* minták esetében ismeretes előttünk, hogy melyik osztályba tartoznak.

Minden minta egy-egy neurális hálózat bemenete lesz. Ezek a neurális hálózatok az én kutatásom során teljes mértékben megegyeznek. (Speciális esetben egyébként lehet a kérdéses mintához tartozó hálózat különböző a *support* mintákhoz tartozótól Ezért jelzi az ábra is különbözőknek: f és g neurális hálózatoknak).

Ezek a neurális hálózatok úgynevezett kódoló hálók. Ez azt jelenti, hogy a kimeneti rétegük kis elemszámú a bemenethez képest. Ez az én architektúrám esetében 16-os elemszám volt. Így minden neurális hálózat kimenete egy 16 hosszú vektor lett.

A következő lépésben a *cosinus distance* függvényt használja a Matching Networks. Ez a függvény minden *support mintához tartozó vektor*nak egyfajta távolságát adja vissza a *kérdéses mintához tartozó vektor*hoz képest. Tehát a függvény minden *(support vektor – kérdéses vektor)* párhoz rendel egy számot, mely a minták egymástól való távolságát reprezentálja. A *cosinus distance* kimenetére a softmax függvényt kell meghívni, hogy eloszlást kapjunk. Ez lesz a háló kimente.

A *outputot* úgy értelmezhetjük, hogy minden *support* mintához tartozó kimenet azt adja meg, hogy mennyi annak a valószínűsége, hogy a kérdéses minta az adott *support* minta osztályába tartozik. A háló válaszaként azt az osztályt fogadjuk el, amelyiknek ez a valószínűség a legnagyobb.

5.1.2 Tanítás

A hibafüggvény generálásához a *cross entropy* függvényt (ld. bővebben a 2.4.5 Osztályozás részben) használja a Matching Networks. A tanítóalgoritmus megválasztható. Kutatásom során az *Adam optimizer*-t [24] használtam, mely a sztochasztikus *gradient descent* algoritmus egy megvalósítása.

Ahogy már említettem, a neurális hálózatok a *suppotok* és a *query* esetében is megegyeznek. Ez azt jelenti, hogy a tanítás során is csak ezt az egy hálót optimalizáljuk és ezt használjuk fel minden minta esetében.

A Matching Networks nagy előnye, hogy asszociatív memóriaként tud működni. [1] Ez azt jelenti, hogy a betanított hálózat képes olyan osztályokból származó mintákat is jó hatékonysággal felismerni, amilyeneket nem figyelt meg egyáltalán a tanítás során. Ez úgy valósul meg, hogy a hálózat lényegében nem az egyes osztályokra tanul rá, hanem arra, hogy hogyan gyűjtse ki a mintákból azokat a tulajdonságokat melyek a döntéshozás szempontjából lényegesek. Természetesen csak akkor működik ez a jelenség hatékonyan, ha a tanított osztályok és a tesztelő osztályok hasonló tulajdonságok szerint osztályozhatók. Például kutyafajok – mint a fenti képen látható példában – fejalakjuk, színük, fülméretük stb. szerint. Másik jó példa a kézzel írott számok, melyeket a kutatásom során használtam.

5.2 A SUPPORT HALMAZ VÁLASZTÁSÁNAK VIZSGÁLATA

A *support* halmaz központi jelentőségű a Matching Networks architektúrában. Ezt azért feltételezem mert a halmaz elemi fogják az egyes osztályokat reprezentálni. Ha a választott *support* nem elég általános, akkor könnyen előfordulhat, hogy az adott *support* minta osztályába tartozó *query* mintát rosszul osztályozza a háló.

A support mintákat a Matching Networks attyjai [1] véletlenszerűen választják. Azt állítottam fel hipotézisként, hogy lehet találni a véletlenszerűen választott support halmaznál jobb halmazt, mellyel a háló pontossága nagyobb lesz.

