Mesterséges neurális hálózatok matematikai alapjai

2019.

Szakdolgozat

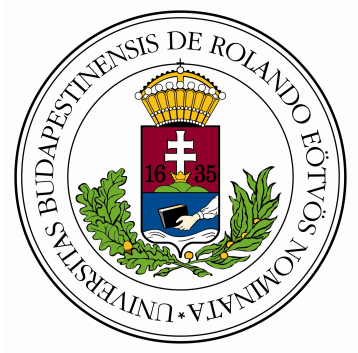
Borza Marcell

Alkalmazott matematikus szakirány

Témavezető:

Kiss Attila

ELTE TTK Számítógéptudományi Tanszék



Eötvös Loránd Tudományegyetem

Természettudományi Kar

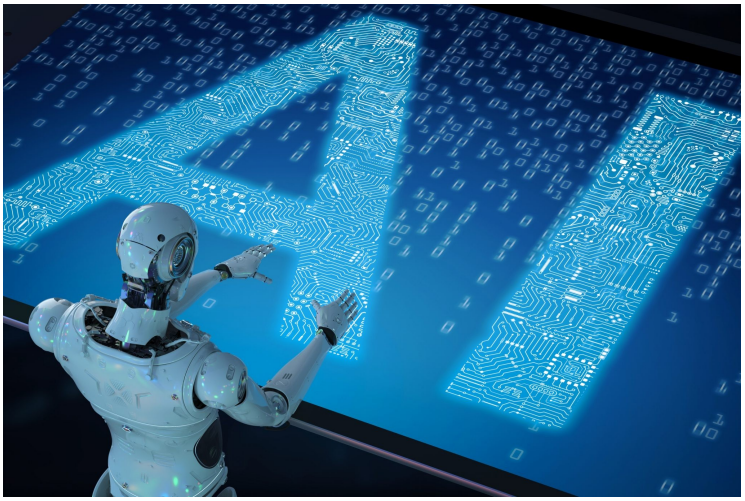
Köszönetnyilvánítás

Szeretném megköszönni témavezetőmnek, Kiss Attilának, hogy felkeltette érdeklődésemet a téma iránt, szakértő iránymutatással látott el és válaszolt a felmerülő kérdeseimre. Köszönet illeti minden jelenlegi és volt matematika tanáromat, akik mélyítették tudásomat, különös tekintenttel volt középiskolás tanáraimra, Dr. Gaál Istvánné tanárnőre és Kovács Péter tanár úrra. Emellett hálával tartozom édesanyámnak, aki elindított az úton és mindenben támogatott.

Tartalomjegyzék

1. Bevezetés

Napjaink egyik legjobban csengő hívószava a mesterséges intelligencia (AI), amely sokak számára csillogó robotokat, másoknak hihetetlen varázslatot jelent, ám a valóság, sokak számára csalódásként, ennél sokkal tudományosabb. A mesterséges



intelligencia a XX. század második felében indult hódító útjára és már ekkor nagyon nagy jövöt jósoltak neki a tudósok. Marvin Minsky egészen odáig ment, hogy a következőt jósolta 1970-ben: „Három-nyolc éven belül lesznek olyan gépeink, amelyek egy átlagos ember általános intelligenciájával rendelkeznek. Úgy értem, olyan gépek, amelyek képesek lesznek Shakespeare-t olvasni, politikai játszmákba bonyolódni, viccet mesélni, harcolni. Ettől fogva a gépek fantasztikus sebességgel kezdik el tanítani önmagukat. Néhány hónapon belül elérik a zsenik szintjét, majd ismét néhány hónap múlva erejük immár beláthatatlanná válik.” [9]. Ma már tudjuk, hogy ez nem következett be, mégis mindennapjainkat egyre növekvő mértékben határozza meg az AI. Ebben nagy szerepet játszik az 1990-es években megjelenő egyre nagyobb adatmennyiség (big data), amely segítségével a szakemberek már hatékonyan tudták betanítani és tesztelni gépeiket. Ma már számos példát találhatunk arra, hogy a számítógépek legyőzik az embert olyan dolgokban, amelyekről sokáig azt hittük csak az emberek képesek rá, például képesek arcokat, hangokat, érzelmeket

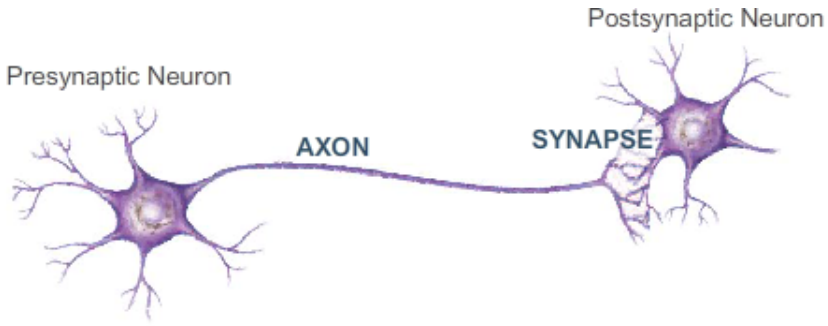
felismerni vagy olyan játékokat eredményesen játszani, amelyekhez nem elég pusztán az óriási számítási kapacitás. Erre kiváló példa a go játék és az erre kifejlesztett gép, az AlphaGo, amely nem hogy legyőzte az aktuális go világbajnokot, de olyan stratégiát alkalmazott, amelyet még a legnagyobb mesterek sem láttak. Ez a húzás azóta beépült a humán go játékosok játékéba is, így ebben az esetben mindenképp elmondható, hogy miután az ember megtanította a gépet az ember tanult a géptől. Hasonló elsöprő győzelmek születtek pókerben, sakkban vagy az intuíciót igénylő Jeopardy! nevű amerikai tévés vetélkedőben, ahol a játékosoknak a válaszok alapján kell kitalálniuk a kérdést.

A mesterséges intelligencia és a gépi tanulás egyik fontos szelete a mesterséges neurális hálózatok elmélete, ennek a témának az alapjaival foglalkozom a dolgozatomban. A 2. fejezetben bemutatom az emberi agy működéséből származó alapötletet és ennek a megfogását a matematika eszközeivel, a fejezet végén pedig ennek működését egy egyszerű példán. A 3. fejezetben megvizsgálom néhány, a való életben is használt fajtáját a mesterséges neurális hálóknak. A 4. fejezetben approximációs szempontból vizsgálom a hálók képességeit, az 5. fejezetet pedig a gyakorlati megvalósításnak szentelem. A téma szegényes magyar szakirodalommal rendelkezik, nincs kialakult magyar nyelvezete. Dolgozatomban többször szerepel a háló kifejezés, amely minden esetben hálózatot jelent.

2. Egyszerű mesterséges neurális hálók

2.1. Biológiai indíttatás

Az idegrendszer legkisebb alkotóelemei az idegsejtek, latin nevén neuronok, melyekről sokáig azt hitték a kutatók, hogy hozzávetőleg 100 millárd darab található egy ember testében. Friss eredmények szerint ez a szám 86 milliárd körüli egy átlagos egészséges ember esetében, megtalálhatóak az agyban, a gerincvelőben és a ganglionnak nevezett idegdúcokban. Ezek a sejtek alkalmasak elektromos töltés



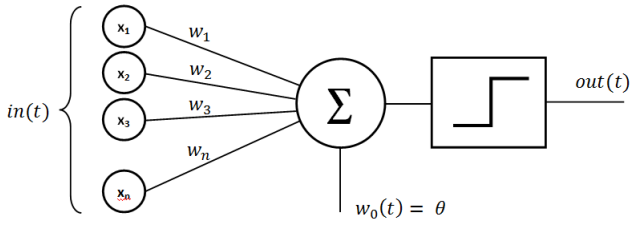
1. ábra. Neuron felépítése

vezetésére és tárolására. A neuronok alakja és mérete nagyon változatos, de vannak közös jellemzőik. Mindannyian rendelkeznek sejttesttel és nyúlványokkal, előbbi végzi a tényleges információfeldolgozást, utóbbiak pedig az ingerületet szállítják. A nyúlványok közül egy kitüntetett, amely az ingerületet a sejttesttől távolodó irányba szállítja, az úgynevezett axon. Egy neuron más neuronok axonjaihoz dendriteken keresztül kapcsolódik, ezért a dendritek tekinthetőek a neuron bemeneteinek. Az a pont ahol a két nyúlvány kapcsolódik a szinapszis, az itt létrejövő szinaptikus kapcsolat erőssége határozza meg az emberi agy tanulását. Az idegsejtek óriási hálózatot alkotnak, minden egyes neuron közvetlen kapcsolatban áll több ezer másikkal, így lehetővé téve az információ áramoltatását egy időben több irányba. Ennek következtében egyszerre nagyon sok neuron aktív. Azt, hogy egy adott pillanatban épp mely neuronok aktívak aktivitási mintának nevezzük, amely értel-

mezhető egy adott mentális állapotnak. Az elektromos impulzus gyorsan áthalad ezen a sűrűn átszőtt hálózaton, eljutva nagyon sok neuronhoz megváltoztatva ezen mentális állapotot. Ezen működés sokkal finomabb egy egyszerű bináris aktivitási mintázatnál, ahol csak az számít, hogy egy csomópont épp be van-e kapcsolva vagy nincs, itt a kapcsolat erősségén van a hangsúly, ebben rejlik az emberi agy tanulási képességének ereje. Ha az aktivitási mintákat agyi tudatállapotként értelmezzük, akkor a tudás úgy definiálható, mint egyik mentális állapotból egy másikba történő impulzusáramlás. Ez azt jelenti, hogy a tanulás során új szinapszisok jönnek létre, amelyek vagy erősítik, vagy gyengítik az eddigieket. Ezekből a megállapításokból adódóan bármilyen hosszútávú tudás létrejötte az agy anatómiájának változásával jár. Agyi sérülések esetén bizonyos neuronok közötti szinaptikus kapcsolatok megváltozhatnak vagy eltűnhetnek, ez vezethez ahhoz, hogy a sérültnek újra kell tanulnia bizonyos, korábban már megtanult dolgokat. Súlyosabb esetben a betegnek olyan alapvető mechanizmusokat is újra kell tanulnia mint a járás vagy a beszéd. Az emberi agy szinaptikus összeköttetésével és tanulásával kapcsolatban érdemes megemlíteni Donald O. Hebb kanadai neuropszichológus nevét, aki a XX. században lefektette a téma alapjait [8]. Ezen ismeretek lehetőséget adnak ennek a nagyon összetett rendszernek bizonyos szintű leutánzására.

2.2. Átültetés a matematikába

A mesterséges neurális hálók (ANN-artificial neural networks) az agyhoz hasonlóan csomópontokból (amire itt is használják a neuron szót) és ezek közötti súlyozott kapcsolatokból állnak. Az ANN modellek közül a legegyszerűbb az úgynevezett egyszerű perceptron, amely Frank Rosenblatt nevéhez fűződik 1958-ból [7]. Az egyszerű perceptron kétféle csomópontból áll: néhány bemeneti és egyetlen kimeneti csomópontból. Ebben a modellben minden egyes bemeneti csomópont súlyozott kapcsolatban áll a kimeneti csomóponttal, amely először képzi bemeneteinek egy súlyozott összegét, majd ennek eredményére alkalmaz egy aktivációs függvényt melyekről később esik szó. A súlyozás reprezentálja a korábban említett szinap-



2. ábra. Egyszerű perceptron felépítése

tikus kapcsolatot. A modell alkalmas például két lineárisan szeparálható halmaz szétválasztására egy alkalmas szeparáló hipersíkkal.

