Szoftver mély neuronhálók alkalmazásához

7. előadás

Kovács Bálint, Varga Viktor
ELTE IK Mesterséges Intelligencia Tanszék

Előző órán - Felügyelt tanulás

Adott: A tanítóminta (training set), input-címke párok halmaza

$$\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$$

$$x \in \boldsymbol{X} \subset \mathbb{R}^n,\ y \in \boldsymbol{Y} \subset \mathbb{R}^k$$

Feladat: A címke (az elvárt output) minél jobb becslése az inputból.

Azaz, keresünk olyan h_θ függvényt (hipotézisfüggvényt), melyre:

$$h_\theta(x) = \hat{y} \approx y$$

Eddig - A felügyelt tanulás két fő feladata

Regresszió: Folytonos értékű címke becslése

$$|Y| = \infty$$

Példa: Lakosság száma → Autók száma
Portré → Életkor

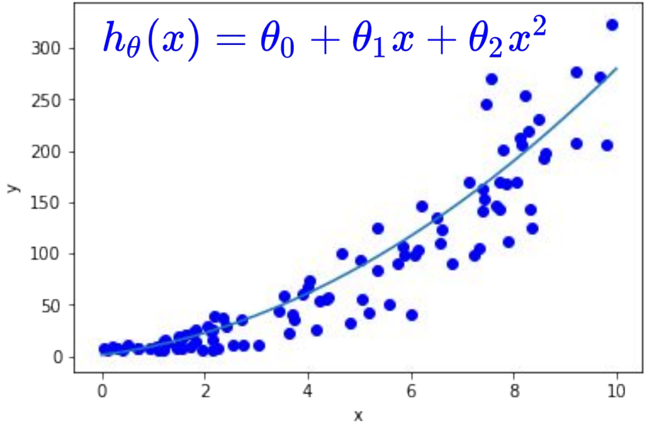
Klasszifikáció: Diszkrét értékű címke becslése

$$|Y| < \infty$$

Példa: Lakosság száma → Város, vagy falu?
Portré → Foglalkozás

Előző órán - Polinomiális regresszió

Hipotézisfüggvény:



Egyváltozós:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots$$

Többváltozós, pl:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2 + \dots$$

Költségfüggvény (MSE marad):

$$J(\theta) = \frac{1}{2m} \sum_{j=1}^{m} (\overbrace{h_\theta(x^{(j)})}^{\hat{y}^{(j)}} - y^{(j)})^2$$

Előző órán - Polinomiális regresszió

Hipotézisfüggvény:

Egyváltozós:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots$$

Többváltozós, pl:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2 + \ldots$$

Ne feledjük: $x^{(j)}, y^{(j)}$ mintaelemek adottak a tanítóhalmazban, tehát konstansnak tekinthetők.
A hipotézis függvény a theta paraméterek szerint továbbra is lineáris, így a költségfüggvény kvadratikus, konvex → a gradiensmódszer globális minimumot talál.

Költségfüggvény (MSE marad):

$$J(\theta) = \frac{1}{2m} \sum_{j=1}^{m} (\overbrace{h_\theta(x^{(j)})}^{\hat{y}^{(j)}} - y^{(j)})^2$$

Előző órán - Polinomiális regresszió

Vegyük észre, hogy a megoldás ugyanaz, mint lineáris regresszió esetében, így **lineáris regresszióként oldjuk meg**:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2 + \ldots$$

helyett

$$h_\theta(x) = \theta_0 + \theta_1 x_{1'} + \theta_2 x_{2'} + \theta_3 x_{3'} + \theta_4 x_{4'} + \theta_5 x_{5'} + \ldots$$

pl. $x_{4'} := x_2^2$

Az azonos változó különböző hatványaiból származó új változók a hatványok miatt egészen különböző nagyságrendből lehetnek, így érdemes **megoldás előtt azonos nagyságrendre hozni az új változókat** (feature scaling).

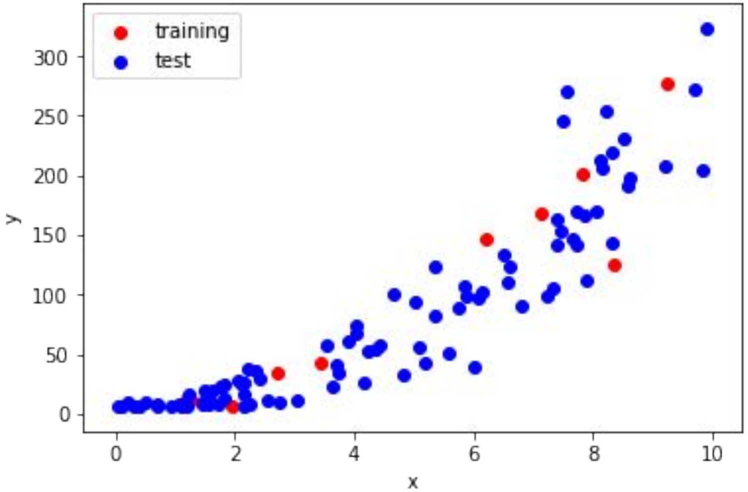
Előző órán - A minta felosztása

Modell életciklusa:

Ne használjuk tanításra!

1. A modell **betanítása** a **tanítóhalmazon** (training set)
2. A modell **kiértékelése** a **teszthalmazon** (test set)

A két halmaz diszjunkt!