5.2.1 Implementáció

Tensorflow [19] segítségével implementáltam a fent leírt Matching Networks architektúrát. A neurális hálózat, melyet minden support és a query esetében használtam, a „4.2. A vizsgálatokhoz használt neurális hálózatok architektúrája” részben leírthoz hasonló, azzal a különbséggel, hogy a kimeneti réteg 16 neuronból áll, mivel a vektorokat, ahogy már említettm, 16 hosszúnak választottam. 32-es batch méretet

használtam. Tanítóhalmazként az MNIST adatbázis [4] felcímkézett kézzel írott számokat ábrázoló képeit használtam.

5.2.2 A kutatás

A kutatásom célja az volt, hogy megvizsgáljam, hogyan függ a Matching Networks architektúra pontossága a support halmaz választásától, illetve keresni egy olyan *support* halmaz választási technikát (ha van), ami jobb, mint a véletlenszerű választás.

Az kutatásomhoz az MNIST adatbázis három **részhalmazát** hoztam létre:

- Tanítóhalmazt

  2, 3, 4-es számjegyeket tartalmaz, elemszáma 30 000
- Optimalizációs halmazt

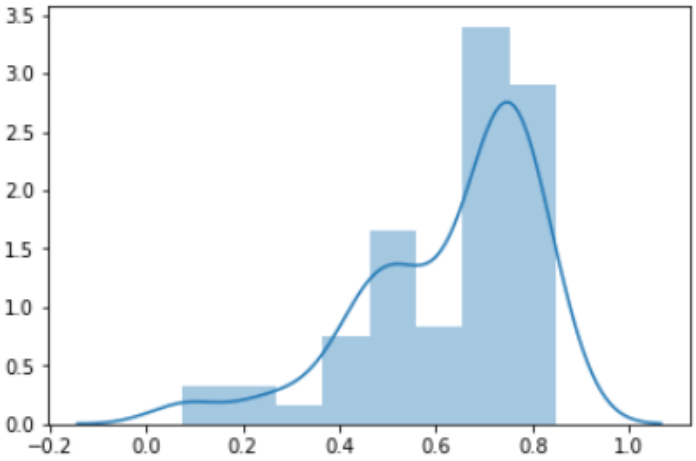
  7, 8, 9-es számjegyeket tartalmaz, mindegyikből 5-5 darabot
- Validációs halmazt

  7, 8, 9-es számjegyeket tartalmaz, mindegyikből 1000-1000 darabot

Első lépésben a hálózat betanítottam a tanítóhalmazon. Mind a *supportokat*, mind a *query*-ket a tanítóhalmazból választottam. A tanítást 5000 iterációig futtattam, mivel itt a pontosság értéke már kellően változatlan volt. A modellt elmentettem, hiszen ezen már nem fogok változtatni, ezt fogom használni a kutatásom során.

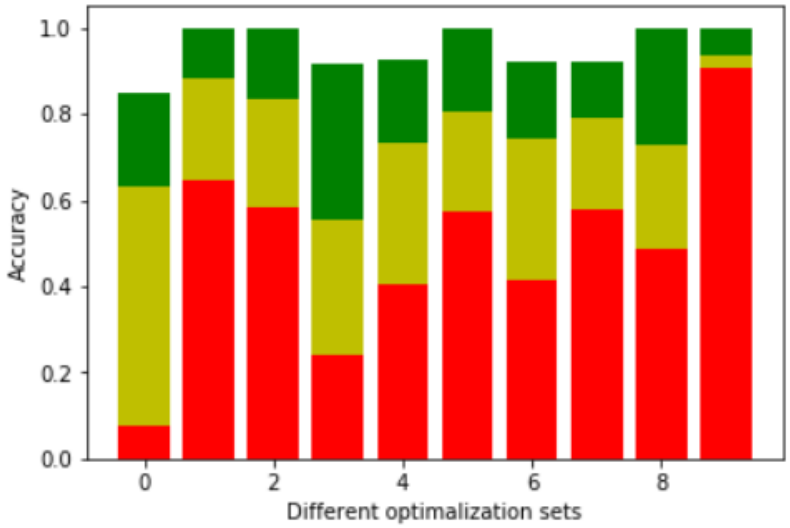
Ez után a betanított hálót a validációs halmazon teszteltem (tehát ebből választottam a query-t) oly módon, hogy a *supportokat* mindig az optimalizációs halmazból vettem.

