

# 1. fejezet

## Mesterséges neuron

*Ebben a részben a mesterséges neuron struktúrájának az ismertetése, neuronhálókkal kapcsolatos elemek, alapfogalmak, aktivációs függvénytípusok szemléltetése és a neuron matematikai modellje kerül bemutatásra.*

### 1.1. A neuronok felépítése

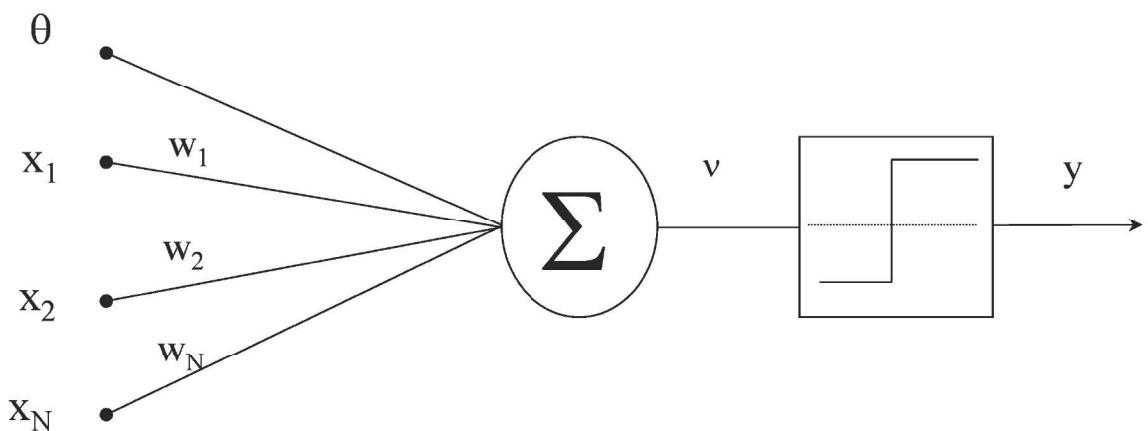
Egy neurális hálózat nagy számú, egyszerű felépítésű processzáló elemek, neuronok rendezett, reguláris felépítését tartalmazza. A neurális hálózat azonos vagy hasonló felépítésű, lokális feldolgozást végző műveleti elemekből épül fel. A feldolgozó elemeket neuronoknak nevezzük.

A neuronhálóban a feldolgozó elemek általában rendezett topológiájúak és jól meghatározott módon vannak összekapcsolva. A feladat végrehajtása során fontos szerepet játszik a neuronháló szerkezete. A neuronhálók belső párhuzamos felépítéséből adódóan a számítások párhuzamosan hajthatók végre, ezáltal nagy feldolgozási sebességet biztosítva. Így a neuronhálók kimondottan alkalmasak valós időben működő feladatok megoldására.

A neuronháló párhuzamos feldolgozást tesz lehetővé és lehetséges szoftver vagy hardver alapú megvalósításra. Speciális hardvereszközök alkalmazhatók a neuronhálók megvalósítására, amelyek kihasználják a neuronhálók párhuzamos szerkezeti felépítését. Számos neuronháló-típusnak megfelelően neuroprocesszor is rendelkezésre áll, amelyek többé-kevésbé kiaknázzák egy adott neuronhálóban rejlő párhuzamosságot. A valós idejű működés mellett a párhuzamos szerkezeti felépítésből adódóan a neuronhálók redundáns magas fokú hibatűrő rendszernek bizonyulnak [2].

A neurális hálózat azonos vagy hasonló felépítésű, lokális feldolgozást végző műveleti elemekből épül fel. A feldolgozó elemeket neuronoknak nevezzük. A neuronhálóban a feldolgozó elemek általában rendezett topológiájúak és jól meghatározott módon vannak összekapcsolva. A neuronháló párhuzamos feldolgozást tesz lehetővé és lehetséges szoftver vagy hardver alapú megvalósítása [2].

A neuron több bemenetű, egy kimenetű feldolgozó (műveletvégző) elem, amely rendelkezhet lokális memóriával, amelyben akár bemeneti, akár kimeneti értékeket tárol [3]. A bemeneti és tárolt értékekből az aktuális kimeneti értéket tipikusan nemlineáris (vagy lineáris) transzferfüggvény segítségével hozza létre, amelyet aktiváló függvénynek nevezünk.