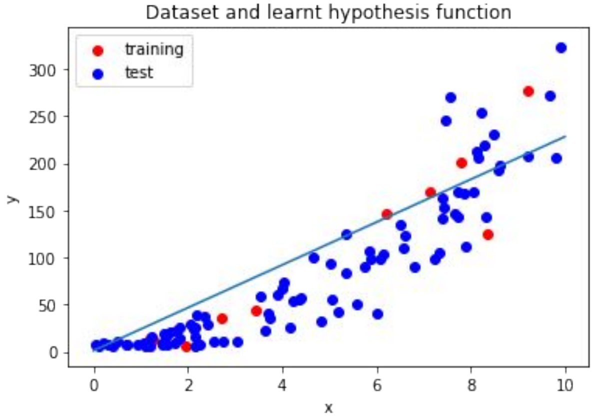


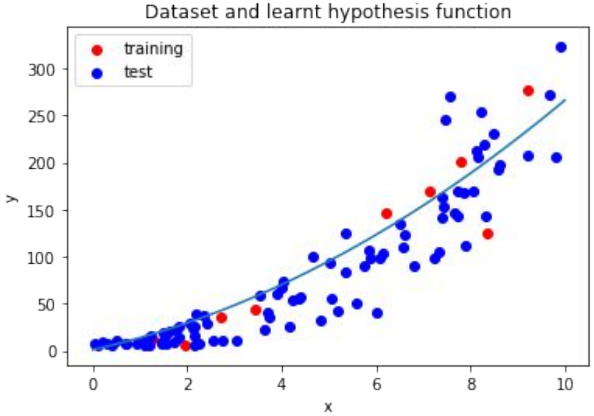
Előző órán - Polinomiális regresszió

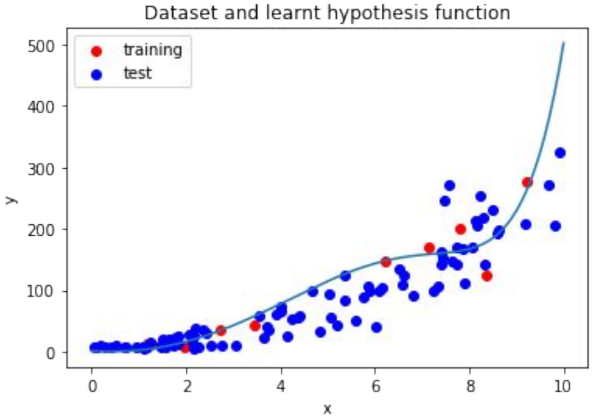
$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

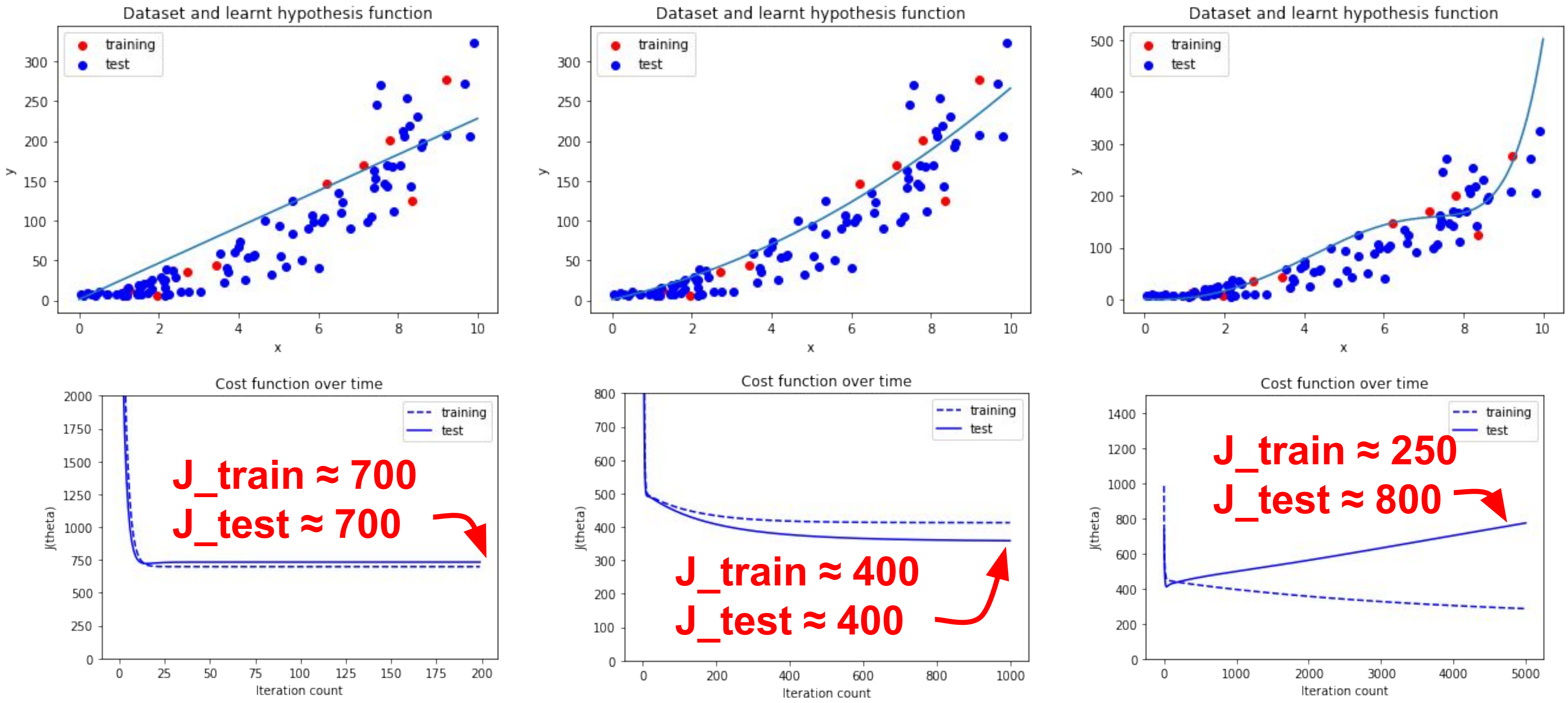
$$h_\theta(x) = \theta_0 + \theta_1 x + \dots + \theta_9 x^9$$







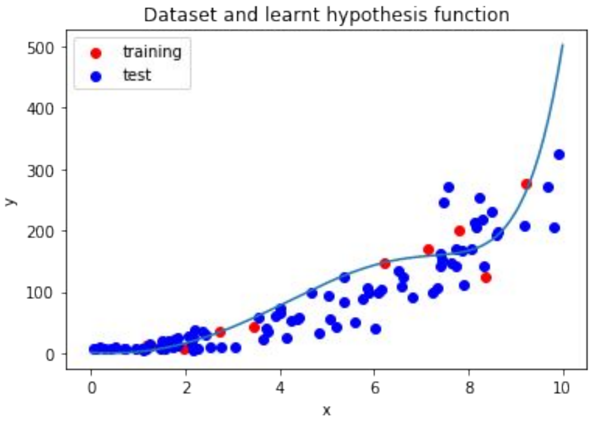
Előző órán - Alultanulás / “éppen jó” / túltanulás



Előző órán - Túltanulás

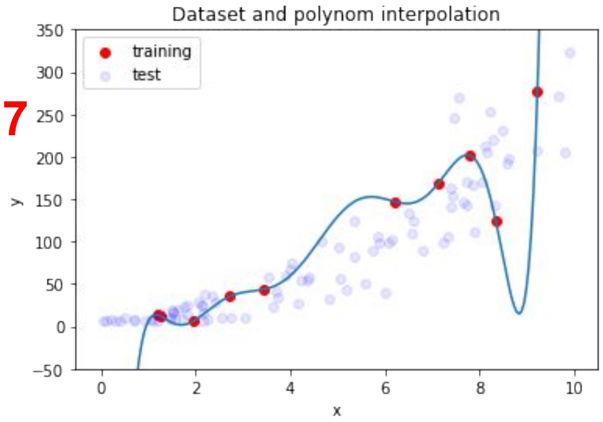
A pontos polinom illesztés (polinom interpoláció) a túltanulás extrém eseteként is értelmezhető:

- A **tanítóhalmaz** elemein a hipotézisfüggvény **hiba nélkül** becsül
- Ennek ellenére a **teszthalmazon hatalmas** a becslés **hibája**

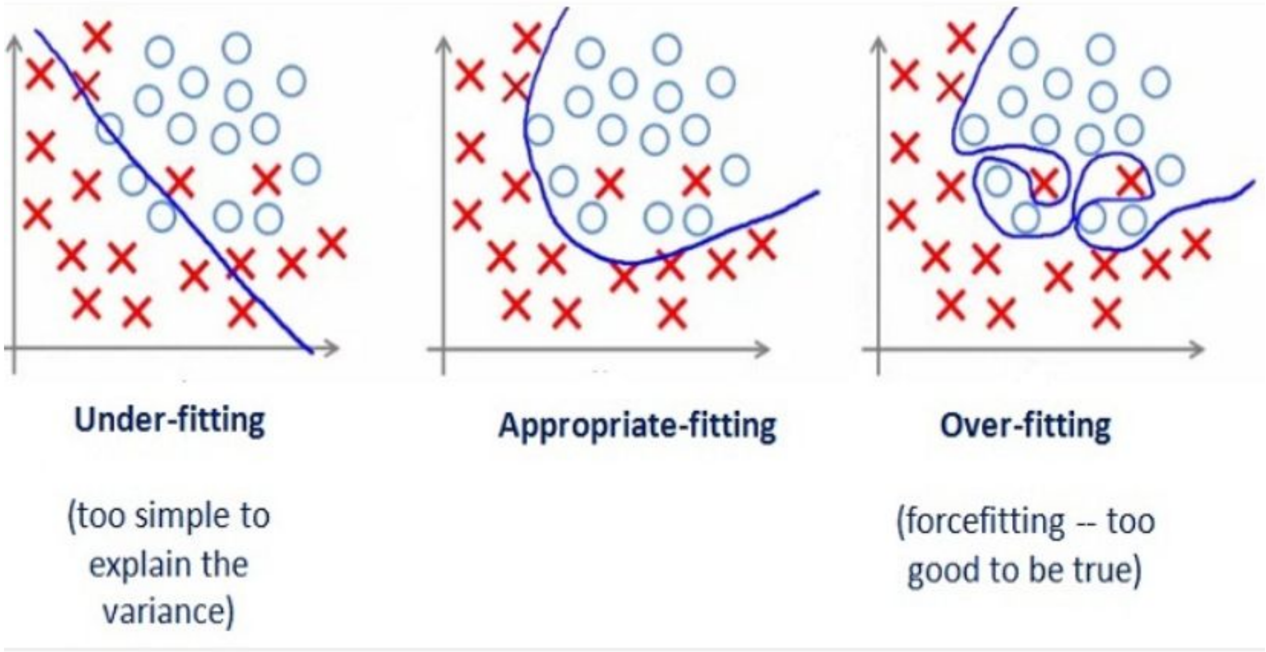


J_train ≈ 250
J_test ≈ 800

J_train = 0
J_test ≈ 252507



Előző órán - Alul- és túltanulás klasszifikáció esetén



Kép forrása: Andrew Ng Machine learning video series

Előző órán - Alultanulás (underfitting) és kezelése

Alultanulás (underfitting): A modell komplexitása túl kicsi ahhoz, hogy a címkéket jól közelítsük az inputból.

Kezelése: Bonyolultabb modell szükséges a becslési hiba csökkentéséhez.

Például lineáris regresszió helyett mély neuronháló...

Előző órán - Túltanulás (overfitting) és kezelése

Túltanulás (overfitting): A modell elég összetett ahhoz, hogy pontosan rátanuljon a tanítóhalmaz egyes elemeinek sajátosságaira, elvesztve az általánosítóképességét. A teszthalmazon a túltanult modell gyengén teljesít.

Kezelése:

- Használjunk egyszerűbb modellt (pl. kevesebb paraméter)!
- Szerezzünk be több tanítóadatot!
- *Regularizáció (pl.* $||\theta||_2^2$ *tag a költségben - L2 reg.)*
- *Early stopping*

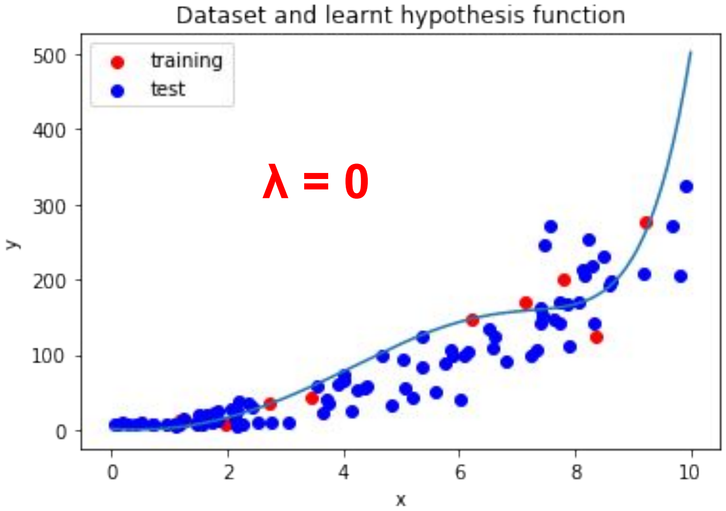
Előző órán - Túltanulás kezelése: L2 regularizáció