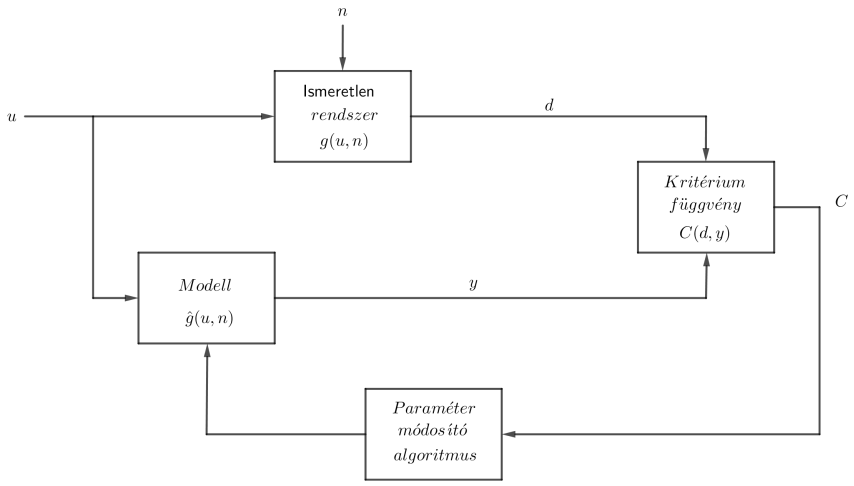
2.1. Definíció (Lináris szeparálhatóság)**.** *N dimenziós pontoknak egy* $H = H_1 \cup H_2$ *halmazát lineárisan szeparálhatónak nevezzük, ha létezik olyan* $N - 1$ *dimenziós s hipersík, melyre s nem tartalmaz H-beli pontot és egy* $h \in H$ *pont pontosan akkor esik a hipersík kijelölt oldalára, ha* $h \in H_1$*. Ekkor az s hipersíkot szeparáló hipersíknak nevezzük.*

Ha a bemenet nem lineárisan szeparálható, akkor a tökéletes kettéosztás ezzel a módszerrel nem tehető meg, ebben az esetben olyan megoldás keresése elképzelhető ami a mintapontok többségét jól osztályozza. A lineárisan nem szeparálható esetre egy lehetséges megoldás a későbbiekben bemutatott többrétegű perceptron. A perceptron tanítása a csomópontok közötti összefüggések erősségenek változtatása valamilyen szabály szerint a kívánt állapotig. A kezdeti súlyok általában tetszőlegesek. Ez a tanítás lehet felügyelt vagy nem felügyelt, ezzel a két fogalommal a következő két alfejezetben foglalkozom.

2.2.1. Felügyelt tanítás

Felügyelt tanítás esetében szükségünk van (x, d) párokra, úgynevezett tanulóesetekre, ahol x a bemenő attribútumok halmaza, d pedig az ismert, az adott tanulóesethez tartozó helyes kimenet. Ezután alkalmazzuk a modellt a tanulóesetek bemenő attribútumaira és vizsgáljuk az y predikált kimenet és d viszonyát. A tanulás folyamata a

következőképpen szemléltethető. A feladatunk a $d = g(u, n)$ tényleges leképezéshez



3. ábra. Tanulás folyamata

egy $y = \hat{g}(w, u)$ változtatható paraméterű modell illesztése, ezt mutatja a 3. ábra. Ebben az esetben az illesztés minimumkeresést jelent, ahol az $\varepsilon(d, y)$ hibafüggvény minimumát keressük a **w** súlyozás függvényében. Leggyakrabban d és y átlagos négyzetes eltérést választjuk hibafüggvénynek:

$$\varepsilon(d, y) = \sum (d_i - y_i)^2,$$

A zaj jelenléte miatt ezt gyakran várható értékként értelmezzük:

$$\overline{\varepsilon}(d, y) = E[(d - y)^T (d - y)] = E[\sum_j (d_j - y_j)^2].$$

A súlyok javítása általában gradiens alapú módszerek alapján történik, melyekről később esik szó. A tanítás során a tanulóadatokat általában többször áramoltatjuk át a hálózaton, minden tanulóeset egyszeri átáramoltatását egy epochnak nevezzük. A tanítás megtervezésekor az epochok számát is meg kell határoznunk, választhatunk fix értéket vagy bizonyos átlagos hibahatár elérésig is áramoltathatjuk a tanulóada-

tokat vagy kombinálhatjuk is a két gondolatot, így akkor áll le az áramoltatás, ha a kettő közül az egyiket elértük.

2.2.2. Nem felügyelt tanítás

Nem felügyelt tanítás esetén cimkézetlen tanulóesetekkel dolgozunk, nem várunk el előre meghatározott kimenetet. A hálózat ekkor nem kap semmilyen közvetlen visszajelzést működésének milyenségéről. Ebből is látszik, hogy teljesen más típusú problémákra várhatunk így megoldást. Címkék hiányában azt tudjuk vizsgálni, hogy az adatok milyen mintázatokat követnek, mennyire hasonlítanak egymáshoz páronként vagy csoportonként,vannak-e sűrűbb területek és így tovább. Tipikus probléma aminél felhasználható a nem felügyelt tanítás a klaszterezés, emellett jól használható az előfeldolgozás során, akár egy felügyelt tanítással megoldható problémánál az adatok előkészítésére.

2.2.3. A jóság mérése

Ahhoz, hogy ellenőrizni tudjuk azt, hogy a modell hogy teljesít új adatokon szükségünk van számára ismeretlen inputokra, teszt esetkre. A megfelelő következtetések levonása érdekében ennek mérete többszöröse a tanulóesetek számának. A gyakorlatban ha egy adott adathalmaz alapján szeretnénk tanítani modellünket, akkor az összes rendelkezésre álló adat körülbelül negyedét használjuk tanulóesetként a maradékon pedig tesztelünk. Ahhoz hogy a tesztelés eredményét jobban megfoghatóvá tudjuk tenni szükségünk van a helyességet mérő számokra. Ilyen például a pontosság és a precizitás.

2.2. Definíció (Pontosság)**.** *A 4. ábra jelöléseit használva:*

$$Pontosság = \frac{VP + VN}{VP + ÁP + VN + ÁN}$$

2.3. Definíció (Precizitás)**.** *A 4. ábra jelöléseit használva:*

$$Precizitás = \frac{VP}{VP + ÁP}$$

		vizsgálat eredménye		
		negatív (–)	pozitív (+)	
valós állapot (arany standard)	egészséges (–)	valós negatív (VN)	álpozitív (ÁP)	→ specificitás VN/(VN+ÁP)
	beteg (+)	álnegatív (ÁN)	valós pozitív (VP)	→ szenzitivitás VP/(ÁN+VP)
		↓ szegregancia VN/(VN+ÁN)	↓ relevancia VP/(ÁP+VP)	pontosság

4. ábra. Jóságot leíró néhány mennyiség

2.2.4. Az egyszerű perceptron működése

Az egyszerű perceptron felépítésének megértéséhez Fazekas István könyve [1] lehet segítségünkre. Tekintsük a korábban említett perceptron modellt, ennek a csomópontjait két osztályba soroltuk: bemeneti és kimeneti. Ez a kétféle a neuron teljesen másért felelős, míg a bemeneti neuron csak továbbítja a kapott adatot bármiféle transzformáció nélkül, a kimeneti neuron egy erős matematikai eszköz, képzi a bemeneteinek és egy torzítási tényezőnek egy lineáris kombinációját és ennek eredményére alkalmaz egy nem feltétlenül lineáris aktivációs függvényt. Matematikailag ez a következő alakban írható:

$$v(n) = b(n) + \sum_{i=1}^{m} w_i(n)x_i(n)$$

$$y(n) = \varphi(v(n)) = \varphi\Big(\sum_{i=0}^{m} w_i(n)x_i(n)\Big)$$

itt $b(n)$ a torzítás, $v(n)$ jelöli a neuron által képzett lineáris kombináció eredményét, w_i, x_i az i-edik súly, illetve bemeneti komponens, φ pedig az aktivációs vagy transzfer függvény. Az egyszerűbb tárgyalhatóság kedvéért gyakran használják a $w_0(n) = 1$ és $x_0(n) = b(n)$ jelöléseket. Ezeket felhasználva felírhatjuk a következő rövid összefüggéseket:

$$v(n) = \sum_{i=0}^{m} w_i(n)x_i(n) = \mathbf{w}^T(n)\mathbf{x}(n)$$
$$y(n) = \varphi\big(\mathbf{w}^T(n)\mathbf{x}(n)\big)$$

A tényleges adattranszformációt végző neuront processzáló neuronnak hívjuk, jelen esetben ezek a kimeneti neuronok. Ezek után definiálhatjuk a hálózat négyzetes hibáját:

$$\varepsilon(n) = (d(n) - y(n))^2.$$

Célunk az ezt minimalizáló súlyok megtalálása a hálózat ellenőrzött tanításával. Ehhez képeznünk kell a hibafüggvény súlyok szerinti parciális deriváltjait és negatív gradiens irányba kell lépnük a súlyokkal.

$$w_i(n+1) = w_i(n) + \lambda \frac{\partial \varepsilon(n)}{\partial w_i}$$

A képletbeli λ az úgynevezett tanulási tényező, ami azt határozza meg, hogy mekkorát lépünk a negatív gradiens által meghatározott irányba. Nagy λ értékek esetén gyorsabban haladunk a hibafüggvény minimumhelyéhez, ám könnyen túl is léphetünk rajta, ami oszcillációhoz vezethet, kis λ-kra viszont a konvergencia sebessége lassul. Egy fix λ helyett gyakran használnak időben változó, általában csökkenő $\lambda(n)$ értékeket , így a minimumhelyhez közeledve csökken a lépéshossz. A gyakorlatban többféle aktivációs függvényt szoktak alkalmazni, ilyen például a szigmoid, szignum, tanh, küszöb, ReLU (rektifikált lineáris egység) vagy a szakaszonként lineáris.

$$szigmoid(x) = \frac{1}{1 + e^{-ax}}$$

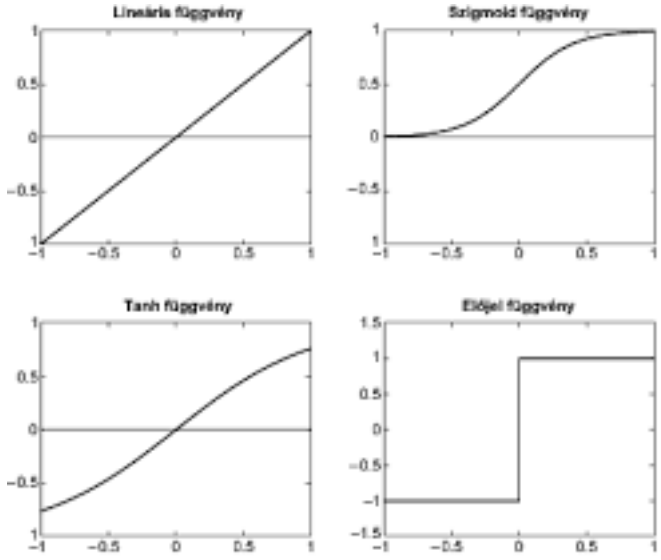
$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

$$szaklin(x) = \begin{cases} 0, & \text{ha } x < -0,5 \\ x + 0,5, & \text{ha } -0,5 \leq x < 0,5 \\ 1, & \text{ha } x \geq 0,5 \end{cases}$$

$$sgn(x) = \begin{cases} -1, & \text{ha } x < 0 \\ 0, & \text{ha } x = 0 \\ 1, & \text{ha } x > 0 \end{cases}$$

$$\text{küszöb}(x) = \begin{cases} 1, & \text{ha } x \geq 0 \\ 0, & \text{ha } x < 0 \end{cases}$$

$$ReLU(x) = \max(0, x)$$



5. ábra. Néhány lehetséges aktivációs függvény

2.2.5. Két lineárisan szeparálható halmaz szétválasztása perceptronnal

Ebben a részben megmutatjuk, hogy két olyan halmaz esetén melyek m dimenziós pontokat tartalmaznak és található hozzájuk olyan eggyel kevesebb, tehát $m-1$ dimenziós hipersík, amely a két halmaz pontjait tökéletesen elválasztja egymástól, a szeparáló hipersík minden esetben meghatározható perceptronnal véges sok lépés alatt. A két halmazt jelölje H_1 és H_2. A bizonyítás során különbséget kell tennünk az eredeti és a kibővített bemeneti vektorok között, ezért jelölje a továbbiakban $\hat{\mathbf{x}}$ az eredeti, míg $\mathbf{x}$ a bővített bemeneti vektort.

$$\mathbf{x}(n) = (1, \hat{x}_1(n), \dots \hat{x}_m(n))^T = (x_0(n), x_1(n), \dots x_m(n))^T$$