1.1. ábra. Egy általános neuron szerkezete

- A neuron bevezetése során a következő jelöléseket alkalmazzuk:
- $x_1, x_2 \dots x_N$  – a neuron bemenetei
- $X = [x_1, x_2 \dots x_i \dots x_N]$  – bemeneti vektor
- $N$  – neuron bemeneteinek a száma
- $\theta$  – konstans bemenet (bias – eltolási érték)
- $\nu$  – inger (súlyozott összeg)
- $y$  – a neuron kimenete
- $w_i$  – az i-edik bemenethez kapcsolódó súlytényező
- $W = [w_1 w_2 \dots w_i \dots w_N]$  – súlyvektor
- $\varphi$  – aktivációs függvény

A neuron bemenetei között (1.1. ábra) megkülönböztetünk konstans és változó értékű bemeneteket. A konstans bemenetnek a tanulás során van

szerepe. Az  $x_i$  bemenetek változó értékűek, míg a  $\theta$  konstans marad, miután az értékét meghatároztuk.

Az  $x_i$  skalár-bemenetek  $w_i$  súlyozással kerülnek összegzésre, majd a súlyozott összeg egy nemlineáris elemre kerül. A bemeneti jelek súlyozott összegét, mely az aktivációs függvény bemenete, ingernek (excitation), míg a kimeneti jelet válasznak (activation) nevezzük. A  $\varphi$  függvényt aktiváló függvénynek (activation function) nevezzük. A neuron kimenete a következőképpen számolható ki:

$$y = \varphi \left( \sum_{i=1}^N x_i w_i - \theta \right).$$

A súlytényezők a neuronok lokális környezetében lévő más neuronokkal való kapcsolatok erősségeit határozzák meg. A neuronhálók működésének szabályozása a súlytényezőkkel történik, az információ vagy az információfeldolgozó eljárás rögzítése a súlytényezőkben (hangolható paraméterekben) valósul meg a tanítási folyamat során. A neuronháló rendelkezik tanulási algoritmussal (learning algorithm), amely általában minta alapján való tanulást jelent.

A neurális hálózatok működése tipikusan két fázisra választható szét:

1. Tanulási fázis – a hálózatban valamilyen módon eltároljuk a kívánt információfeldolgozó eljárást.
2. Előhívási fázis (recall) – a tárolt eljárás felhasználásával elvégezzük az információfeldolgozást.

A két fázis a legtöbb esetben időben szétválik, rendszerint a tanulási fázis lassú, több iterációt, esetleg sikertelen tanulási szakaszokat is hordoz. Általában a tanítás korszakokba (epoch) van szervezve, és egy-egy korszak lefuttatása után lehetőség van a tanítási paraméterek újrahangolására. Az előhívási fázis gyors feldolgozást jelent [4] és rendszerint az alkalmazás pillanatában történik. Az előhívási fázis során a pillanatnyi bemeneti értékek alapján meghatározzuk a neuronháló kimenetét.

Adaptív rendszerek esetében a két fázis nem válik szét, az információ-előhívással párhuzamosan valósul meg a tanulás, az előhívási szakaszban történik a paramétereik módosítása is.

## 1.2. Aktivációs függvények

A neuronok működésében fontos szerepet töltnek be az aktivációs függvények. A neuronhálókban elsősorban a nemlineáris és folytonosan differenciálható aktivációs függvényeknek tulajdonítható fontosabb szerep. Megfelelő neuron mellett a nemlineáris aktivációs függvény alkalmazása lehetővé teszi bármilyen nemlineáris függvény neuronhálóval való modellezését. Lineáris aktivációs függvény alkalmazása eredményeként a neuronháló is nemlineáris. Ahhoz, hogy a neuronhálót nemlineárissá tegyük, legalább egy nemlineáris aktivációs függvényt kell alkalmazni. A differenciálhatóság is fontos, mert a gradiens alapú tanítás a leggyakrabban alkalmazott módszer a neuronhálók súlyainak a hangolására.