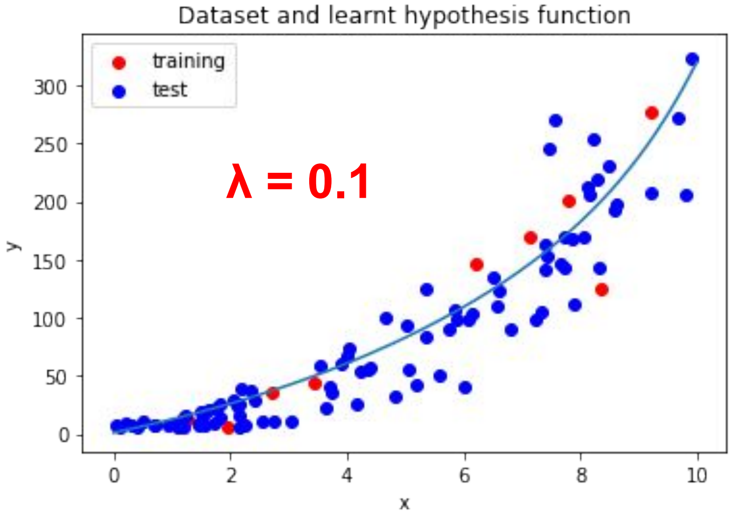
(L2) regularizáció:

$$h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_9 x^9$$

$$|X_{train}| = 10$$

$$J(\theta) = \frac{1}{2m}\sum_{j=1}^{m}(h(x)^{(j)} - y^{(j)})^2 + \boxed{\lambda \sum_{i=1}^{n} \theta_i^2}$$

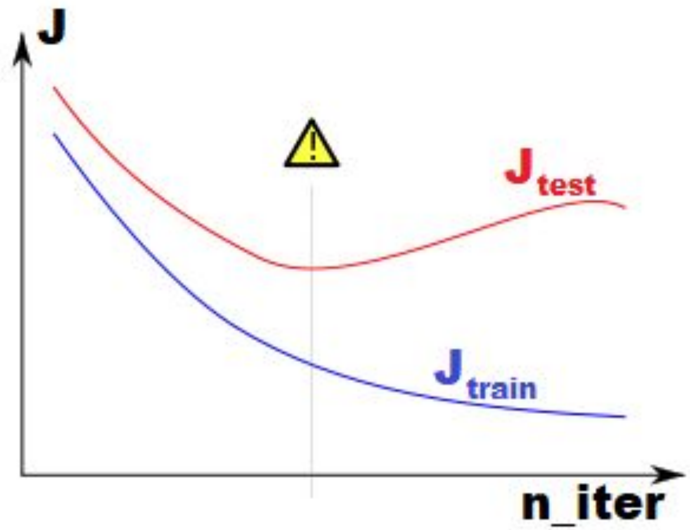




Előző órán - Túltanulás kezelése: early stopping

Early stopping: **break loop when J_test stopped decreasing**

$$
\begin{aligned}
&\text{repeat until convergence} \quad \{ \\
&\quad \text{for i} \leftarrow 1 \ldots \text{n} \quad \{ \\
&\qquad grad_i = \frac{\partial}{\partial \theta_i} J(\theta) \\
&\quad \} \\
&\quad \text{for i} \leftarrow 1 \ldots \text{n} \quad \{ \\
&\qquad \theta_i = \theta_i - \alpha \, grad_i \\
&\quad \} \\
&\}
\end{aligned}
$$



Előző órán - Validációs halmaz

Probléma: az early stopping a teszthiba alapján választotta ki a végső modellt → hozzáigazítjuk a modellt a teszthalmazhoz

Ezt nem lenne szabad, ezért a mintát háromfelé osztjuk ezentúl:

Tanítóhalmaz, **validációs halmaz,** teszthalmaz

A validációs halmazt a **hiperparaméterek** optimalizálására használjuk:

- Pl. gradiens módszer iterációinak száma, tanulási ráta mérete, modellarchitektúra (polinom foka, vagy neuronháló rétegeinek száma)

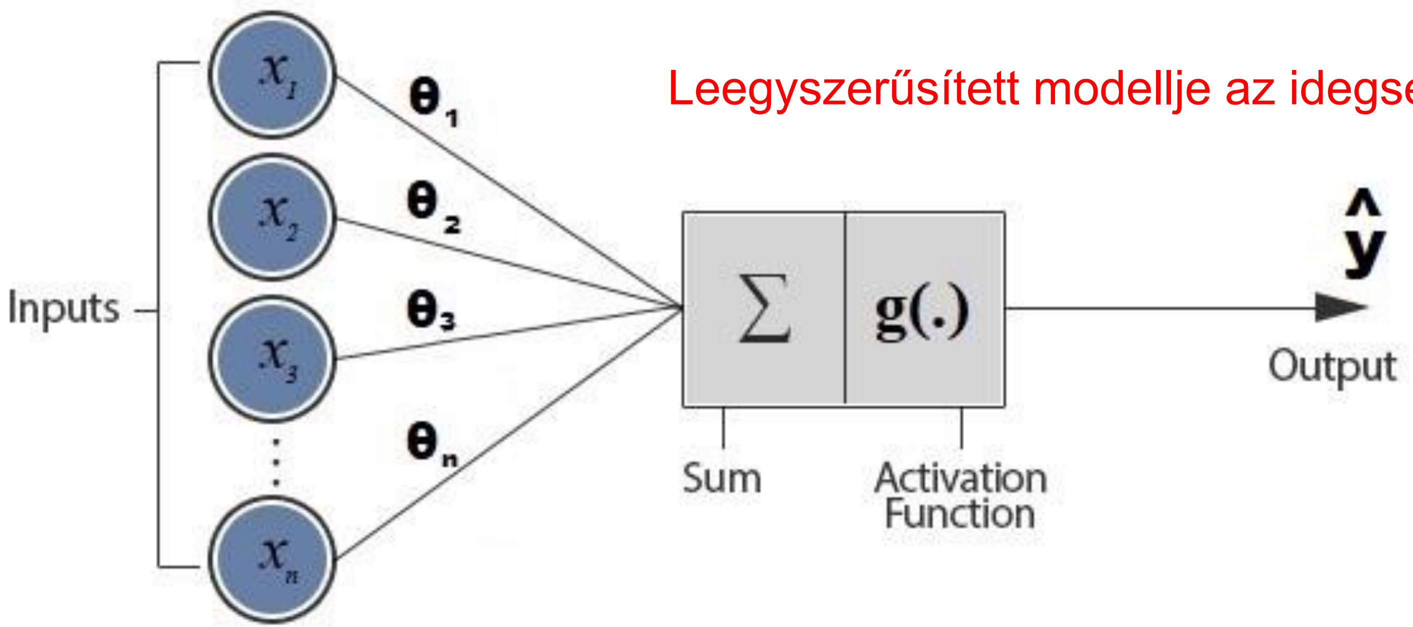
Előző órán - A betanítás folyamata

Feladat: $$\psi^*, \theta^* = argmin_\psi \; argmin_\theta \; J_\psi(\theta)$$

1) ψ hiperparaméter konfiguráció választása
2) A modell paramétereinek optimalizálása:
   θ^* keresése gradiensmódszerrel a **tanítóhalmazon**
3) Betanított modell kiértékelése a **validációs halmazon**, GOTO 1