Minden optimalizációs halmazbeli *support* kombinációt (összesen $5^3 = 125$) külön leteszteltem. Minden ilyen tesztet 100 iterációig futtattam a 32-es batch méret mellett (a *suport-batch*-eket ekkor mindig ugyanazzal a kombinációval töltöttem fel), tehát ez 3200 mérésnek felel meg. A kombinációkhoz tartozó pontosság értékeket kiátlagoltam és elmentettem egy 125 méretű *'accuracies_of_sets'* nevű listába. Ahogy arra számítottam is előzetesen, a lista meglehetősen változó pontosság értékeket mutatott. A jelenség az alábbi ábrán megfigyelhető.



14. ábra: Az különböző support halmazok pontosságértékeinek eloszlása. Látható, hogy a halmaz választásától erősen függ a válaszadás pontossága.

Az iménti mérést elvégeztem még néhányszor más optimalizációs halmaz választásával. Minden *'accuracies_of_sets'* listának vettem a maximumát, az álagát és a minimumát. Ezen értékeket mutatja az alábbi ábra.



15. ábra: A különböző optimalizációs halmazokból vett support halmazok pontosságértékeinek optimális (zöld sáv teteje), átlagos (sárga sáv teteje) és minimális (piros sáv teteje) értékei.

Az diagrammon jól látható, hogy mekkora eltérések lehetnek a pontosságban. Látható az is, hogy a agy eltérés nem törvényszerű (ld. az utolsó mérés a diagrammon), de az eredményből látszik: jelentős esély van rá.

A *support* halmazok véletlenszerű választásakor a maximum és a minimum közötti érték bármelyike kijöhet. Tehát szerencsétlen esetben akár a legrosszabb kombináció is. Az adatsorból az következik, hogy ennél az étéknél átlagosan 46 %-kal kedvezőbb kombináció létezik.

Ha statisztikus szemmel, a várható érték, azaz az átlagos pontosságértékű kombináció generálására számítunk, akkor az optimális ennél várhatóan 19 %-kal jobb lesz.

Láthatjuk tehát, hogy a hipotézisem beigazolódott. Jelentősen növelhető a válaszadás pontossága, ha nem véletlen generáljuk a *support* halmazt, hanem megkeressük azt, amelyikkel a legjobban működik a háló.

Megjegyzendő, hogy valószínűleg, ha több azonos osztálybeli *support* mintát is megengednénk az architektúrában, akkor valószínűleg kisebb lenne a pontosságok szórása. Ráadásul a kombinációk kipróbálása is jóval több időt venne igénybe, mert *support* mintaszám növekedésével a kombinációk száma robbanásszerűen növekszik.

A másik oldalon az is igaz, hogy a *support* halmaz elemszámának növelésével a háló sokkal számításigényesebb lesz, ezért a válaszadás sebessége lassul. Az általam alkalmazott optimalizáció ezzel szemben *training* időben zajlik, am általában nem jelent hátrányt, hiszen a felhasználás során már a háló nem tanul – legalábbis vannak olyan felhasználások, amikor nem.

Összességében az mondható, hogy ha az általam bemutatott kombinációs optimalizációt erősen ajánlott használni akkor, ha

- a kéznél lévő mintahalmaz, melyet *support* halmazként használhatunk nagyon kicsi
- vagy ha fontos, hogy gyors legyen a döntéshozás
- és ha a *training* idő hosszúsága nem jelent problémát.

6 Összefoglalás

Kutatásom ihletője az emberi tanulás volt, mégpedig az emberi agy azon képessége, hogy egy osztályozási feladat elvégzésére már kevés példa felvillantása után is képes. Bár az ember saját képességeinek reprezentálására egy gép segítségével teljes mértékben nem lehet képes, de közelítheti azt. Erre tettem kísérletet én is most oly módon, hogy az elmúlt időszakban nagy sikereket elérő gépi tanulást alkalmaztam kevés minta esetén.