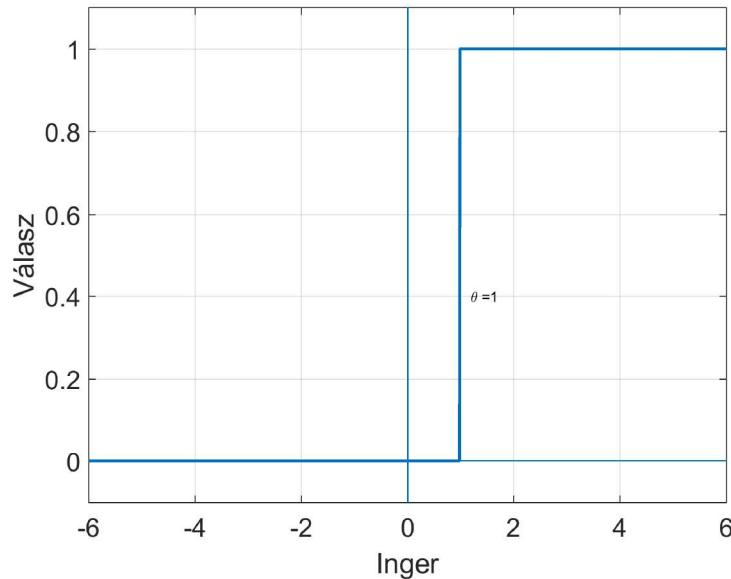
A mesterséges neuron esetében a következő aktivációs függvények alkalmazhatóak [2], [3], [4], [1]:

- Küszöbfüggvény vagy egységugrás-függvény (1.2. ábra)
- Lépcsőfüggvény (1.3. ábra)
- Lineáris aktivációs függvény (1.4. ábra)
- Logisztikus vagy szigmoid alakú karakterisztika
- Tangens hiperbolikus függvény (1.7., 1.8. ábra)
- Telítéses lineáris függvény (1.9. ábra)
- Gauss-függvény
- ReLU aktivációs függvény (1.11. ábra)
- Leaky ReLU (szivárgó ReLU)
- PReLU aktivációs függvény
- ELU aktivációs függvény (1.13. ábra)
- Softmax aktivációs függvény

### 1.2.1. Küszöbfüggvény vagy egységugrás-függvény

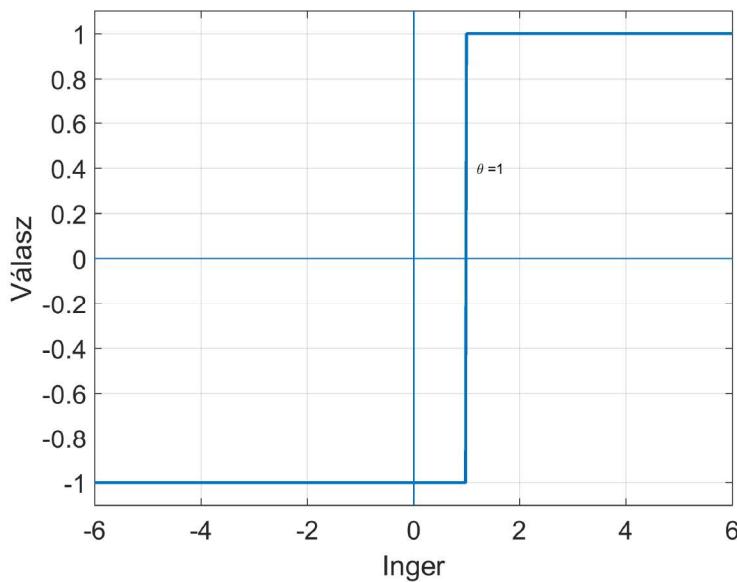
Egységugrás aktivációs függvényre, ha az inger értéke meghaladja a  $\theta$  küszöböt, a neuron kimenete egyes, különben zérus értéket vesz fel (1.1).

$$\varphi(x) = \begin{cases} 0 & \text{ha } x \leq \theta \\ 1 & \text{ha } x \geq \theta \end{cases} \quad (1.1)$$



1.2. ábra. Küszöbfüggvény

### 1.2.2. Lépcsőfüggvény



1.3. ábra. Lépcsőfüggvény

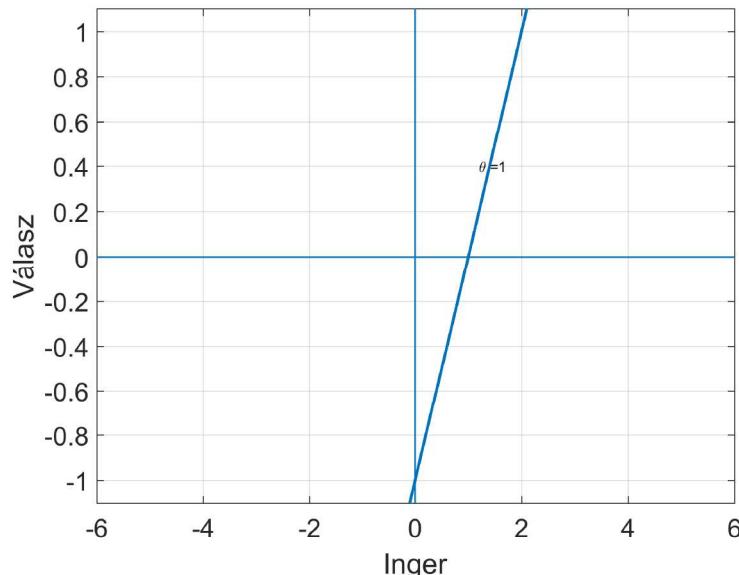
Lépcsőfüggvény aktivációs függvényre, ha az inger értéke meghaladja a  $\theta$  küszöböt, a neuron kimenete egyes, különben minusz egy (1.2).