Végül: a legjobbnak talált ψ hiperparaméterekkel betanított modell kiértékelése a **teszthalmazon**

Előző órán - A mesterséges neuron modell

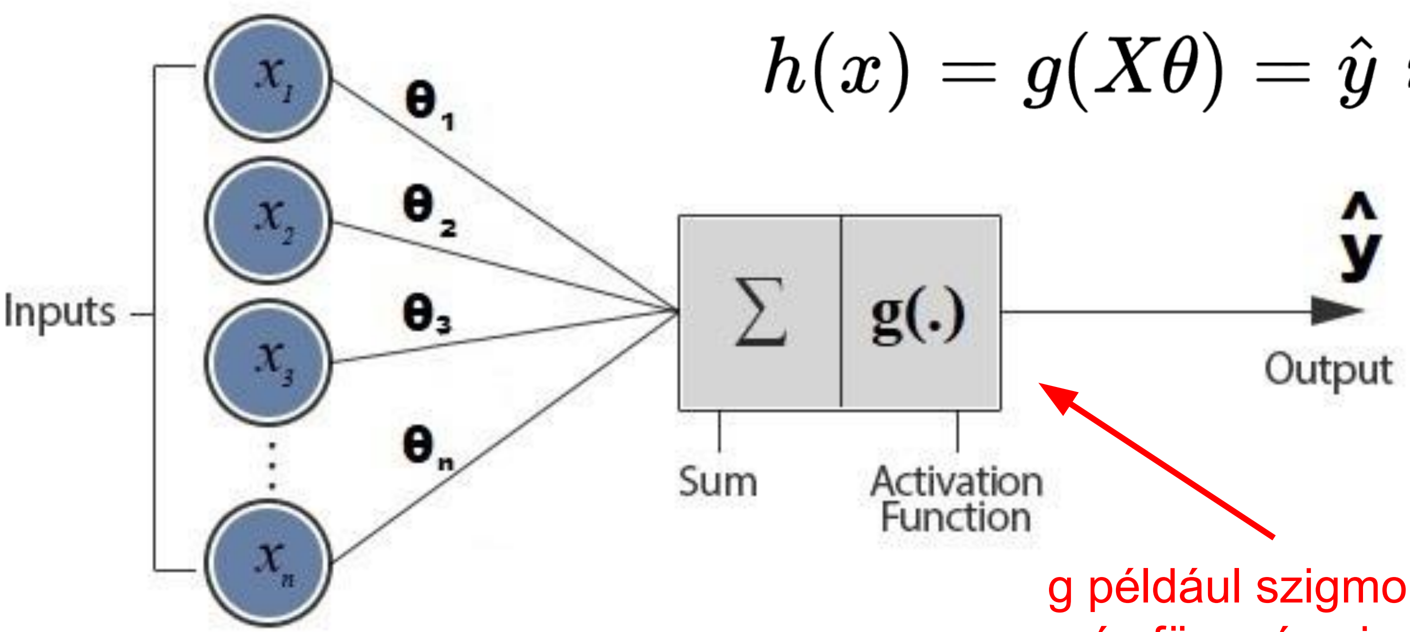


Leegyszerűsített modellje az idegsejteknek.

Mire hasonlít?

Előző órán - A mesterséges neuron modell

$$h(x) = g(X\theta) = \hat{y} \approx y$$

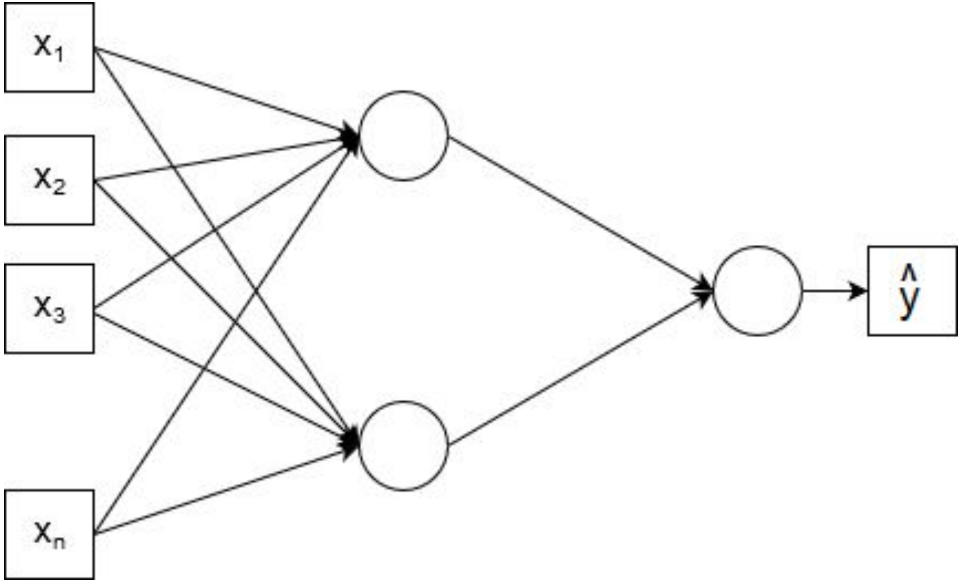


g például szigmoid, de más függvény is lehet

A mesterséges neuron egy (többváltozós) logisztikus regresszió!