Ezek után legyen $\mathbf{t}$ a keresett hipersík normálvektora, amely a H_1-et tartalmazó féltérbe mutat, $\mathbf{a}$ pedig a hipersík egy rögzített pontja. Ekkor egy $\mathbf{x}$ mintapont akkor tartozik a H_1 ponthalmazba, ha $(\hat{x}-a)$ hegyesszöget zár be t-vel: $(\hat{x}-a)^T t > 0$, tehát $\hat{x}^T t - a^T t > 0$. Legyen $\mathbf{w} = (-a^T t, t_1, \dots, t_m)^T$, ez lesz a perceptron súlyvektora, $\mathbf{x}$ pedig a bemenete. Ezesetben $\mathbf{x}$ akkor és csak akkor tartozik H_1-be, ha $\mathbf{x}^T \mathbf{w} > 0$, hasonlóan akkor és csak akkor tartozik H_2-be, ha $\mathbf{x}^T \mathbf{w} < 0$. Tehát az átalakítások után az eggyel magasabb dimenziós térben elég az origón átmenő hipersíkokkal foglalkoznunk. Az itt megkeresett m dimenziós hipersík első koordinátájának elhagyásával nyert külső vetület lesz a számunkra megfelelő szeparáló hipersík. A perceptron tanításánál soros tanítást alkalmazunk, ez azt jelenti, hogy a tanító pontokat egyesével dolgozzuk fel és minden egyes tanító pont átáramoltatása után kiértékeljük a kapott kimenetet és esetleges súlyváltoztatásokat eszközölünk. A tanítás alapelve, hogy megvizsgáljuk az aktuális pont az aktuális hipersík mely oldalára esik, ha az ismert helyes kimenet szerintire, akkor tovább haladunk a következő pontra, a hipersíkot nem változtatjuk, ha rossz oldalára, akkor a hipersíkot a rossz pont felé forgatjuk. Ha a teljes pontkészlet egyszeri átáramoltatása során már nem történik korrekció, akkor megállunk, hiszen az aktuális hipersík megfelelő. Könnyen látható, hogy a kapott szeparáló hipersík nem egyértelmű. A következőkben megmutatjuk, hogy korlátos, pozitív margóval szétválasztható tanítópontok esetében ezen eljárás

megfelelő alkalmazásával a perceptron véges sok lépésben talál egy megfelelően szeparáló hipersíkot, ezt a következő tétel formájában fogalmazhatjuk meg [1].

2.1. Tétel (Perceptron konvergencia tétel)**.** *Tegyük fel, hogy A_1 és A_2 úgy szeparálható lineárisan, hogy $\|\mathbf{w}^T\mathbf{x}(n)\| \geq \delta > 0$ és $\|\mathbf{x}(n)\|^2 \leq R < \infty$ teljesül minden tanító pontra. Legyen a tanulási paraméter $\eta > 0$ állandó. Ekkor a perceptron algoritmusa véges sok lépésben véget ér a tanító pontokat helyesen szeparáló hipersíknál.*

Bizonyítás. A bizonyítás során jelöljük $\mathbf{w}$-vel a keresett súlyvektort és $\mathbf{w}(i)$-vel ennek i-edik közelítését. Rossz helyre besorolt pont esetén az új súlyok meghatározása a fentebb említettek szerint a következőképpen zajlik:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - sgn\big(\mathbf{w}(n)^T\mathbf{x}(n)\big)\eta\mathbf{x}(n) \tag{1}$$

Be fogjuk látni, hogy csak véges sokszor állhat ez fenn. Az előző egyenlet mindkét oldalának normanégyzetét véve:

$$\begin{aligned}\|\mathbf{w}(n+1)\|^2 = \|\mathbf{w}(n) - sgn\big(\mathbf{w}(n)^T\mathbf{x}(n)\big)\eta\mathbf{x}(n)\|^2 = \\ = \|\mathbf{w}(n)\|^2 + \eta^2\|\mathbf{x}(\mathbf{n})\|^2 - 2sgn\big(\mathbf{w}(n)^T\mathbf{x}(n)\big)\eta\mathbf{w}(n)^T\mathbf{x}(n) \leq \\ \leq \|\mathbf{w}(n)\|^2 + \eta^2\|\mathbf{x}(\mathbf{n})\|^2\end{aligned}$$

Itt felhasználtuk, hogy η pozitív $\mathbf{w}(n)\cdot\mathbf{x}(n)$ pedig a saját előjelével van megszorozva, így a kivonás jel utáni tag biztosan pozitív. Megvizsgálva a becslés elejét és végét azt láthatjuk, hogy a súlyvektor $(n+1)$-edik közelítése felülről becsülhető az eggyel korábbi közelítés segítségével. Ezt visszafejtve az első közelítésig és felhasználva, hogy $\|\mathbf{x}(n)\|^2 \leq R$:

$$\begin{aligned}\|\mathbf{w}(n+1)\|^2 \leq \|\mathbf{w}(n)\|^2 + \eta^2\|\mathbf{x}(\mathbf{n})\|^2 \leq \cdots \leq \|\mathbf{w}(1)\|^2 + \eta^2(\|\mathbf{x}(1)\|^2 + \ldots \|\mathbf{x}(n)\|^2) \leq \\ \leq \|\mathbf{w}(1)\|^2 + \eta^2 nR.\end{aligned}$$

Most (1) mindkét oldalát $\mathbf{w}^T$-tal szorozva:

$$\mathbf{w}^T\mathbf{w}(n+1) = \mathbf{w}^T\left[\mathbf{w}(n) - sgn\big(\mathbf{w}(n)^T\mathbf{x}(n)\big)\eta\mathbf{x}(n)\right] =$$
$$= \mathbf{w}^T\mathbf{w}(n) - \eta sgn\big(\mathbf{w}(n)^T\mathbf{x}(n)\big)\mathbf{w}^T\mathbf{x}(n) \geq \cdots \geq \mathbf{w}^T\mathbf{w}(1) + n\delta\eta.$$

A becslésnél felhasználtuk, hogy:

$$-\eta \cdot sgn\{\mathbf{w}(n)^T\mathbf{x}(n)\}\mathbf{w}^T\mathbf{x}(n) = \eta|\mathbf{w}^T\mathbf{x}(n)|,$$

mivel korrekciós pontról van szó, tehát $\mathbf{w}(n)^T\mathbf{x}(n)$ és $\mathbf{w}^T\mathbf{x}(n)$ előjele ellentétes ($\mathbf{w}$ esetén jó, míg $\mathbf{w}(n)$ esetén rossz oldalra esik). Továbbá felhaszáltuk, hogy

$$\eta|\mathbf{w}^T\mathbf{x}(n)| \geq \eta\delta,$$

ezután pedig visszafejtettük a rekurziót az első közelítésig. Elég nagy n-ekre az egyenlőtlenséglánc végén kapott kifejezés pozitív, hiszen δ és η is pozitív. Ekkor felhasználva a $|a^T b| \leq \|a\|\|b\|$ Cauchy-egyenlőtlenséget:

$$\|\mathbf{w}(n+1)\| \geq \frac{|\mathbf{w}(n+1)^T\mathbf{w}|^2}{\|\mathbf{w}\|^2}$$

Ebbe behelyettesítve a kapott eredményeket:

$$\|\mathbf{w}(1)\|^2 + \eta^2 nR \geq \frac{(\mathbf{w}^T\mathbf{w}(1) + n\delta\eta)^2}{\|\mathbf{w}\|^2} \tag{2}$$

Itt minden tag pozitív a bal oldal n-ben elsőfokú a jobb oldal pedig másodfokú, ami nem állhat fenn végtelen sok n esetén, tehát csak véges sokszor sorol be egy pontot rossz helyre a perceptron a futása alatt. Ezek közül van egy utolsó, ami után már minden tanulópontot a neki megfelelő halmazba osztályoz, tehát helyesen szeparál. Ha a kiinduló állapotban $\mathbf{w}(1) = 0$, akkor (2) szerint:

$$n \leq \frac{\|\mathbf{w}\|^2 R}{\delta^2}.$$

Ez azt jelenti, hogy a tanítókészletet újra és újra átáramoltatva ennél több rosszul szeparált pontot nem találunk. □

Az előző tétel Rosenblatt és Novikoff nevéhez fűződik [10]. Megkaptuk tehát, hogy az egyszerű perceptron tökéletesen szét tudja választani a pontokat lineárisan szeparálható esetben, ám ez egy elég erős feltétel. Általános esetben a perceptron osztályozó képességét leíró mennyiség a kapacitás.

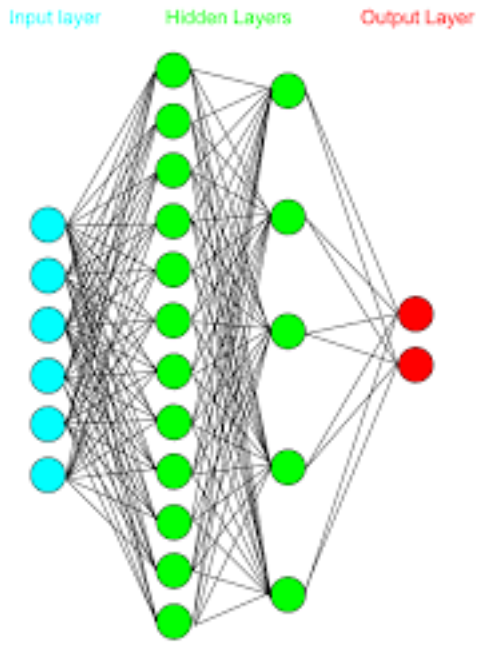
2.4. Definíció (Kapacitás)**.** *A perceptron kapacitását $C(P, N)$-nel jelöljük és a következőképpen számolhatjuk ki:*

$$C(P,N) = \begin{cases} 1, & ha\ P \leq N \\ \frac{2}{2^P} \sum_{i=0}^{N-1} \binom{P-1}{i}, & ha\ P > N \end{cases}$$

ahol P a véletlenszerűen kiválasztott pontok száma, N pedig a dimenziószám.

3. Többrétegű neurális hálózatok

A modellek fejlődésében az egyszerű perceptron csak a kezdet volt, természetesen egy mesterséges neurális háló általában ennél sokkal bonyolultabb. A neuronokat rétegekbe szervezhetjük a modellben betöltött szerepük és helyük szerint, például az előbb említett egyszerű perceptron áll egy bemeneti és egy kimeneti rétegből. Ha ezek közé még további réteget vagy rétegeket helyezünk többrétegű perceptront (MLP) kapunk, így növelve a modell bonyolultságát és ezzel együtt felhasználhatóságát. Az ilyen beszúrt csomópontokat és rétegeket rejtett csomópontoknak,



6. ábra. Többrétegű perceptron felépítése

illetve rejtett rétegeknek nevezzük.Attól függően hogy egy adott réteg neuronjai mely más rétegek neuronjaival állnak kapcsolatban beszélhetünk előrecsatolt és visszacsatolt mesterséges neurális hálókról. Ha egy réteg neuronjai csak a későbbi rétegek neuronjaihoz kapcsolódnak, akkor előrecsatolt neurális hálóról beszélünk, ha ilyen megkötést nem teszünk, tehát lehet kapcsolat visszafelé is vagy egy réte-

gen belüli két neuron között is, akkor visszacsatolt hálóról. Az előrecsatolt neurális háló egy hierarchikus modell, ahol minden réteg egy lineáris transzformáció eredményére alkalmaz egy nemlineáris aktivációs függényt. A k-adik réteg bemenetét a $k-1$-edik réteg kimenete jelenti. Ebben az esetben ha N darab D dimenziós tanító bemenetünk van, akkor ezt reprezentálhatjuk egy $X \in \mathbf{R}^{N\times D}$ mátrixszal, ahol minden egyes sor egy tanítópontnak, minden oszlop pedig egy attribútumnak felel meg. Jelölje $X_k \in \mathbf{R}^{N\times d_k}$ a k-adik réteg kimenetét $d_0 = N$ és legyen $X_0 = X$, jelölje továbbá $W^k \in \mathbf{R}^{d_{k-1}\times d_k}$ a k-adik réteg által megvalósított lineáris transzformáció mátrixát. A k-adik réteg aktivációs függvényét ψ_k-val jelölve a k-adik réteg kimenete a következő formában adható meg:

$$X_k = \psi_k(X_{k-1}W^k).$$

Ezek alapján a K rétegű (előrecsatolt) neurális háló kimenetét a következőképpen számolhatjuk ki:

$$\Phi(X, W^1, \ldots, W^K) = \psi_K(\psi_{K-1}(\ldots\psi_2(\psi_1(XW^1)W^2)\ldots W^{K-1})W^K).$$

A Φ kimenet tehát egy $N \times d_K$ dimenziós mátrix [5].