$$\varphi(x) = \begin{cases} -1 & \text{ha } x < \theta \\ 1 & \text{ha } x \geq \theta \end{cases} \quad (1.2)$$

### 1.2.3. Lineáris aktivációs függvény

A neuron válasza, lineáris aktivációs függvény alkalmazása esetében, lineárisan növekszik az inger értékével. Az aktivációs függvényt az (1.3) egyenlet írja le.

$$\varphi(x) = x \quad (1.3)$$

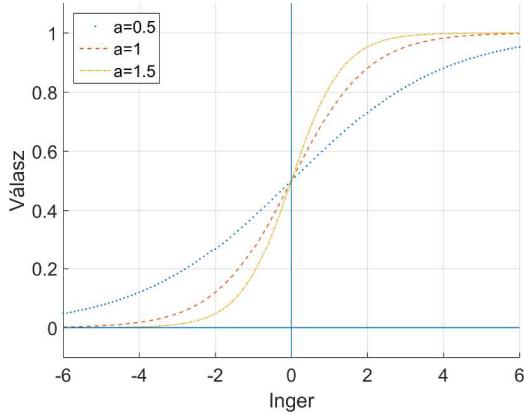


1.4. ábra. Lineárisan növekvő aktivációs függvény

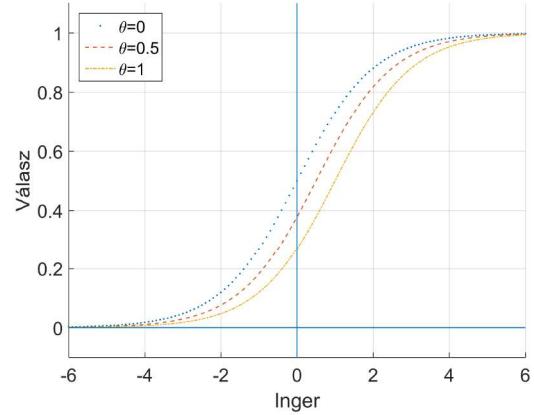
### 1.2.4. Logisztikus vagy sigmoid alakú karakterisztika

A sigmoid alakú aktivációs függvény értéke az egyenlet (1.4) alapján számolható ki. Az  $a$  paraméterrel lehet változtatni a függvény szaturációs részét. A függvény különböző  $a$  paraméter értékekre az 1.5. ábrán, valamint  $\theta$  értékekre az 1.6. ábrán van szemléltetve.

$$\varphi(x) = \frac{1}{1 + e^{-ax-\theta}} \quad (1.4)$$



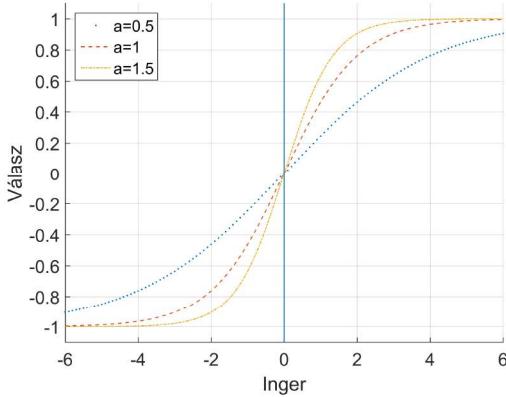
1.5. ábra. Sigmoid aktivációs függvény különböző a értékekre



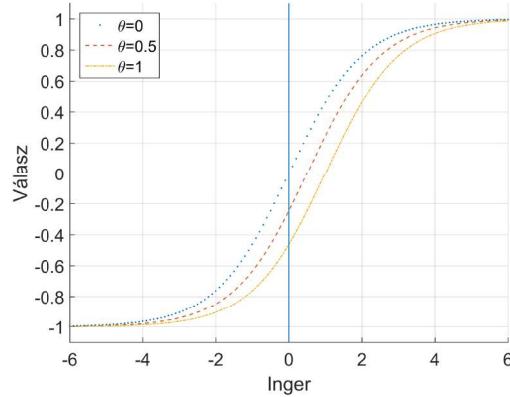
1.6. ábra. Sigmoid aktivációs függvény különböző  $\theta$  értékekre

A szigmoid a biológiából inspirált aktivációs függvény, a teljes tartományban sima és differenciálható. Korábban a többrétegű perceptron típusú neuronhálókban nagyrészt szigmoid aktivációs függvényt alkalmaztak. A szigmoid függvény szaturálódik az inger értékeinek növelésével, vagyis a deriváltja nagyon kicsi lesz, aminek a hatására a súlytényezők hangolása elakad. Azt is szokás mondani, hogy az adott neuron elhal.