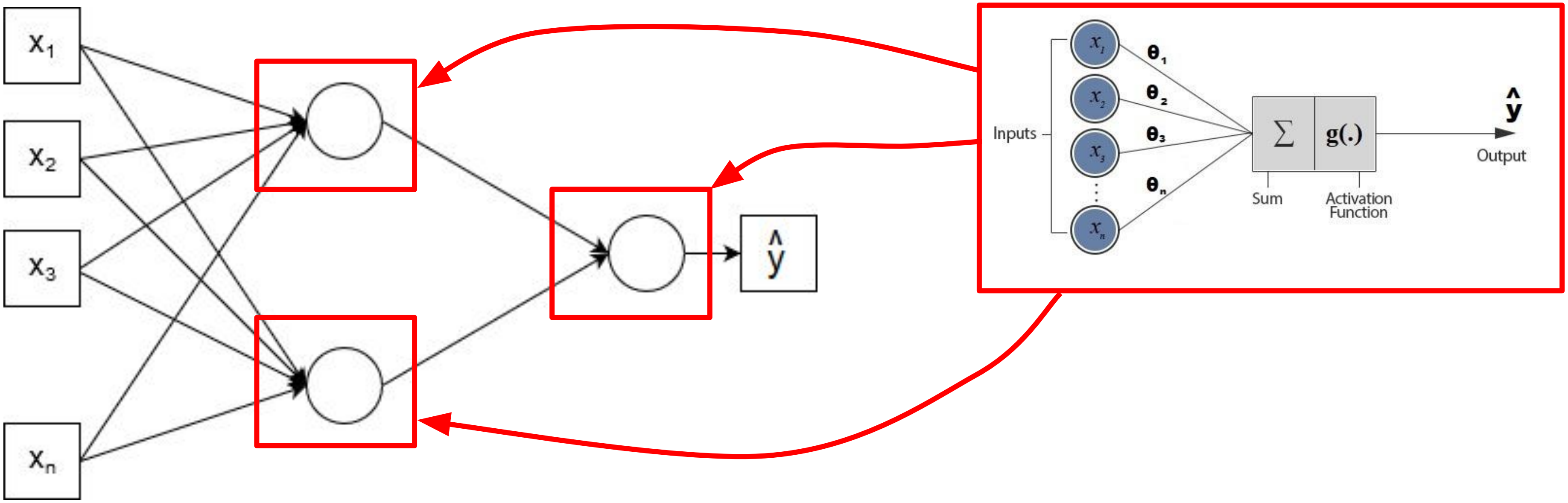
Multilayer Perceptron (MLP)

A mesterséges neuronhálók egyik alapvető fajtájának (Multilayer Perceptron, MLP) **építőkövei a mesterséges neuronok.**

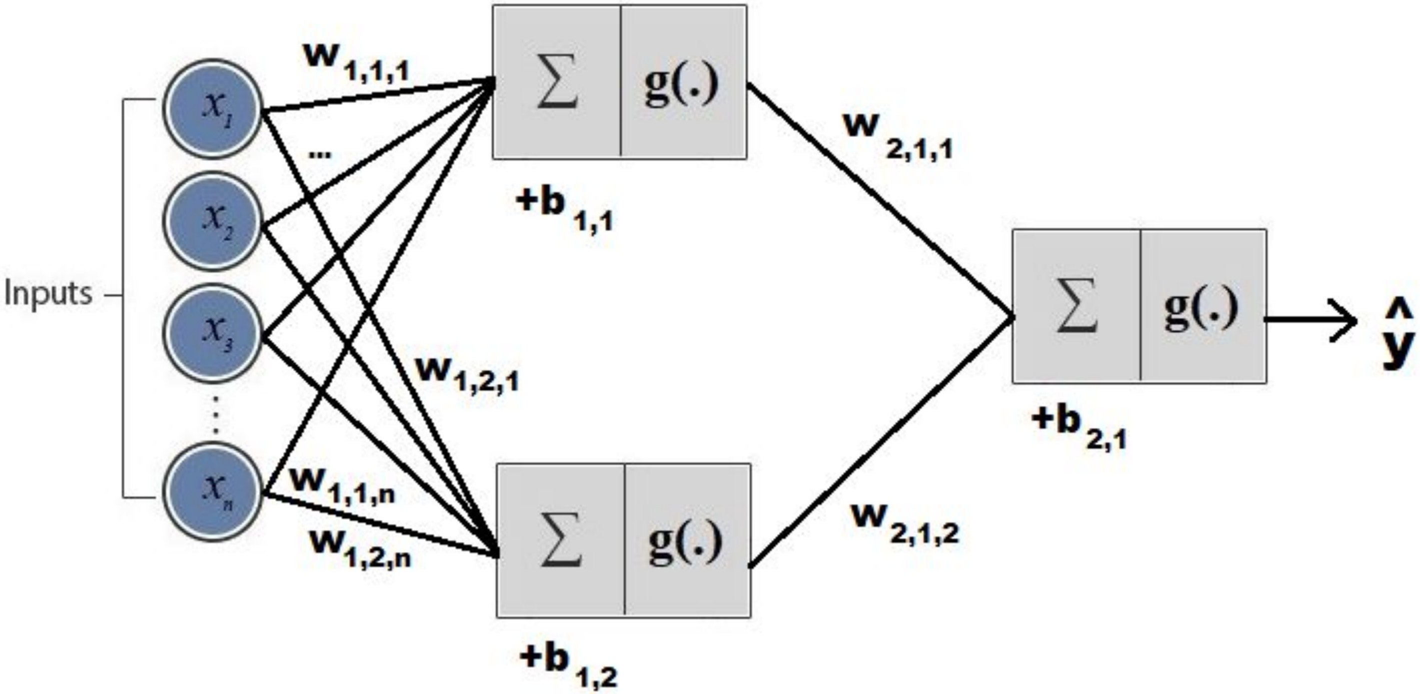


Multilayer Perceptron (MLP)

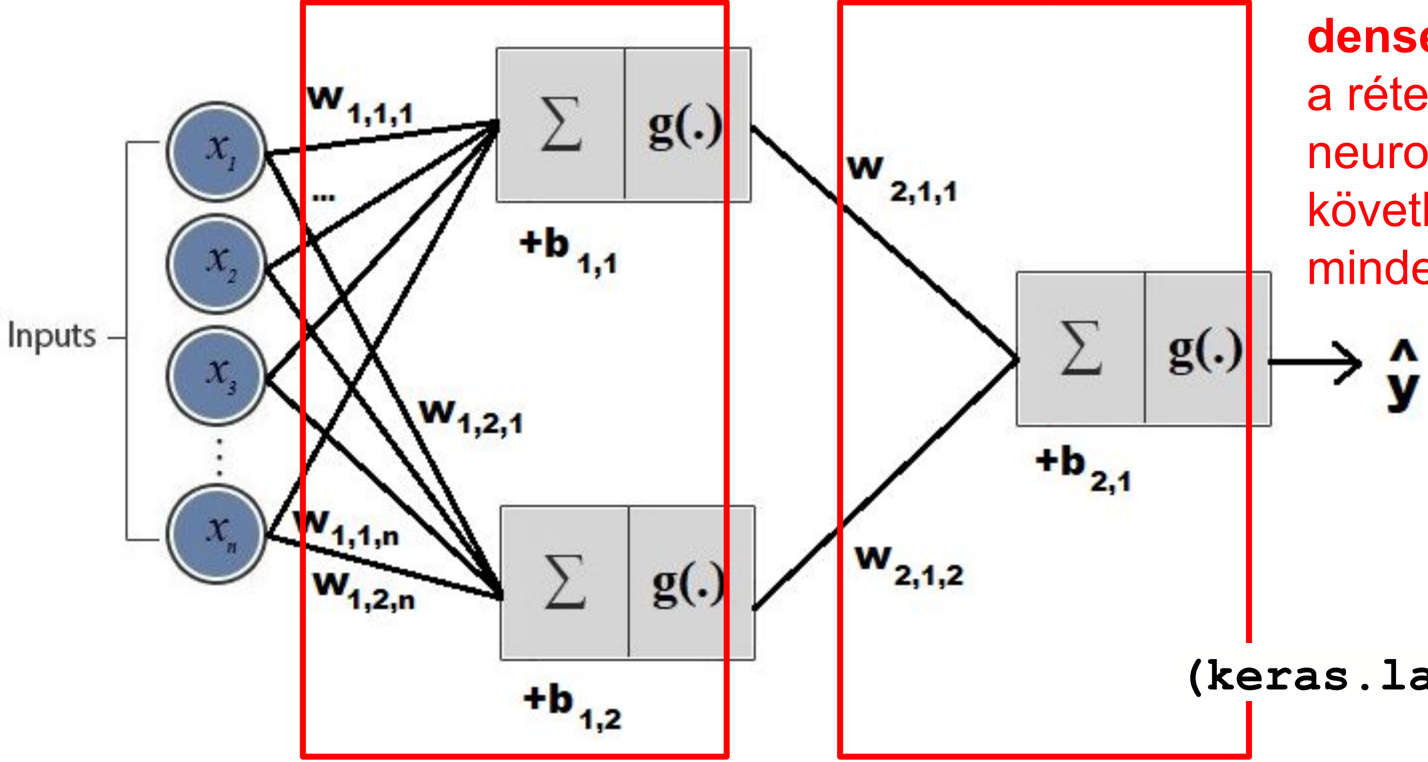
A mesterséges neuronhálók egyik alapvető fajtájának (Multilayer Perceptron, MLP) **építőkövei a mesterséges neuronok.**