A többrétegű perceptronon kívül vannak más típusú többrétegű mesterséges neurális hálók is, melyek közül néhány a következő alfejezetekben kerül bemutatásra, ezek a radiális bázisfüggvényes hálózatok, a CMAC hálózatok, a szekvenciális hálók és a konvolúciós neurális hálók, ám először tekintsük át kicsit részletesebben a többrétegű perceptront.

3.1. A többrétegű perceptron

A többrétegű perceptron egy előrecsatolt neurális hálózat, amely rétegekbe szervezett neuronokból áll. Háromféle réteget különböztetünk meg: bemeneti, kimeneti és rejtett réteget, melyek közül az első kettőből mindig egyetlen egy található a hálózatban, rejtett rétegből tetszőleges számú lehet. A hálózat mélységét a rejtett rétegek száma adja meg. A perceptronhoz hasonlóan minden neuronhoz tartozik egy aktivációs függvény, egy súlyvektor és egy torzítási tényező. Fontos, hogy ezek az egyes neuronokhoz tartoznak, külön-külön változtathatóak. Egy réteg neuronjainak kimenete a következő réteg neuronjainak bemenete, így az információ a bemeneti réteg irányából rétegről rétegre halad egészen a kimeneti rétegig, amely az egyszerű perceptronnal ellentétben akár több kimeneti neuronból is állhat.

3.1.1. Hibavisszaterjesztés

A többrétegű perceptron tanításánál nehézséget jelent, hogy egy neuron kimenetét nem mindig tudjuk közvetlenül minősíteni, hiszen csak a kimeneti réteg esetében van elvárásunk a kimenetre, így csak ezeknél a neuronoknál tudjuk közvetlenül értelmezni a hibát. Erre jelent megoldást a hiba visszaterjesztéses módszer, amely kihasználja hogy egy adott neuron bemenete az előző réteg kimenete így a rétegeken visszafelé haladva tudja minősíteni a a neuronok kimenetét és meghatározi a korrekciót [1]. Az előzőekhez hasonlóan célunk a kimenet négyzetes hibájának minimalizálása. Legyen K a kimeneti réteg ekkor az n-edik tanítóponthoz tartozó hiba:

$$\varepsilon(n) = \sum_{j \in K} (d_j(n) - y_j(n))^2 = \sum_{j \in K} e_j^2(n),$$

ahol $d_i(n)$, $y_i(n)$ az elvárt, illetve a predikált kimenet, $e_i(n)$ pedig a hiba az i-edik neuron és az n-edik tanulópont esetén. Jelölje a továbbiakban w_{ji} a súlyt az i-edik neuronból a j-edik neuronba ($w_{j0} = b_j$ a j-edik neuron torzítása), v_j a j-edik neuron által képzett súlyozott összeg (az aktivációs függvény bemenete), φ_j a j-edik neuron aktivációs függvénye. Ezek alapján a j-edik neuron kimenete a következőképpen

határozható meg:

$$y_j(n) = \varphi_j(v_j(n)).$$

A súlyok változtatását itt is gradiens alapúan, a delta szabály szerint végezzük, a nehézség a rejtett neuronok irány menti deriváltjainak meghatározása.

$$w_{ji}(n+1) = w_{ji}(n) - \lambda \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)}$$
$$\Delta w_{ji}(n) = -\lambda \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)},$$

ahol λ pozitív tanulási tényező. Az itt szereplő gradiensek direkt kiszámítása általában bonyolult, viszont rekurzívan könnyen kiszámolhatóak felhasználva, hogy a kimeneti réteg parciális deriváltjai közvetlenül számolhatóak. A vizsgált iránymenti derivált a következőképpen írható fel a láncszabály szerint:

$$\frac{\partial \varepsilon(n)}{\partial w_{ij}(n)} = \frac{\partial \varepsilon}{\partial e_j} \cdot \frac{\partial e_j}{\partial y_j} \cdot \frac{\partial y_j}{\partial v_j} \cdot \frac{\partial v_j}{\partial w_{ji}}.$$

Végezzük a következő helyettesítéseket, melyek helyessége könnyen igazolható:

$$\frac{\partial \varepsilon(n)}{\partial e_j(n)} = e_j(n)$$
$$\frac{\partial e_j(n)}{\partial y_j(n)} = \varphi_j'(v_j(n))$$
$$\frac{\partial y_j(n)}{\partial v_j(n)} = -1$$
$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n).$$

Ezt felhasználva:

$$\Delta w_{ji}(n) = \lambda e_j(n) \varphi_j'(v_j(n)) y_i(n).$$

Kimeneti neuron esetén készen is vagyunk ebből kiszámolható a gradiens, ám rejtett neuron esetén $e_j(n)$ értékét nem ismerjük, ezért további számításokra van szükség.

Jelöljük $\delta_j(n)$-nel a lokális gradienst, tehát:

$$\delta_j(n) = -\frac{\partial \varepsilon(n)}{\partial v_j(n)} = e_j(n)\varphi_j'(v_j(n)).$$

Ezt behelyettesítve:

$$\Delta w_{ji}(n) = \lambda \delta_j y_i(n).$$

Rejtett neuron esetén ugyan δ_j közvetlenül nem számítható ki, ám rekurzívan a következőképpen igen:

$$\delta_j(n) = -\frac{\partial \varepsilon(n)}{\partial v_j(n)} = -\sum_{k \in \text{j utáni réteg}} \frac{\partial \varepsilon(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial v_j(n)} = \\ = \sum_{k \in \text{j utáni réteg}} \delta_k(n) w_{kj}(n) \varphi_j'(v_j(n)).$$

Ezzel a képlettel már minden súly szerinti parciális derivált kiszámolható és a súlymódosítás elvégezhető hiszen a kimeneti réteg δ értékei közvetlenül meghatározhatóak. A hiba visszaterjesztéssel tehát csökken a hálózat hibája, ám ez nullára nem feltétlenül csökkenthető, ezért gondoskodnunk kell valamilyen megállási feltételről. Tipikusan ez a gradiens nagyságával kapcsolatos, mivel kis gradiens esetén már kis mértékben változnak a súlyok. A gradiens alapú szélsőértékkereső eljárások hátránya, hogy érzékenyek a kezdőpont megválasztására és könnyen olyan lokális minimumba ragadhatnak, ami nem globális minimum, emellett hátrány az oszcillációra való hajlam. Az oszcilláció elkerülésének egyik lehetséges módja a momentum módszer, amelyet a delta szabály módosításával nyerhetünk.

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \lambda \delta_j(n) y_i(n) \\ \Delta w_{ji}(0) = 0,$$

ahol α a korábbi lépés bevonásának mértékét leíró, momentum konstans nevű mennyiség. A differencia egyenletet megoldva:

$$\Delta w_{ji}(n) = \lambda \sum_{t=1}^{n} \alpha^{n-t} \delta_j(t) y_i(t) = -\lambda \sum_{t=1}^{n} \alpha^{n-t} \frac{\partial \varepsilon(t)}{\partial w_{ji}(t)}$$

A momentum módszer az oszcilláció kivédése mellett a konvergenciát is gyorsítja.

3.2. Önnormalizáló neurális hálózatok

Ebben a részben szintén egy előrecsatolt neurális háló típussal foglalkozom, az önnormalizáló neurális hálózatokkal (Self-normalizing Neural Networks, SNNs) [6]. A háló megalkotásának célja a nulla várható értékű, egy szórású kimenet elérése, ezáltal segítve a robusztus tanulást sok rétegen. Ezek a hálók speciális aktivációs függvénnyel (SELU) rendelkeznek, amely önszabályozó képességet biztosít, szabályozva a réteg kimenetének szórását, ezzel meggátolva a gradiens felrobbanását vagy eltűnését. Az aktivációs függvénnyel később foglalkozom részletesebben. A neurális hálózat bemenetére valószínűségi változóként is tekinthetünk, ekkor két egymás utáni réteget vizsgálva legyen a két réteg közötti súlymátrix $\mathbf{W}$, a korábbi réteg kimenete az $\mathbf{x}$ vektor, az aktivációs függvény pedig f. Ezekkel a jelölésekkel $\mathbf{z} = \mathbf{W}\mathbf{x}$ az aktivációs függvény bemenete, $\mathbf{y} = f(\mathbf{z})$ pedig a későbbi réteg kimenete, ami így szintén egy valószínűségi változó. A továbbiakban tegyük fel hogy minden x_i-re $\mu = \mathbf{E}(x_i)$ és $\nu = \mathrm{Var}(x_i)$, tehát az $\mathbf{x}$ koordinátái azonos várható értékűek és szórásúak. Legyen továbbá a későbbi réteg egy y kimenetére $\overline{\mu} = \mathbf{E}(y)$ és $\overline{\nu} = \mathrm{Var}(y)$. A későbbi réteg neuronjaira értelmezzünk a hozzá tartozó $\mathbf{w}$ súlyvektor koordinátáinak összegét és második momentumát: $\omega = \sum_{i=1}^{n} w_i, \tau = \sum_{i=1}^{n} w_i^2$. Ezek után definálják a az önnormalizáló neurális hálózatokat.

3.1. Definíció (Önnormalizáló neurális hálózat)**.** *Egy neurális hálózat önnormalizáló neurális hálózat, ha létezik hozzá egy olyan* $g : \Omega \mapsto \Omega$ $(\Omega = \{(\mu, \nu) : \mu \in [\mu_{min}, \mu_{max}], \nu \in [\nu_{min}, \nu_{max}]\})$ *minden* y *neuronkimenethez, amely a korábbi réteg várható értékét és szórását képzi le az ezt követő réteg várható értékére és szórására,*

úgy hogy ennek a g-nek van fix pontja Ω-ban, $g(\Omega) \subset \Omega$ és minden Ω-beli pontra g-t iterálva az eredmény a fix ponthoz tart.

Egy réteg kimenetét normalizáltnak tekintjük, ha a várható érték és a szórás egy előre meghatározott intervallumba esik minden minta esetén. Ha létezik az előbb definiált g és $\mathbf{x}$ várható értéke és szórása már a megadott intervallumban van, akkor $\mathbf{y}$ várható értéke és szórása is oda fog esni, mivel a két érték konvergál ezen az intervallumon ha g-t alkalmazzuk, így a normalizáltság tranzitív a rétegek közt [6].