### 1.2.5. Tangens hiperbolikus függvény



1.7. ábra. Tangens hiperbolikus aktivációs függvény különböző a értékekre



1.8. ábra. Tangens hiperbolikus aktivációs függvény különböző  $\theta$  értékekre

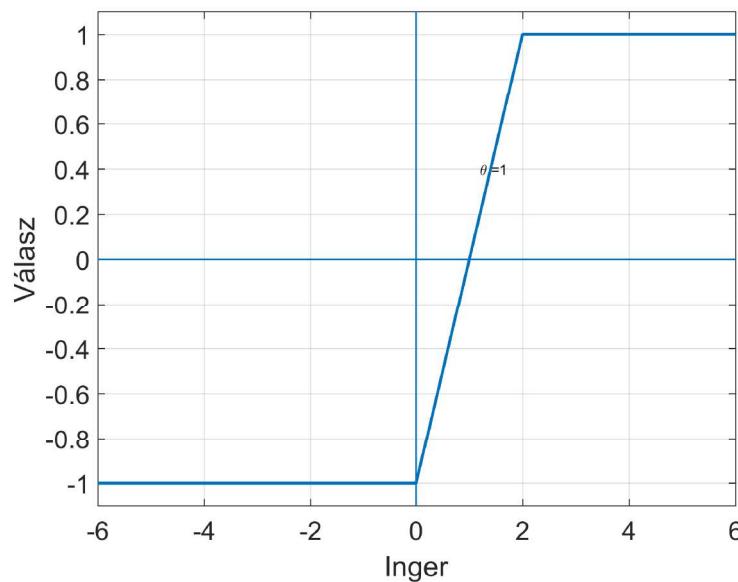
A tangens hiperbolikus aktivációs függvény értéke az (1.5) egyenlet alapján számolható ki. Az 1.7. ábrán a tangens hiperbolikus függvény különböző  $a$ , valamint  $\theta$  értékekre van ábrázolva.

$$\varphi(x) = \frac{1 - e^{-ax-\theta}}{1 + e^{-ax-\theta}} \quad (1.5)$$

Előnye a szigmoid aktivációs függvényhez képest, hogy az origóhoz közelí értékekre a függvény értéke is zérus. Ez egy fontos szempont, mivel gyorsítja a gradiens módszer konvergenciáját. Azonban hasonlóan a szigmoid függvényhez, a tangens hiperbolikus is szaturálódik az inger abszolút értékeinek növelésével.

### 1.2.6. Telítéses lineáris függvény

A lineáris függvényre az aktiválás arányos a bemenettel. Ha egy többrétegű neuronháló mindenik rétege lineáris aktivációs függvényre épül, nem számít, hány rétege van a neuronhálónak, mert csak lineáris leképzést képes megvalósítani. A neuronháló egyetlen nemlineáris aktivációs függvényekre épülő réteggel ekvivalens. A telítéses lineáris függvény (1.9. ábra) a lineáris



1.9. ábra. Telítéses lineáris aktivációs függvény

függvényre épül, szaturálva egy-egy adott érték felett (és alatt) a kimenetet (1.6).

$$\varphi(x) = \begin{cases} -1 & \text{ha } x < -1 \\ x & \text{ha } -1 \leq x < 1 \\ 1 & \text{ha } x \geq 1 \end{cases} \quad (1.6)$$