A kutatásom első felében azt a hipotézist állítottam fel, hogy ha kicseréljük az ReLU aktivációs függvényt egy *leaky*, limitált változatára, javulni fog a háló generalizáló képessége, és így a válaszadásának pontossága is. Hipotézisem igaznak bizonyult a vizsgálatom eredményei alapján. Több mint 8%-os relatív teljesítménynövekedést sikerült elérnem. Ez a kis változtatás sokat javíthat a kevés minta alapján tanító algoritmusok teljesítményén.

Ezek utána a Matching Networks neurális hálózat-architektúrát vizsgáltam meg. Sikerült kimutatnom, hogy a hálózat *support* halmazának választása jelentősen befolyásolj a háló válaszadásának pontosságát. Megmutattam, hogy az *archotekzúra* tervezői által javasolt véletlenszerű supportgenerálás helyett a kombinációk tesztelésével várhatóan körülbelül 20 %-kal jobb eredmény érhető el, mely optimalizáció kifejezetten ajánlott, ha kevés minta áll rendelkezésünkre, hiszen ekkor nem tudunk nagy elemszámú supporthalmazt generálni.

Ennek a munkának egy továbbfejlesztése lehet a *supportok* kiválasztása helyett egy szintetikusan előállított *support* halmaz használata, mellyel a háló még optimálisabban működhetne.

A jövőben érdemes lehet megvizsgálni a kutatott módszereket más adatbázison is. Érdekes lehet, hogy milyen eredményt hoznak többcsatornás képi bemeneteken, vagy akár akusztikus mintán.

7 Köszönetnyilvánítás

Nagy köszönettel tartozom Horváth András témavezetőmnek a sok támogatásért, a kitartó magyarázásokért, és hogy mindig kész volt segíteni, amikor szükségem volt rá. Hálásan köszönöm Czenthe Lilinek, hogy mindvégig mellettem volt és tartotta bennem a lelket, kitartóan végigkísért a nehéz szakaszokon át is. Végül, de nem utolsó sorban köszönöm édesapámnak, Ábrahám Péternek a segítő tanácsokat, támogatást, a nyelvtani lektorálást.

8 Irodalomjegyzék

[1] Vinyals, Blundell, Lillicrap, Kavukcuoglu és Wierstra, „Matching networks for one shot learning," *arXiv:1606.04080v2 [cs.LG] 29 Dec 2017,* 2016.

[2] Russell, Norvig és .. Fordította: Édelkraut et al., Mesterséges Intelligencia, Hungarian Translation Panem Könyvkiadó, 2005.

[3] L. Hunyadi , „Bayesi gondolkodás a statisztikában," *Statisztikai Szemle,* %1. kötet89., %1. szám10-11., pp. 1151-1171., 2011.

[4] Y. LeCun, L. Bottou, Y. Bengio és P. and Haffner, „Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* %1. kötet86., p. 2278–2324., 1998..

[5] A. W. Harley, „An Interactive Node-Link Visualization of Convolutional Neural Networks," *ISCV,* pp. 867-877, 2015.

[6] Kawaguchi és Kiyoshi, A multithreaded software model for backpropagation neural network applications, ETD Collection for University of Texas, El Paso, 2000.

[7] L. Widrow, „30 years adaptiv neural networkd: perceptron, Madaline, and backpropagation?," *Proceedings of IEEE,* %1. kötet78, %1. szám9, pp. 1415-1442, 1990.

[8] J. Hámori, „Mit tud az emberi agy?," *Mindentudás Egyeteme,* pp. 113-131, 2003.

[9] J. H. Byrne, „Introduction to Neurons and Neuronal Networks," McGovern Medical School at UTHealth, Department of Neurobiology and Anatomy, 1997. [Online]. Available: https://nba.uth.tmc.edu/neuroscience/s1/introduction.html. [Hozzáférés dátuma: 20 05 1996].

[10] Blum és Li, „Approximation theory and feedforward networks," *Neural Networks,* pp. 511-515, 1991.

[11] A. S. Walia, „Aktivation functions and it's types - Which is better?," 29 05 2017. [Online]. Available: https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f. [Hozzáférés dátuma: 18 05 2019].