3.2.1. Megvalósítás

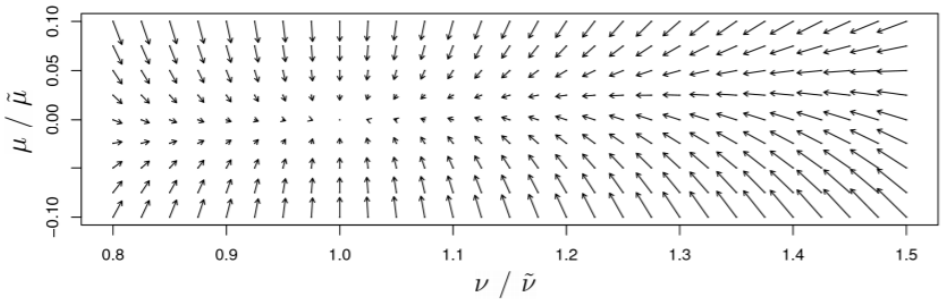
Ahhoz, hogy a fentebb definiált g létezését biztosítsuk mindössze két dologban van szabadságunk: az aktivációs függvényben és a kezdő súlyokban. A kezdő súlyoknál minden neuron esetében a $\omega = 0$ és $\tau = 1$ választás a megfelelő. Aktivációs függvény esetében a következőkben definiált SELU függvények biztosítják az SNN számára a megfeleő g létezését.

$$selu(x) = \begin{cases} \lambda x, & \text{ha } x > 0 \\ \lambda(\alpha e^x - \alpha), & \text{ha } x \leq 0 \end{cases}$$

A képletben α és λ szabad paraméter. Ha egy réteg x_i kimeneteleiről most azt tételezzük fel, hogy függetlenek, ám azonos várható értékkel és szórással rendelkeznek, akkor az ezt követő réteg egy neuronjának aktivációs függvényének z bemenetéről a következők mondhatóak el:

- $z = \mathbf{w}^T \mathbf{x}$
- $\mathbf{E}(z) = \sum_{i=1}^{n} w_i \mathbf{E}(x_i) = \mu\omega$
- $Var(z) = Var(\sum_{i=1}^{n} w_i x_i) = \nu\tau$

Ekkor a centrális határeloszlás tétel szerint n növelésével z eloszlása normális eloszláshoz tart: $z \sim \mathcal{N}(\mu\omega, \sqrt{\nu\tau})$, $p_N(z; \mu\omega, \sqrt{\nu\tau})$ sűrűségfüggvénnyel. Ezek alapján



7. ábra. Normalzált esetben ($\omega = 0$, $\tau = 1$) g fixpontja $(0, 1)$

a későbbi réteg kimenetének várható értéke és szórása következőképp határozható meg:

$$\overline{\mu}(\mu, \omega, \nu, \tau) = \int_{-\infty}^{+\infty} selu(z) p_N(z; \mu\omega, \sqrt{\nu\tau}) dz \tag{3}$$

$$\overline{\nu}(\mu, \omega, \nu, \tau) = \int_{-\infty}^{+\infty} selu(z)^2 p_N(z; \mu\omega, \sqrt{\nu\tau}) dz - \overline{\mu}^2 \tag{4}$$

hiszen $\overline{\mu} = \mathbf{E}(y)$ és $\overline{\nu} = Var(y)$. Ezek az integrálok analitikusan kiszámíthatóak, így megadható a $g : (\mu, \nu) \mapsto (\overline{\mu}, \overline{\nu})$ függvény.

3.2.2. Normalizált súlyok

Tegyük fel, hogy a súlyvektor normalizált, $\omega = 0$, $\tau = 1$. Adott (μ, ν) esetén meg tudjuk oldani (3)-t és (4)-t, tehát meg tudjuk az aktivációs függvény α és λ szabad paramétereit. Válasszunk $(\mu, \nu) = (0, 1)$ szerint ekkor megoldva a $\mu = \overline{\mu} = 0$, $\nu = \overline{\nu} = 1$ fixpontegyenletet: $\alpha_{01} \approx 1,6733$ és $\lambda_{01} \approx 1,0507$ (az alsó index azt mutatja mely μ és ν értekhez tartozó α, illetve λ értékekről van szó). Ezek után a Banach-féle fixpont tétel szerint g Jacobi-mátrixának normáját kell megvizsgálnunk, ha ez kisebb mint 1, akkor létezik g-nek stabil fixpontja a zárt halmazban. A Jacobi-mátrix spekrálnormáját kiszámítva, azt kapjuk, hogy legnagyobb sajátértéke $0,7877 < 1$, tehát g egy kontrakció és $(0, 1)$ stabil fixpontja, ezt mutatja a 7.ábra [6].

3.2.3. Nem normalizált súlyok

Az előző részben megvizsgáltuk, hogy normalizált súlyok esetén hogyan viselkedik a háló, ám a tanulás folyamán nem garantálható végig a normalizáltság. A következő tétel azt mondja ki, hogy SELU aktivációs függvény és $\alpha = \alpha_{01}$, $\lambda = \lambda_{01}$ választás mellett, ha (ω, τ) elég közel van $(0, 1)$-hez, akkor g-nek még mindig van fix pontja és ez közel van $(0, 1)$-hez. Ez azt jelenti, hogy a normalizáltságnál gyengébb feltétel is elég a fixpont létezéséhez [6].

3.1. Tétel (Stabil fixpont létezése)**.** *Legyen $\alpha = \alpha_{01}$ és $\lambda = \lambda_{01}$, továbbá tegyük fel, hogy $\mu \in [-0.1, 0.1]$, $\omega \in [-0.1, 0.1]$, $\nu \in [0.8, 1.5]$ és $\tau \in [0.95, 1.1]$, ezek Descartes-szorzatát jelölje Ω. Legyen g értelmezési tartománya Ω. Ekkor $\omega = 0$, $\tau = 1$ esetben $(0, 1)$ g-nek fixpontja, míg más értelmezési tartománybeli (ω, τ) párra g-nek van $(\hat{\mu}, \hat{\nu})$ fixpont úgy, hogy $\hat{\mu} \in [-0.03106, 0.06773]$, $\hat{\nu} \in [0.80009, 1.48617]$. Továbbá g-t iterálva minden értelmezési tartománybeli (μ, ν) pont ezen fixponthoz konvergál.*

A tétel bizonyítása a normalizált esethez hasonlóan a Banach fixpont tétel segítségével zajlik. A tétel leírja a háló viselkedését abban az esetben, ha a (μ, ν) pár Ω-beli, ezt azonban nem mindig tudjuk garantálni, ezért a következőkben azt az esetet vizsgáljuk amikor (μ, ν) Ω-n kívüli. Ekkor különösen fontos ν nagysága, magas értéke a gradiens elszállásának, túl alacsony értéke pedig a gradiens eltűnésének felel meg, ezért ν megfelelő korlátok közé szorításával elkerülhető ezen két jelenség. Erről szól a következő két tétel [6].

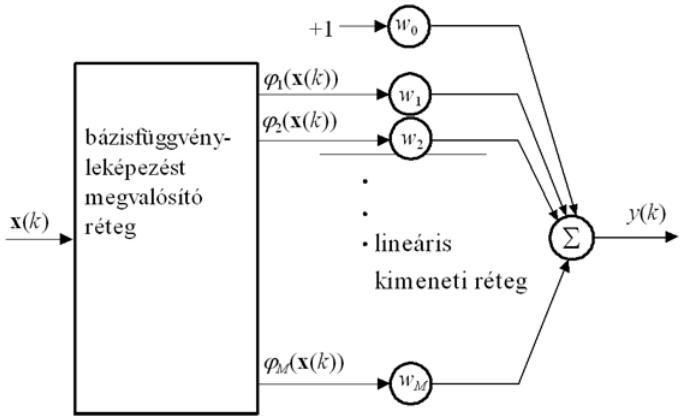
3.2. Tétel (Csökkenő ν)**.** *Legyen $\lambda = \lambda_{01}$ és $\alpha = \alpha_{01}$ az előzőek szerint és legyen Ω^+: $-1 \leq \mu \leq 1$, $-0.1 \leq \omega \leq 0.1$, $3 \leq \nu \leq 16$ és $0.8 \leq \tau \leq 1.25$ ekkor a (4) szerint származtatott $\overline{\nu}(\mu, \omega, \nu, \tau, \lambda, \alpha)$-ra teljesül hogy $\overline{\nu}(\mu, \omega, \nu, \tau, \lambda, \alpha) < \nu$.*

3.3. Tétel (Növekvő ν)**.** *Legyen $\lambda = \lambda_{01}$ és $\alpha = \alpha_{01}$ az előzőek szerint és legyen Ω^-: $-0.1 \leq \mu \leq 0.1$, $-0.1 \leq \omega \leq 0.1$, $0.02 \leq \nu \leq 0.16$ és $0.8 \leq \tau \leq 1.25$ ekkor a (4) szerint származtatott $\overline{\nu}(\mu, \omega, \nu, \tau, \lambda, \alpha)$-ra teljesül hogy $\overline{\nu}(\mu, \omega, \nu, \tau, \lambda, \alpha) > \nu$. (Ugyanez elmondható $0.02 \leq \nu \leq 0.24$, $0.9 \leq \tau \leq 0.24$ esetén)*

Kísérleti eredmények azt mutatják, hogy az önnormálizálódó neurális hálózatok beleképzik (μ, ν) párt Ω-ba [6].

3.3. Radiális bázisfüggvényes hálózatok

A radiális bázisfüggvényes hálózatok (RBF) két aktív réteggel rendelkező előrecsatolt hálózatok, melyekben a rejtett réteg valamilyen radiális bázisfüggvényt tartalmaz [2]. Ez azt jelenti, hogy a bázisfüggvényhez tartozik egy középpontparaméter, amitől azonos távolságra lévő pontok között a függvény nem tesz különbséget, tehát a függvény-érték csak a c középponttól vett távolságtól függ. Ezeket a függvényeket gömbszimmetrikus függvényeknek nevezzük, a név arra utal, hogy egy n dimenziós gömb felületén lévő pontokat ugyanoda viszi. A kimeneti réteg neuronjai ezen függvény-értékek egy lineáris kombinációját véve határozzák meg az y kimenetet (8. ábra).



8. ábra. Radiális bázisfüggvényes hálózat

$$y = \sum_i w_i g_i(x) = \sum_i w_i g(\|x - c_i\|)$$

A leggyakrabban használt radiális bázisfüggvények a Gauss-függvények:

$$g_i(x) = exp\left(\frac{-\|x - c_i\|^2}{2\sigma_i^2}\right),$$

ahol c_i a középpontparaméter, σ_i pedig a függvényhez tartozó úgynevezett szélességparaméter. Más használatos radiális bázisfüggvények ($r = \|x - c_i\|$):

$$g(r) = r$$
$$g(r) = r^3$$
$$g(r) = (c^2 + r^2)^\beta \quad 0 < \beta < 1$$

Az RBF hálózat megtervezése tehát a háló méretének, a bázisfüggvények szabad paramétereinek és a kimeneti réteg súlyainak meghatározásából áll. A kimeneti réteg w_i súlyait meghatározhatjuk analitikusan, a bázisfüggvények szabad paramétereinek meghatározására több eljárást dolgoztak ki, ezek közül a következő két alfejezetben bemutatom a K-közép eljárást, mellyel a c_i középpontok határozhatóak meg és az R legközelebbi szomszéd módszerét, mellyel a szélességparamétert számolhatjuk ki.

3.3.1. K-közép eljárás

A K-közép (K-means) eljárás célja K darab olyan középpont ($c_i, \quad i = 1, 2 \ldots K$) meghatározása, hogy a tanítópontok és a hozzájuk legközelebb eső középpont négyzetes távolságainak összege minimális legyen. Jelölje X a tanítópontok halmazát, $X^{(s)} \subset X$ pedig azon tanítópontok halmazát, melyekhez a legközelebbi középpont c_s, ekkor a következő kifejezést megoldó c_i-ket keressük:

$$\min_c \sum_{k=1}^{K} \sum_{i:x_i \in X^{(k)}} \|x_i - c_k\|^2 \tag{5}$$

Az algoritmus lépései [2]:

1. Véletlenszerűen választunk K középpontot.
2. Határozzuk meg minden tanítópontra, hogy melyik c_k középpont van hozzá legközelebb, tehát partícionáljuk X-et $X^{(i)}$-kre $i = 1, 2, \ldots, K$

3. Határozzuk meg az $X^{(i)}$ halmazok új középpontjait, egy ilyen halmaz új középpontja legyen a benne lévő pontok átlaga:

$$c_k = \frac{1}{|X^{(k)}|} \sum_{i:x_i \in X^{(k)}} x_i$$

4. Ha a c_i középpontok változtak 3. során akkor térjünk vissza a 2. ponthoz, ha nem akkor álljunk meg.

Ezen eljárással tehát a tanítókészlethez igazodó középpontokat határozhatunk meg, melyek minimalizálják a (5) kifejezést és számuk jóval kisebb a tanítópontok számánál, így biztosítva nagy mintapontszám esetén is a megvalósítható hálóméretet.

3.3.2. R legközelebbi szomszéd módszere

Az R legközelebbi szomszéd módszere egy heurisztikus eljárás a szélességparaméter meghatározására. Az eljárás egy szélességparamétert az adott bázisfüggvény középpontparaméterének és a hozzá legközelebb eső R darab másik középpontparaméter átlagos távolságaként határoz meg.

$$\sigma_k = \frac{1}{R} \sum_{j=1}^{R} \|c_k - c_j\|,$$

ahol $c_j \quad j = 1, \ldots R$ a c_k-hoz legközelebb eső R különböző középpont. A gyakorlatban gyakran használnak minden rejtett neuronra azonos σ szélességparamétert, a tapasztalatok azt mutatják, hogy ekkor egy középpont és a hozzá legközelebb eső másik középpont távolságainak átlaga jó választás:

$$\sigma = \frac{1}{K} \sum_{i=1}^{K} \|c_i - c_j\|,$$

ahol c_j a c_i-hez legközelebb eső középpont. Jól látható, hogy az előző két képlet között csak annyi a különbség, hogy hány távolság átlagát vesszük [2].