### 1.2.7. Gauss-függvény

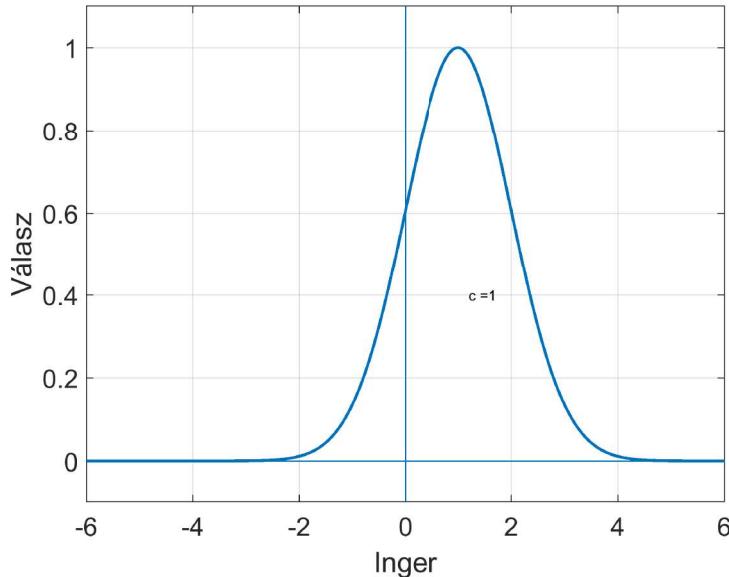
A Gauss-függvényt (1.10. ábra) például az RBF típusú hálók és Kernel-hálók esetében alkalmazzák. Az RBF típusú hálót a 6. fejezet mutatja be részletesen. A Gauss-függvény az (1.7) egyenlet szerint számolható.

$$\varphi(\underline{x}, \underline{c}, \sigma) = e^{\frac{-\|\underline{x} - \underline{c}\|^2}{2\sigma^2}} \quad (1.7)$$

ahol

$\sigma$  – szórás

$\underline{c}$  – középpont



1.10. ábra. Gauss aktivációs függvény  $c=1$ ,  $\sigma = 1$

$$\|\underline{x} - \underline{c}\| = \sqrt{(x_1 - c_1)^2 + (x_2 - c_2)^2 + \dots + (x_N - c_N)^2}$$

ahol

$\underline{x}$  – bemeneti vektor

$\underline{c}$  – középpont vektor

$\sigma$  – szórás

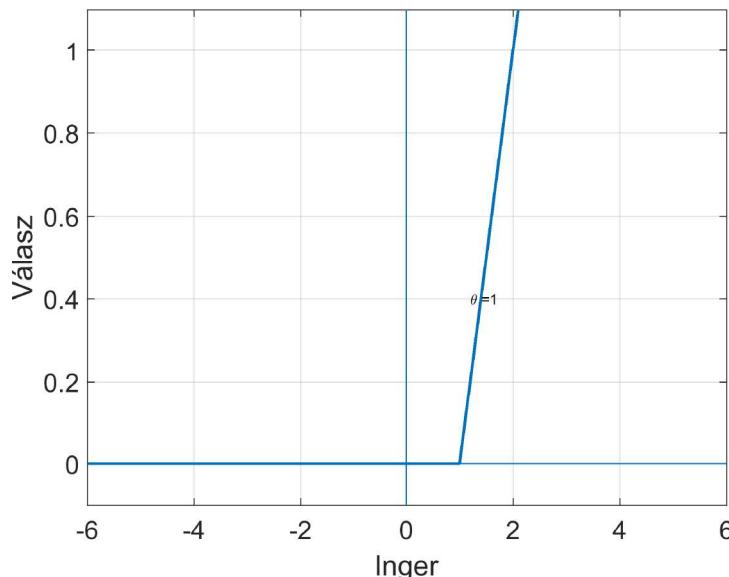
### 1.2.8. ReLU rektifikált lineáris

Az utóbbi években az egyik legnépszerűbb aktivációs függvény [1]. A bevezetését követően helyettesíti a leggyakrabban alkalmazott szigmoid aktivációs függvényt, a gépi tanulási feladatokból származó pozitív hatás következményeként. A közelmúltban bebizonyosodott, hogy hatszor gyorsabb konvergenciát biztosít, összehasonlítva a tangens hiperbolikus aktivációs függvénnnyel.

A ReLU-t és deriváltját az (1.8) és (1.9) egyenletek írják le.

$$\varphi(x) = \max(0, x) \quad \text{vagy} \quad \varphi(x) = \begin{cases} 0 & \text{ha } x \leq 0 \\ x & \text{ha } x \geq 0 \end{cases} \quad (1.8)$$

A sok réteget tartalmazó neuronhálókat mély neuronhálóknak is szokás nevezni. A szigmoid és tangens hiperbolikus aktivációs függvények alkalmazása a kevés réteget tartalmazó neuronhálókra szükül. Az említett függvények gradiensse eltűnik a mély neuronhálók esetében, megakadályozva a neuronháló tanulását. A ReLU aktivációs függvény deriváltja minden egy és nincs szaturálva a kimenet. Jó megoldást biztosít a mély neuronhálók esetében. Negatív ingerértékekre zérus kimeneti neuronokat eredményez. A neuronok, amelyek minden esetre zérus kimenetet eredményeznek, egyszerűen kivághatók a neuronhálóból, csökkentve a számításigényt.