Multilayer Perceptron (MLP)



Multilayer Perceptron (MLP)

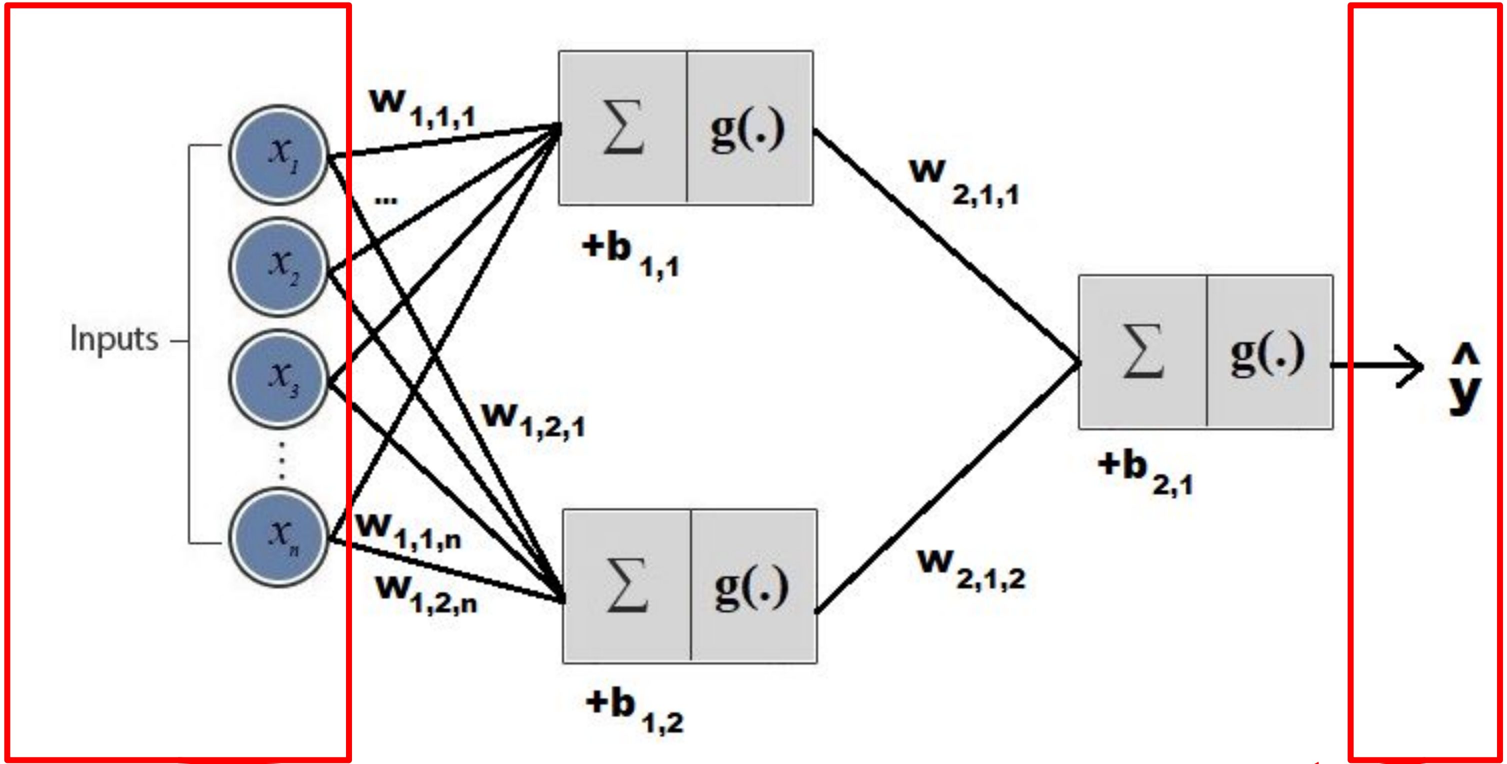


két **teljesen összekötött** (**fully connected** vagy **dense**) réteg: a réteg mindegyik neuronja összekötve a következő réteg mindegyik neuronjával

`(keras.layers.Dense)`

Multilayer Perceptron (MLP)