3.4. CMAC hálózatok

A CMAC hálózatok a bázisfüggvényes hálózatok és a perceptron keverékének tekinthető [2]. A bemenet itt is egy nemlineáris transzformáción esik át, jelen esetben ez egy asszociációs vektorra való leképezés. Ehhez a hálózat diszkrét bemeneti értékeket tételez fel, ezekhez kölcsönösen egyértelűmen rendel asszociációs vektorokat. A diszkrét bemenet eléréséhez az adatokat először kvantálni kell, a kvantálás határozza meg, hogy hányféle bemenet lehetséges, N-dimenziós adatoknál b-bites kvantálót használva ez az érték 2^{Nb}. Minden diszkrét bemeneti értékekhez tartozó asszociációs vektor bináris, rögzített hosszúságú és a hálózatra jellemző C darab egyest tartalmaz. A közöttük lévő különbséget a C darab egyes helye határozza meg, ezen helyek meghatározása úgy történik, hogy két bemenet minél közelebb van egymáshoz, annál nagyobb legyen az átfedés az asszociációs vektoruk aktív bitjei között, ezáltal megőrizve a közelség által hordozott információt. Az asszociációs vektorra tekinthetünk úgy, mint egy neuronrétegre, amelyben minden diszkrét bemenet esetén pontosan C kerül gerjesztett állapotba, vagyis ad 1 értékű kimenetet, míg minden más aktiválatlan neuron 0-t. Ez úgy valósítható meg, hogy minden neuronhoz hozzá rendelünk egy érzékelési mezőt, amibe eső bemenet esetén aktivizálódik. Ez biztosítja, hogy a közeli bemenetek esetén nagy legyen az átfedés az aktivizálódó neuronok között. Az értelmezési mező valójában a neuronhoz tartozó bázisfüggvény, amely ebben az esetben egy egyszerű indikátor függvény. Bázisfüggvényes hálózat révén, a kimenet itt is a bázisfüggvényes réteg $a(x)$ válasza és a w súlyvektor skaláris szorzata:

$$y(x) = a(x)^T w.$$

Az asszociációs vektor bináris mivolta miatt ez a skaláris szorzat igazából csak w C darab koordinátájának kiválasztása és összegzése:

$$y(x) = \sum_{a_i(x)=1} w_i.$$

A kimenet számítása csak összeadást igényel, ami előny a legtöbb más hálózattal szemben. A kvantált bemeneten kívül másik nagy különbség az RBF-hez képest, hogy ebben az esetben a bázisfüggvénynek nincsenek szabad paraméterei, az egyetlen megválasztható paraméter C, amelyet előre rögzítenünk kell.

3.4.1. Tömörítés

Mivel a lehetséges bemenetek leképezése az asszociációs vektorokra kölcsönösen egyértelmű, az asszociációs vektor hossza (M) meghatározható a lehetséges bemenetek számának (R) és az aktiválódó neuron számának (C) ismeretében. Egydimenziós esetben:

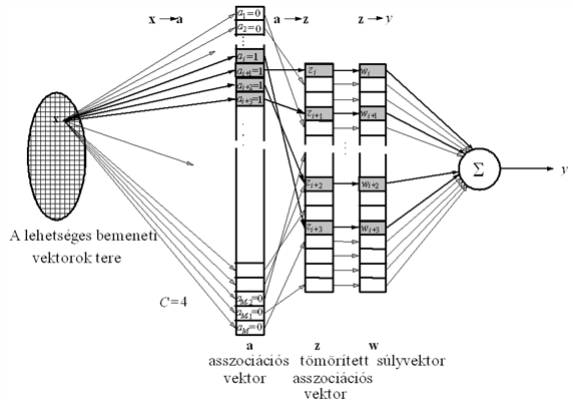
$$M = R + C - 1.$$

Ez a mennyiség határozza meg a háló komplexitását. Az N dimenziós esetben teljes lefedést tekintve:

$$M = \prod_{i=1}^{N} (R_i + C - 1),$$

amely már kis dimenziószám és kis bites kvantálás esetén is megvalósíthatatlanul nagy. Az asszociációs vektor hosszának csökkentésére szolgál a tömörítés, amely során az asszociációs vektorhoz egy új, rövidebb, úgynevezett tömörített asszociációs vektort rendelünk. Jelöljük ezt a tömörített vektort z-vel. A hozzárendelés úgy történik, hogy az eredeti a vektor minden koordinátájához véletlenszerűen rendeljük z egy koordinátáját. A súlyok kiválasztását ezután már z szerint végezzük. Ezzel a hozzárendeléssel megtartjuk a közeli bemenetek közötti átfedéseket az aktivizálódó neuronok terén, ám vegyük észre, hogy a kölcsönösen egyértelműség sérül. Ez azonban nem okoz gondot a gyakorlatban, ennek oka, hogy a lehetséges bemeneti adatoknak csupán kis százaléka kerül valóban a hálózatra. A tömörítéssel a használt asszociációs vektorunk már kisebb, de ennek ára van. Eredetileg egymástól távol lévő tanulóesetek közel kerülhetnek egymáshoz, ezt a jelenséget ütközésnek nevezzük. Az ütközés egy másik formája mikor két a-beli cella ugyanarra a z-belire mutat, ilyen biztosan előfordul, hiszen $|a| > |z|$. Ekkor egy olyan bemenet esetén,

amelyre két ilyen a-beli cella aktiválódik, C-nél kevesebb egyes lesz a tömörített asszociációs vektorban. Ezeknek az ütközéseknek a valószínűsége a tömörítés mértékétől és az aktiválódó bitek számától függ. Megmutatható, hogy ha az aktiválódó neuronok száma kisebb, mint a a tömörített asszociációs vektor hosszának 1%-a, akkor az ütközések valószínűsége már elég alacsony a hatékony megvalósításhoz.



9. ábra. CMAC hálózat tömörítése

3.4.2. CMAC hálózat tanítása

A CMAC hálózat csak a kimeneti lineáris rétegében tartalmaz tanítható súlyokat, ezért tanítása is jelentősen leegyszerűsödik más felépítésű hálózatokhoz képest. A kimeneti rétegre már csak bináris adatok érkeznek bemenetként, ezért a súlymódosítás a következőképpen zajlik:

$$w_i(k+1) = \begin{cases} w_i(k) + \mu\varepsilon(k), & \text{ha } a_i(k) = 1 \\ w_i(k), & \text{ha } a_i(k) = 0 \end{cases}$$

tehát a C darab aktiválódó súly azonos mértékben módosul, míg minden más súly változatlanul marad, ennek több előnye is van. A tanítópontok hatása csak korlátos bemeneti tartományra terjed ki, hiszen kellően távol lévő pontok esetén az aktiválódó neuronok között nincs átfedés, mivel a bázisfüggvények tartója véges, egy

C kvantum oldalhosszúságú hiperkocka. Ennek következtében az egymástól távoli bemenetekhez tartozó válaszok egymástól függetlenül alakíthatóak ki. Az, hogy egy tanítóeset hatására a súlyoknak csak egy meghatározott része módosul nagyban eltér például az MLP tanításától, ahol minden körben minden súlyt módosítunk. Kutatási eredmények azt mutatják, hogy az általánosítási hiba csökkenthető, ha törekszünk a minél egységesebb súlyokra. Ennek elérését célozza következő regularizációs eljárás, ahol a κ kritériumfüggvényt a következőképpen definiáljuk:

$$\kappa(w) = \frac{1}{2}(d(k) - y(k))^2 + \frac{\rho}{2} \sum_{i:a_i=1} \left(\frac{d(k)}{C} - w_i(k) \right)^2.$$

A kritériumfüggvény alakjából látható, hogy a $\frac{d}{C}$ értéktől való eltérést is hibának tekinti, a kétféle hiba közötti viszonyt a ρ regularizációs együttható határozza meg. Ezen kritériumfüggvényhez tartozó tanító eljárás a következő:

$$w_i(k+1) = w_i(k) + \mu\varepsilon(k) + \rho\left(\frac{d(k)}{C} - w_i(k) \right).$$

A szabály neve súlykiegyenlítő regularizáció [2].

3.5. Szekvenciális hálók

Az eddig bemutatott hálókban közös volt, hogy a predikált kimenet csak magától a hálótól és a bemenettől függött, tehát a kapcsolat statikus volt. Szekvenciális vagy más néven időfüggő hálózatoknál azonban ez nincs így, egy kimenet függ a korábbi kimenetektől vagy bemenetektől vagy akár mind a kettőtöl, így a kapcsolat nem statikus hanem dinamikus. A korábbi értékektől való függés miatt a hálónak rendelkeznie kell memóriával, ahol eltárolható a szükséges adatmennyiség. A háló működése diszkrét idejű esetben a következőképpen írható le általánosan:

$$y(k) = f(\Theta, \varphi(k)).$$

Ez rögzíti a modell struktúráját, a $\varphi(k)$ regresszorvektor mutatja, hogy a k-adik időpillanatban mely korábbi adatok befolyásolják a kimenet kiszámítását, Θ pedig a rendszer paramétereit foglalja össze vektoros formában. Ennél a modellnél a feladatunk általában olyan φ, Θ és f keresése, amellyel a modell $y(k)$ kimenetének négyzetes értelemben minimális az eltérése a $d(k)$ elvárt kimenettől. Ezt az eltérést a továbbiakban jelöljük $\varepsilon(k)$-val.

3.5.1. Modellstruktúrák

Attól függően, hogy egy adott pillanatbeli kimenet az időben korábbi adatok mely részhalmazától függ, különböző struktúrájú szekvenciális hálókról beszélhetünk [2].

- NFIR modellek (nonlinear finite impulse response), φ k-hoz csak a legutolsó N bemenetből álló vektort rendeli, így a hálóban nem keletkezik valódi visszacsatolás, előrecsatolt marad.

$$\varphi(k) = [x(k-1), x(k-2), \ldots, x(k-N)]$$

- NARX (nonlinear autoregressive network with exogenous inputs) modellek, $\varphi(k)$ a korábbi N bemenet mellett a korábbi M elvárt kimenetet is tartalmazza. Elsőre úgy tűnhet, hogy a modell visszacsatolt, de ez nem igaz, hiszen az új kimenet a korábbi elvárt kimenetektől függ, és nem a háló által generáltaktól.

$$\varphi(k) = [x(k-1), x(k-2), \ldots, x(k-N), d(k-1), d(k-2), \ldots, d(k-M)]$$

- NOE (nonlinear output error) modelleknél a vektor hasonló mint a NARX modellek esetében, ám itt már az elvárt kimenetek helyett a korábbi rendszerkimenetek szerepelnek, így ez a modell már valóban visszacsatolt.

$$\varphi(k) = [x(k-1), x(k-2), \ldots, x(k-N), y(k-1), y(k-2), \ldots, y(k-M)]$$

- NARMAX (nonlinear autoregressive moving average with exogenous inputs) modellek, a NARX modellekből származtatható azzal a bővítéssel, hogy a korábbi ε különbség értékek is szerepelnek.

  $$\varphi(k) = [x(k-1), \ldots, x(k-N), d(k-1), \ldots, d(k-M), \varepsilon(k-1), \ldots, \varepsilon(k-L)]$$

- (nonlinear box-jenkins) NBJ modellek, ugyanúgy származtathatóak a NOE modellekből mint az előbb.

  $$\varphi(k) = [x(k-1), \ldots, x(k-N), y(k-1), \ldots, y(k-M), \varepsilon(k-1), \ldots, \varepsilon(k-L)]$$

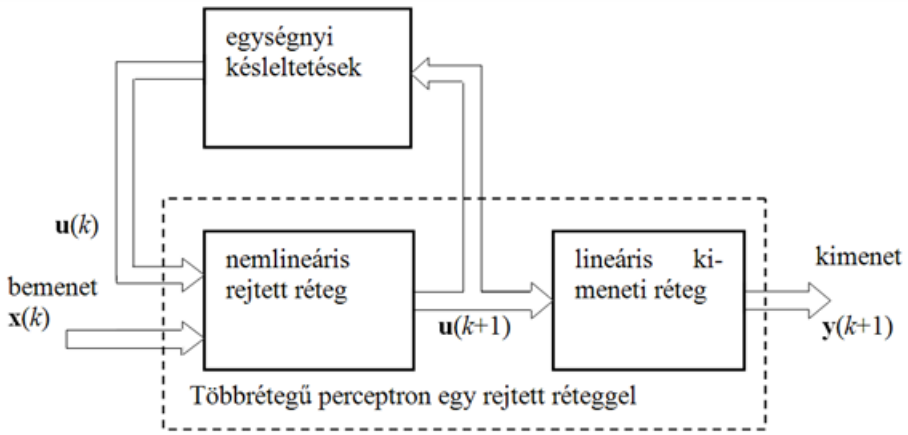
Érdemes megjegyezni, hogy a korábbi adatfajták késleltetésének mértéke páronként eltérő lehet, például lehet hogy $\varphi(k)$ tartalmazza az utolsó három bemeneti éréket, de kimeneti értékből csupán kettőt.