1.11. ábra. ReLU aktivációs függvény

ReLU aktivációs függvény deriváltja:

$$\varphi'(x) = \begin{cases} 0 & \text{ha } x < 0 \\ 1 & \text{ha } x \geq 0 \end{cases} \quad (1.9)$$

Egyes gradiens alapú tanulási módszerek során a ReLU aktivációs függvény is a neuron elhalásához, elvesztéséhez vezethet. Egy olyan súlytényező-adaptálást eredményezhet, amelyet követően a neuron többet egyetlen adatpontra sem aktiválódik. A neuronháló szempontjából az említett neuronok elvesznek, sőt pazarolják a számítási erőforrásokat.

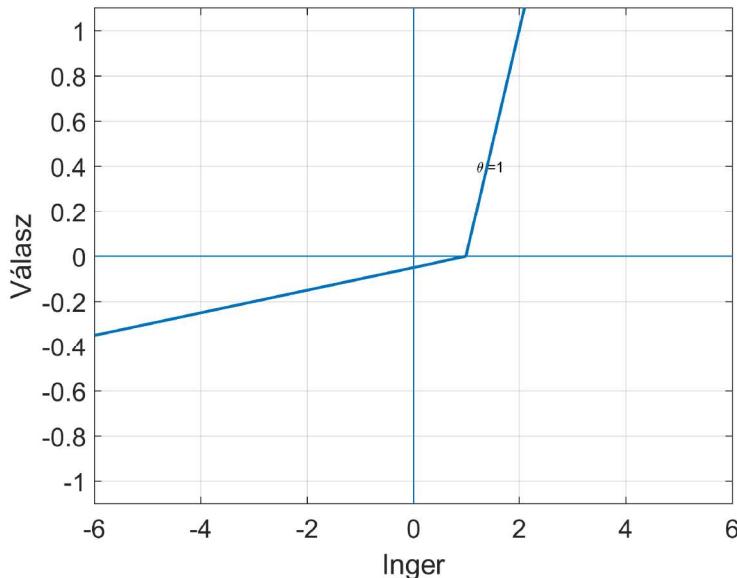
### 1.2.9. Leaky ReLU (szivárgó ReLU)

A szivárgó ReLU (1.10) aktivációs függvényt (1.12. ábra) [1] a ReLU hátrányának kiküszöbölésére vezették be. A függvénynek a negatív részére is egy lejtést biztosít, lehetővé téve ezen a szakaszon is az adaptív súlytényező-beállítást.

$$\varphi(x) = \begin{cases} \alpha x & \text{ha } x < 0 \\ x & \text{ha } x \geq 0 \end{cases} \quad (1.10)$$

LReLU deriváltja (1.11)

$$\varphi'(x) = \begin{cases} \alpha & \text{ha } x < 0 \\ 1 & \text{ha } x \geq 0 \end{cases} \quad (1.11)$$



1.12. ábra. Leaky ReLU aktivációs függvény,  $\alpha = 0,05$

### 1.2.10. PReLU aktivációs függvény

A lényeges különbség a LReLU és a PReLU (1.12) között, hogy a parametrizált rektifikált lineáris egység esetében az  $\alpha_i$  a neuronháló egy tanítható paramétere [1]. Az  $\alpha_i$  paraméter hangolása a neuronháló súlytényezőihez hasonlóan valósítható meg [1].