[12] J. Park és I. W. Sandberg, „Universal Approximation Using Radial-Basis-Function Networks," *Neural Computation,* %1. kötet3, %1. szám2, pp. 246 - 257, 1991.

[13] Y. le Cun, „A Theoratical Framework for Back-Propagation," 1988.

[14] Srivastava, Hinton és A. Krizhevsky et al., „Dropout: A Simple Way to Prevent Neural Networks from," *Journal of Machine Learning Research,* %1. kötet15, pp. 1929-1958, 2014.

[15] L. Fei-Fei, R. Fergus és P. Perona, „One-Shot Learning of Object Categories," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE,* %1. kötet28., %1. szám4., pp. 594-611., 2006..

[16] L. Wan, M. Zeiler, S. Zhang, Y. LeCun és R. Fergus, „Regularization of Neural Network using DropConnect," *International Conference on Machine Learning*, 2013.

[17] Shie, Dori és Reuven, „The cross entropy method for classification," *ICML '05 Proceedings of the 22nd international conference on Machine learning*, pp. 561-568, 2005.

[18] A. Santoro, S. Bartunov, M. Botvinic, D. Wierstra és T. Lillicrap, „One-shot Learning with Memory-Augmented Neural Networks," 2016.

[19] M. Abadi, A. Agarwal, P. Barham és .. et al, „TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: software available from tensorflow.org..

[20] B. Xu, N. Wang, T. Chen és M. Li, „Empirical Evaluation of Rectified Activations in Convolutional Network," arXiv:1505.00853, 2015.

[21] S. Ioffe és C. Szegedy, „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," arXiv:1502.03167, 2015.

[22] D. Bahdanau, K. Cho és a. Y. Bengio, „Neural machine translation by jointly learning to align and translate," *ICLR*, 2014.

[23] J. Weston, S. Chopra és A. Bordes, „Memory networks," *ICLR*, 2014.

[24] D. P. Kingma és J. Ba, „Adam: A Method for Stochastic Optimization," 2015.

9 Mellékletek

A tanítóhalmaz elemszámának és a pontosság összefüggésének kimutatásához végzett mérésem. A pontosság értékek 10-10 futtatás átlagából származnak.

Tanítóhalmaz elemszáma	**Pontosság [%]**
5000	98.00
2000	97.18
500	93.47
200	87.08
50	65.25

Az aktivációs függvény vizsgálatának melléklete

Az aktivációs függvények vizsgálatához végzett méréseim, melyekből az átlagok származnak. (61.18% a ReLU esetében és 66.28% a limitált *leaky* ReLU esetében.)

ReLU	limitált leaky ReLU
71.09%	72.87%
57.32%	61.21%
62.83%	63.45%
57.90%	65.33%
57.94%	65.28%
59.29%	67.24%
56.85%	69.24%
67.40%	65.58%
56.81%	61.91%
64.32%	70.71%

A Matching Networks vizsgálatának melléklete

Az különböző optimalizációs halmazok különböző support halmazainak pontosságadataink egyes mutatói, melyeket az oszlopdiagrammon ábrázoltam.

Az optimalizációs halmaz indexe	**A legkedvezőtlenebb support halmazok pontosságértékei**	**Az átlagos support halmazok pontosságértékei**	**Az legkedvezőbb support halmazok pontosságértékei**
0	0.0759375	0.6297575	0.8478125
1	0.6484375	0.8813349	1.0
2	0.585	0.83501756	1.0
3	0.2428125	0.55321	0.91625
4	0.4053125	0.733255	0.9275
5	0.5746875	0.803745	1.0
6	0.4125	0.7408	0.92375
7	0.57625	0.79267	0.923125
8	0.488125	0.727845	1.0
9	0.9075	0.9371725	1.0

Az alábbi ábra a 1-9 indexű optimalizációs halmazokhoz tartozó mérések különböző supporthalmazinak pontosságértékeinek eloszlását mutatja. A 0. indexű a Maching Networks részben található. Megjegyzendő, hogy a függvények x tartománya nem mindig a [0, 1] intervallum.