3.5.2. Állapotváltozós reprezentáció

Az előbbi modellosztályok mind bemenet-kimenet reprezentációt használtak, ám a gyakorlatban gyakran alkalmazzák az úgynevezett állapotváltozós reprezentációt, itt a modell minden k időindex esetében egy $u(k)$ állapotban van és a következő időindexhez tartozó állapot a korábbi állapot és az új bemenet függvénye (10.ábra). Ez a következő formában írható le matematikailag:

$$u(k+1) = f(u(k), x(k), b_1(k))$$
$$y(k+1) = g(u(k+1), b_2(k+1))$$

itt tehát $u(k)$ a k-adik állapotvektor, x(k) a k-adik bemeneti vektor, f az állapotváltozást leíró nemlineáris átmenetfüggvény, $y(k)$ a k-adik kimeneti vektor, g a kimenetet meghatározó nemlinearitás, b_1, b_2 pedig zajvektorok. A modell felépítése során az első lépés a modellstruktúra-osztály megválasztása, ám ezután a tényleges struktúra felépítése még komoly feladat. Általában nehéz kérdés, hogy mi legyen



10. ábra. Állapotváltozós reprezentáció

a modell fokszáma, tehát hány paramétert használjon. Lehetőségek tárháza áll a tervező rendelkezésére, amelyből viszont csak kevés illik megfelelően a megoldani kívánt problémához, ennek megtalálása nagy lépés a cél felé. Az egyes architektúrák kezelése, tanítása mind-mind külön feladat, amely a modell mély ismeretét igényli, kevés általános módszer létezik rájuk.

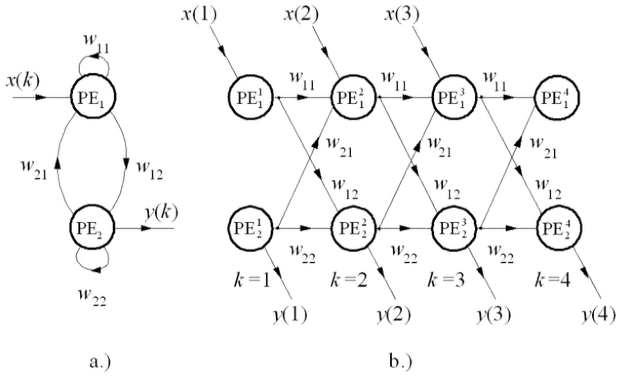
3.5.3. Az időfüggés megvalósítása

Ahhoz, hogy a hálózatot időfüggővé tegyük a legegyszerűbb út az, ha egy adott bemenetet egy késleltető láncon keresztül hozzávesszük a következő bemenethez, így a hálózat új kimenete valóban függeni fog a korábbi adatoktól. Ezt nem csak egy bemenettel, hanem tetszőleges időablakbeli korábbi bemenetekre megtehetjük, például minden bemenetet kiegészítünk a legutóbbi három időindexhez tartozó bemenettel. Az eddigiek alapján tudjuk, hogy így egy NFIR struktúrájú modellt hozunk létre. Kis változtatással minden korábban felsorolt struktúraosztály létrehozható eképpen. Az előző példából az is látszik, hogy az időfüggés létrehozásához nem kell feltétlen visszacsatolás. A input bővítéses módszer egyszerű felépítésű, ám ennek ára van, sok dimenziós esetben a bővítés nagyon nagy inputot eredményez, ami nagyobb számítási kapacitást igényel, ezért a gyakorlatban ez nem sokszor

használható, ezért több más módot dolgoztak ki az időfüggés megvalósítására. Alkalmazhatunk például lineáris szűrőket vagy létrehozhatunk lokális visszacsatolásokat. Ezek a visszacsatolások futhatnak egy rétegen belül vagy visszafelé irányban egy korábbi réteg neuronjához. Ezen lokális visszacsatolások alkalmazásával kapcsolatban a legnehezebb kérdés az időkezelés, amely egyik lehetséges megoldása a háló időbeli kiterítése, melyet a következő alfejezetben foglalok össze.

3.5.4. Időbeli kiterítés

A visszacsatolt hálózat időbeli kiterítése során minden egyes processzáló neuront helyettesítünk egy elemsorozattal, melynek egyes elemei az adott processzáló neuron egy adott időablakbeli eltérő időpillanatokhoz tartozó állapotait reprezentálják [2]. Erre egy egyszerű (két elemű, teljesen visszacsatolt) példát mutat a 11. ábra. Ezen gondolat alapján minden visszacsatolt hálózat megfeleltethető egy előrecsatoltnak, eképpen a kiterítés lehetővé teszi, hogy a korábban tárgyalt hibavisszaterjesztéses algoritmust használjuk a háló tanítására. A kiterítéssel előrecsatolt hálót



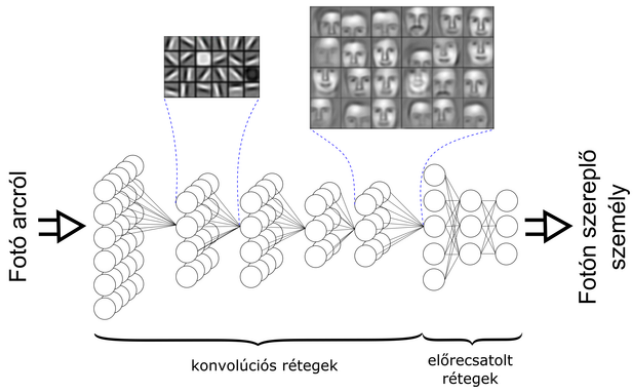
11. ábra. Visszacsatolt háló időbeli kiterítése

kapunk, melynek minden rétege egy adott időpillanathoz tartozik, ezáltal a pontok, és így a súlyok száma is megnövekszik, ám fontos észrevenni, hogy ugyanazon ponthoz különböző rétegben tartozó súlyok azonosak, hiszen ezek az eredeti visszacsatolt háló ugyanazon súlyából keletkeznek. Ennek megfelelően a 11. ábrán is ezek

a fizikailag azonos súlyok ugyanolyan indexszel szerepelnek a különböző rétegekben. A súlyok közti ezen összefüggést figyelembe kell vennünk a kiterített háló tanításánál is. A lépésenkénti hiba ezesetben minden réteg kimenetén megadható, súlymódosításkor ezeket a hibákat kell a keletkezési rétegüktől visszaterjeszteni, ám a súlyok módosítása az előzőek szerint csak az összes rétegben egyszerre és azonos mértékben történhet. Emiatt a súlymódosító-értékeket összeadjuk és az összeggel módosítjuk a súlyokat minden azonos ősképű kiterített súlynál. A módszer fő hátránya, hogy nagy számítási kapacitást és tárhelyet igényel, ugyanis az eredeti háló annyiszorosával dolgozunk amilyen mélyen függ a kimeneti értek a korábbi adatoktól (a korábbiak szerint ezt tárolja a φ függvény). A kiterítésre csak a tanítás során van szükségünk, a betanított hálót már az eredeti, visszacsatolt formában használjuk. Az eljárás másik hátránya, hogy valós időben nem tanítható, a súlymódosítás csak a rekurzió mélységének megfelelő lépés után végezhető el.

3.6. Konvolúciós neurális hálók

A többrétegű neurális hálók egy nagyon érdekes és hasznos fajtája a konvolúciós neurális háló, ennek fő ismertetőjegye a konvolúciós réteg, aminek megalkotása új távlatokat nyitott a neurális hálók felhasználásában, például a képfeldolgozás területén. A réteg működése már ismert részekkel való hasonlóságon, konvolúción alapszik. Egy pici, már ismert képdarabot tolunk végig az egész képen és számszerűsítjük a hasonlóságot a végigtolt szűrő és az éppen soron lévő darab között, így létrehozva egy aktivációs térképet. Ezt több különbőző szűrővel megismételjük és a kapott térképeket egymás mögé rakva nyerjük az új állapotot. Köznapian szólva minden egyes résznél megnézzük az ismert darabokra mennyire hasonlít, így alkotva pontosabb képet a feldolgozandó dologról (12. ábra). A konvolúciós réteggel, főként ha sok van belőle, nagyon megnőhet a reprezentáció mérete, ezért általában társul hozzá úgynevezett összevonó vagy pooling réteg, melynek célja a reprezentáció méretének csökkentése. Ennek leggyakrabban használt fajtája a max pooling és az average pooling. Ezek kis négyzetekre osztják az aktivációs térképeket, például 2x2-esekre és ezeket összevonják. Az összevonás működésében különböznek,



12. ábra. Konvolúciós réteg működése

a max pooling a szereplő maximális értékké vonja össze, míg az average pooling a szereplő értékek átlagává. Az összevonás minden aktivációs térképen egymástól függetlenül zajlik.

4. Neurális hálózat mint approximátor

A mesterséges neurális hálózatok egy érdekes tulajdonsága, hogy felhasználhatóak függvényapproximációra. Érdekesség, hogy ezzel kapcsolatos kérdés már a Hilberttől származó elhíresült 23 pont között is volt, szám szerint a 13. kérdés. Ismeretes, hogy 1900-ban Párizsban nemzetközi kongresszusra gyűlt össze a világ matematikus társadálmának krémje, a kor legkiemelkedőbb matematikusának, David Hilbertnek a részvételével. Ezen alkalomból Hilbert előadást tartott 23 matematikai problémáról, amelyek szerinte meg fogják határozni a XX. század matematikáját, ezek között szerepelt ez a bizonyos 13-as pont. Természetesen a mesterséges neurális hálók tudományága ennél sokkal fiatalabb, ezért az állítás eredeti alakja elsőre ettől távolinak tűnhet. Eredetileg a következőképpen szólt: "Bizonyítsuk be, hogy az $x^7 + ax^3 + bx^2 + cx + 1 = 0$ hetedfokú egyenlet nem oldható meg pusztán kétváltozós függvények segítségével!" [2]. Ez elsőre nagyon speciálisnak és érdektelennek tűnhet, de mégis komoly mögöttes tartalommal bír. A történet külön szépsége, hogy az állítás nem igaz és ezt egy másik híres matematikus, bizonyos Andrej Nyikolajevics Kolmogorov látta be 57 évvel később.

4.1. Approximációs eredmények

Az előző állítás átfogalmazható a következőképp: "Mutassuk meg, hogy van olyan háromváltozós folytonos függvény, mely nem írható fel véges számú kétváltozós folytonos függvény segítségével!" Ez a forma már közelebb áll a neurális hálókhoz, ez azt jelenti a neurális hálók nyelvén, hogy létezik olyan hárombemenetű háló, amely nem vezethető vissza kétbemenetűekre. Kolmogorov a cáfolatnál sokkal többet bizonyított, nem csak hogy minden háromváltozósra, hanem tetszőleges N változós folytonos függvényre bizonyította, hogy felírható mindössze egyváltozós függvények és összeadás felhasználásával [2].