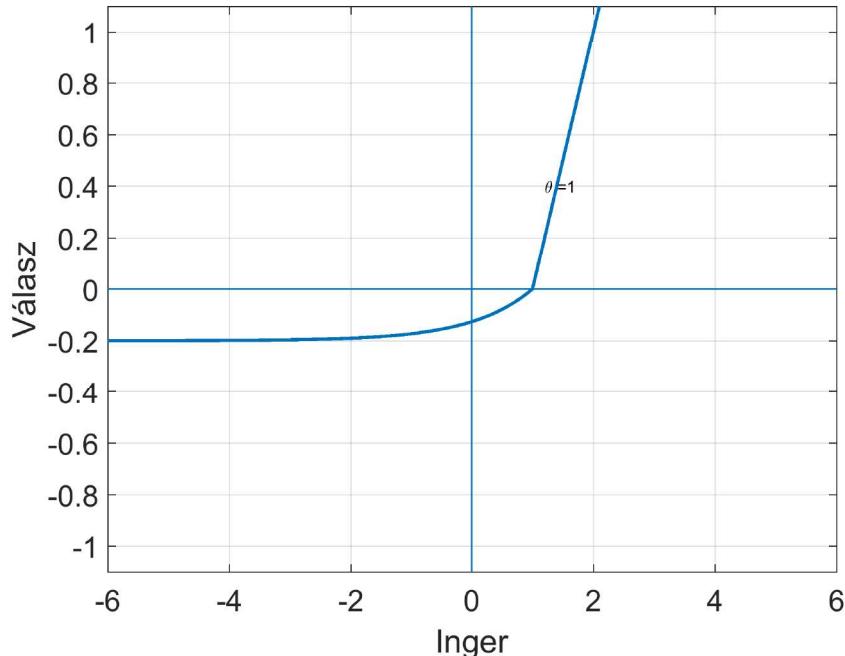
$$\varphi(x) = \begin{cases} \alpha_i x & \text{ha } x < 0 \\ x & \text{ha } x \geq 0 \end{cases} \quad (1.12)$$

PReLU deriváltja (1.13)

$$\varphi'(x) = \begin{cases} \alpha_i & \text{ha } x < 0 \\ 1 & \text{ha } x \geq 0 \end{cases} \quad (1.13)$$

### 1.2.11. ELU aktivációs függvény

Az exponenciális lineáris egység (ELU) egy újabb, a ReLU-ra épülő aktivációs függvény (1.13. ábra). Felgyorsítja a tanulást és enyhíti az eltűnő gradiens problémát [1]. Egy egyszerű konvolúciós neuronhálót alkalmazva, a CIFAR-100 adatbázist használva, összehasonlították a ReLU és ELU aktivációs függvényeket. Az eredmények alapján a neuronháló jobban teljesített az ELU aktivációs függvényt alkalmazva a ReLU-hoz képest [5]. Az ELU-t



1.13. ábra. ELU aktivációs függvény,  $\alpha = 0,2$

az (1.14) egyenlet írja le.

$$\varphi(x) = \begin{cases} \alpha(e^x - 1) & \text{ha } x < 0 \\ x & \text{ha } x \geq 0 \end{cases} \quad (1.14)$$

### 1.2.12. Softmax aktivációs függvény

A softmax aktivációs függvény (1.15) a szigmoid függvényhez hasonlóan minden kimenetet a 0 -1 tartományba vetít (normalizálja). Viszont a kimeneteket elosztja a rendszer kimeneteinek összegével, úgy, hogy a normalizált kimenetek összege 1. A Softmax normalizált exponenciális függvényként is értelmezhető. Osztályozási feladatokban a neuronháló kimenetén általában softmax aktivációs függvényt alkalmaznak. A softmax kimeneti aktivációs függvény több osztály diszkrét valószínűségi eloszlását határozza meg. Nagyon fontos szerepe van a mai sokrétegű konvolúciós neuronhálózatokban, mint például a VGGNet, AlexNet, GoogleNet.

$$\varphi_i = \frac{e^{y_i}}{\sum_{j=1}^N e^{y_j}} \quad (1.15)$$

$y_i$  – csomópontok kimenete,

$\varphi_i$  – normalizált kimenetek.

Az aktivációs függvény típusának a megválasztása attól függ, hogy a neuron kimenete milyen értékeket vehet fel. Ha a kimenet egy bináris érték, aktivációs függvénynek küszöbfüggvényt vagy lépcsőfüggvényt választunk, 0, 1 értékek esetében küszöbfüggvényt, -1, 1 értékek esetében pedig lépcsőfüggvényt. Ha a kimenet folytonos érték, akkor egy folytonos aktivációs függvényt alkalmazunk, annak függvényében, hogy a kimenet csak pozitív, vagy pozitív/negatív értékeket is felvehet. Csak pozitív értékek esetében szigmoid függvényt, míg pozitív-negatív értékek esetében tangens hiperbolikus függvényt alkalmazunk.

A szigmoid, tangens hiperbolikus aktiváló függvényeket a kevés réteget tartalmazó neuronhálókban alkalmaznak. Csak lineáris aktivációs függvény nem alkalmazható egy neuronhálóban, mert csak lineáris kimenetet eredményez. Az egyenirányító karakterisztika alapú aktivációs függvényeket előnyösebb alkalmazni a mély neuronhálókban, mivel nem eredményezik a gradinesből származó neuronelhalást.