4.1. Tétel (Kolmogorov)**.** *Minden $N \geq 2$ egész esetén található $N(2N + 1)$ darab olyan folytonos, monoton növekvő, egyváltozós, az $I = [0, 1]$ intervallumon értel-*

mezett ψ_{pq} függvény, melyek segítségével tetszőleges N változós, valós, folytonos $f : [0,1]^N \to \mathbf{R}$ függvény az alábbi alakban előállítható:

$$f(x_1, x_2 \dots x_N) = \sum_{q=1}^{2N+1} \phi_q(\sum_{p=1}^{N} \psi_{pq}(x_p))$$

Kolmogorov tételében szereplő előállítást felfoghatjuk úgy, mint egy két nem lineáris rejtett réteget tartalmazó mesterséges neurális háló által elvégzett transzformációt. Itt $\phi_q \quad q = 1, \dots, 2N+1$ egyváltozós függvények az adott f függvényhez tartoznak, ez felel meg az első rejtett rétegnek, míg a ψ_{pg} $p = 1, \dots, N$ $q = 1, \dots, 2N+1$ egyváltozós függvények csak N-hez tartoznak, nem az egyes f-ekhez, ők alkotják a második rejtett réteget. A kimeneti réteg pedig egyszerű összegzést végez. Ezen eredmény is azt mutatja, hogy egy kevés rétegből álló neurális hálóban is már nagy potenciál rejlik. Felhasználás szempontjából sajnálatosan a tétel bizonyítása nem konstruktív, így a bizonyítás során nem állítjuk elő a ϕ_q és ψ_{pq} függvényeket, ennek ellenére nagy a tétel elméleti jelentősége, hiszen kimondja, hogy a kívánt előállítás, és így a kívánt neurális háló létezik. Kolmogorov tételének egy lehetséges továbbfejlesztése David Sprecher nevéhez fűződik és a következőképpen szól [2].

4.2. Tétel (Sprecher)**.** *Létezik olyan monoton növekvő, valós $\psi(x) : [0,1] \to [0,1]$ függvény, mely csak N-től függ, és található hozzá $\varepsilon > 0$ úgy, hogy minden folytonos, valós $f : [0,1]^N \to \mathbf{R}$ függvény felírható az alábbi formában:*

$$f(x) = \sum_{q=1}^{2N+1} \phi_q(\sum_{p=1}^{N} \lambda_p \psi(x_p + q\varepsilon)),$$

ahol a ϕ_q függvények valós, folytonos függvények, míg a λ_p értékek megfelelő konstansok.

Ez az alak nagyon hasonló az előző tételben szereplőhöz, ám itt ψ_{pq} függvények helyett már egy közös ψ szerepel, ami már semmilyen módon nem függ az adott, felbontani kívánt f függvénytől. Ezt a ψ-t nem az x_p helyeken értékeljük ki mint a Kolmogorov-féle reprezentációban, hanem onnan eltolva. Neurális hálóval való

megvalósítást tekintve ez a képlet is két rejtett réteget igényel, ahol tehát az első rejtett réteg aktivációs függvénye már f-től független. Az előző két tétel két rejtett rétegű neurális hálókra vonatkozott, ezek alapján kimondhatunk egy ezeknél erősebb tételt, amely Robert Hecht-Nielsen-től származik [2][11].

4.3. Tétel (Hecht-Nielsen)**.** *Minden* $f : [0,1]^N \to \mathbf{R}$ L^2*-beli függvény átlagos négyzetes értelemben tetszőleges pontossággal közelíthető olyan neurális hálózat segítségével, amelynek legfeljebb két feldolgozó rétege van, és a neuronokban szigmoid jellegű nemlinearitást alkalmaz.*

Hecht-Nielsen tétele erős, de két feldolgozó neurális réteg elégségességéről szól. A későbbiekben kiderül, hogy ez tovább erősíthető úgy, hogy csak egy feldolgozó réteget használunk fel.

4.2. Approximáció egyetlen feldolgozó réteggel

Első ránézésre egy rejtett réteg nagyon kevésnek tűnhet, ám már egy ilyen kis mélységű mesterséges neurális hálóban is komoly lehetőségek rejlenek. Erről szól Funahashi következő tétele [2][12].

4.4. Tétel (Funahashi)**.** *Legyen* $g(x)$ *nemkonstans, korlátos, monoton növekvő folytonos függvény. Legyen továbbá* $A \subset \mathbf{R^n}$ *kompakt, és legyen* $f(x_1, \ldots, x_N)$ *egy valós értékű folytonos függvény A-n. Ekkor tetszőleges* $\varepsilon > 0$ *esetére létezik egy olyan* $M \in \mathbf{R}$*, és léteznek* c_i*,* $i = 1, \ldots, M$ *, valamint* w_{ij}*,* $i = 1, \ldots, M$*,* $j = 1, \ldots, N$ *konstansok, hogy:*

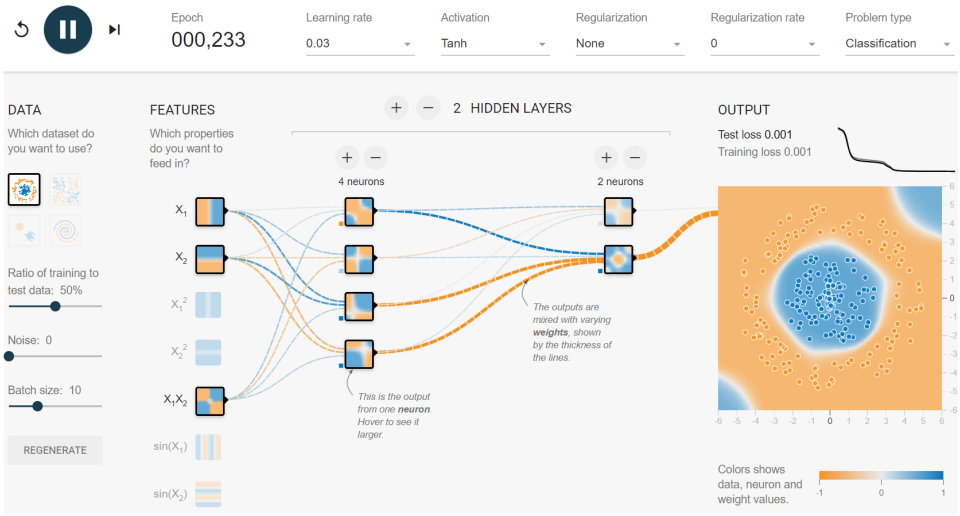
$$\hat{f}(x_1, \ldots, x_N) = \sum_{i=1}^{M} c_i g(\sum_{j=0}^{N} w_{ij} x_j),$$

ahol $x_0 = 1$ *és*

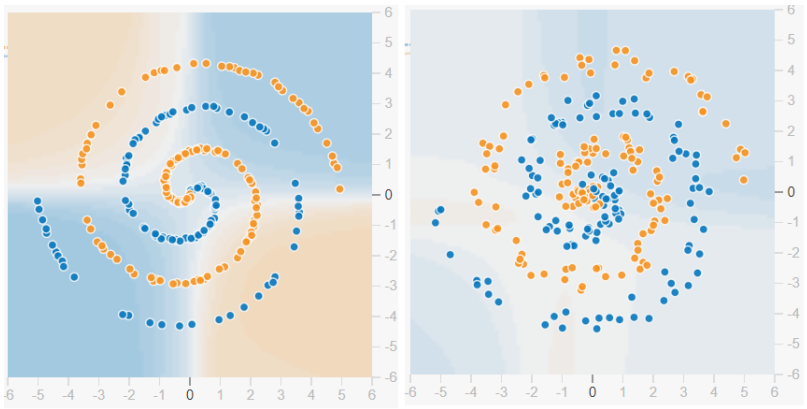
$$\max_{x \epsilon A} |f(x_1, \ldots, x_N) - \hat{f}(x_1, \ldots, x_N)| < \varepsilon$$

.

Itt $\hat{f}$ alakjából látszik, hogy az egyetlen rejtett feldolgozó réteg i-edik neuronja veszi x_j bemeneteinek egy súlyozott összegét és minden neuron erre alkalmaz egy közös g függvényt. Ezután a kimeneti réteg már csak ezen neuronoktól kapott eredmények c_i-vel súlyozott összegét számítja ki. Az adott előállítás tehát nem az eredeti f-et álljtja elő, hanem ahhoz tetszőlegesen közeli függvényt.



13. ábra. Playground.tensorflow.org működése



14. ábra. Pontok eloszlása spirál mintázat esetén kis, illetve nagy zajjal szennyezve

5. Gyakorlati megvalósítás

Saját tapasztalataimból kiindulva a mesterséges neurális hálók elméletének vizsgálata közben a gyakorlati megvalósítás és alkalmazás néha távolinak tűnhet. Szerencsére az interneten több oldal is segíti ezen szakadék áthidalását, és kézzel foghatóvóbbá teszi a mesterséges neurális hálók működését.

5.1. Online vizualizáció

A playground.tensorflow.org egy olyan oldal, ahol valósidőben követhetjük végig egy többrétegű perceptron tanulását és tesztelését kétosztályú osztályozási feladat esetén. A háló felépítését legördülő sávok és csúszkák segítségével állíthatjuk be. Lehetőségünk van a bemeneti neuron, rejtett rétegek és rejtett neuronok számának változtatására, szabályozhatjuk a zajt, az epochok és batchek számát. Különböző aktivációs függvények és tanulási tényezők közül választhatunk. Különböző ponteloszlások állnak rendelkezésünkre, melyek közül kiválaszthatjuk a tanító és teszt pontok milyen mintát kövessenek. A zajjal ennek a szabályosságát állíthatjuk be, kis zaj esetén a kék és narancssárga pontok jól elkülönülnek egymástól, a zajt növelve viszont elkezdenek egyre jobban keveredni, ezzel megnehezítve a háló feladatát.
Az oldal felépítését a 13. ábra, a zaj hatását a 14. ábra mutatja. A play gombra kattintva elindíthatjuk a háló tanulását. Eközben nyomon tudjuk követni a neuronok közti súlyok alakulását, kék vonalak jelölik a pozitív, narancssárgák a negatív súlyokat. Minél nagyobb egy súly abszólút értéke annál vastagabb vonal reprezentálja. A jobb oldali négyzet mutatja, hogy az aktuális állapotában a háló miként osztályozza a pontokat, fölötte a teszt, illetve tanító halmazon elkövetett hibát olvashatjuk. Tanulságos ugyanazon problémát különböző beállításokkal lefutatni, és megfigyelni mennyire más eredmények jöhetnek. Ez is azt mutatja mennyire fontos a háló felépítésének, az aktivációs függvénynek, a tanulásnak és a háló egyéb paramétereinek megfelelő választása, amely a megoldani kívánt feladattól is függ.

Ábrák jegyzéke

Hivatkozások

[1] Fazekas, István. "Neurális hálózatok." Debreceni Egyetem Informatikai Kar (2003).

[2] Altrichter, Márta, et al. "Neurális hálózatok." Panem, Budapest (2006).

[3] Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. Bevezetés az adatbányászatba. Panem Kft., 2011.

[4] Kristóf, Tamás. "A mesterséges neurális hálók a jövőkutatás szolgálatában." Budapesti Közgazdaságtudományi és Államigazgatási Egyetem Jövőkutatási Kutatóközpont (2002).

[5] Vidal, Rene, et al. "Mathematics of deep learning." arXiv preprint arXiv:1712.04741 (2017).

[6] Klambauer, Günter, et al. "Self-normalizing neural networks." Advances in neural information processing systems. 2017.

[7] Rosenblatt, Frank. "The perceptron: a probabilistic model for information storage and organization in the brain." Psychological review 65.6 (1958): 386.

[8] Hebb, Donald Olding. The organization of behavior: A neuropsychological theory. Psychology Press, 2005.

[9] Adee, Sally. "Will AI's bubble pop?." (2016): 16-17.

[10] Novikoff, Albert B. On convergence proofs for perceptrons. STANFORD RESEARCH INST MENLO PARK CA, 1963.

[11] Hecht-Nielsen, Robert. "Theory of the backpropagation neural network." Neural networks for perception. Academic Press, 1992. 65-93.

[12] Funahashi, Ken-Ichi. "On the approximate realization of continuous mappings by neural networks." Neural networks 2.3 (1989): 183-192.