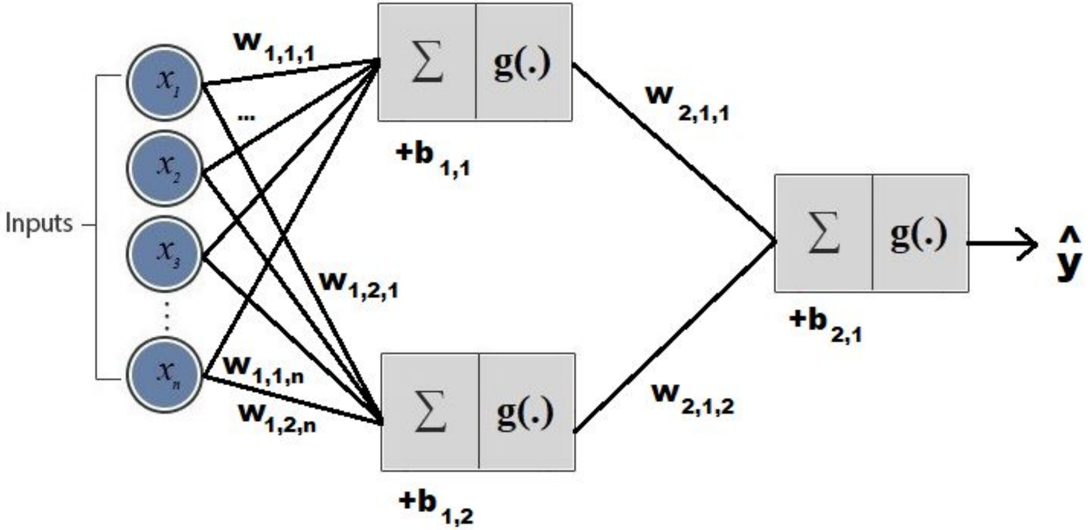
Input és output rétegekben nincsenek neuronok, vagy súlyok (paraméterek).



Input réteg

Output réteg

Multilayer Perceptron (MLP)



$W_1 \in \mathbb{R}^{2\times n}$

$b_1 \in \mathbb{R}^2$

$W_2 \in \mathbb{R}^{1\times 2}$

$b_2 \in \mathbb{R}^1$

Új jelölés: Theta a súlymátrixok és bias vektorok halmaza

$$\Theta = \{W_1, b_1, W_2, b_2\}$$

Multilayer Perceptron (MLP)

Súlymátrixok és bias vektorok mérete általánosan:

$$\boldsymbol{W_k} \in \mathbb{R}^{S_k \times S_{k-1}}$$

$$\boldsymbol{b_k} \in \mathbb{R}^{S_k}$$

ahol $\boldsymbol{S_k}$ a k. rétegben található neuronok száma.

$$S_0 := n$$

$$x \in \mathbb{R}^n$$

Multilayer Perceptron (MLP)

Súlymátrixok és bias vektorok mérete általánosan:

$$W_k \in \mathbb{R}^{S_k \times S_{k-1}}$$

$$b_k \in \mathbb{R}^{S_k}$$

Jelölés: Mátrixos alakban a paramétereket típikusan **W**-vel (weight (súly) matrix) és **b**-vel (bias vector) jelöljük. Ezek a lineáris és logisztikus regressziónál használt **θ** paramétereknek felelnek meg (**b** a konstans tagot, **θ_0** paramétert helyettesíti)

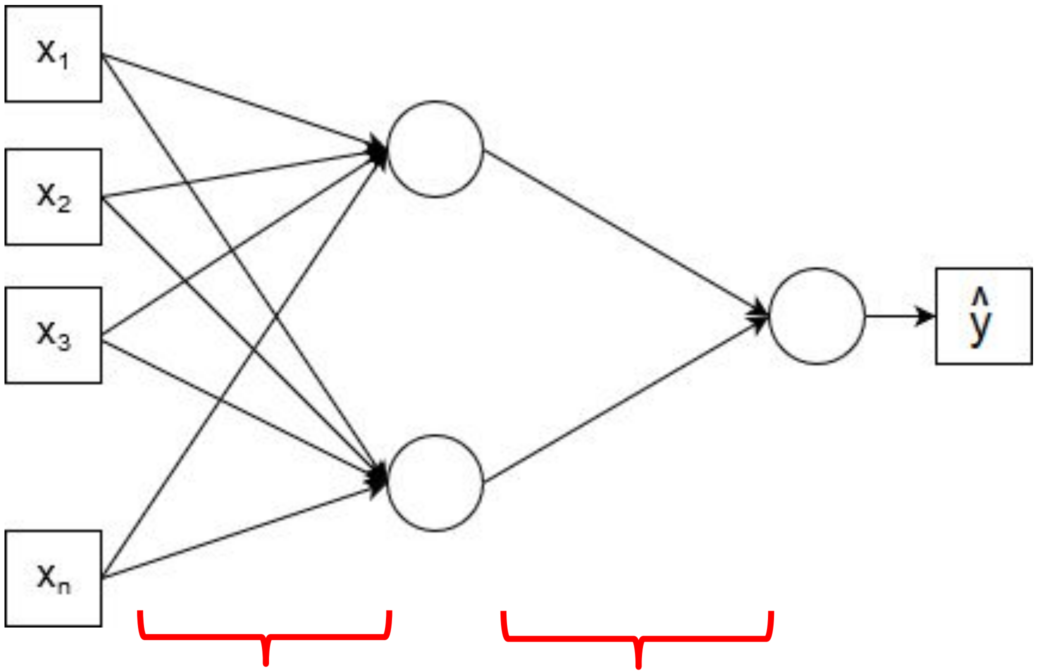
ahol S_k a k. rétegben található neuronok száma.

$$S_0 := n$$

$$x \in \mathbb{R}^n$$

S_0 az input réteg mérete, azaz az input változók száma

Multilayer Perceptron (MLP)



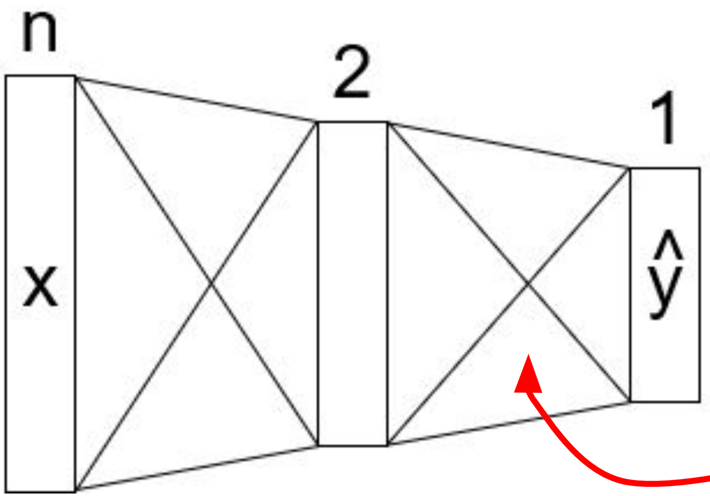
$$W_1 \in \mathbb{R}^{2\times n} \qquad W_2 \in \mathbb{R}^{1\times 2}$$

$$b_1 \in \mathbb{R}^{2} \qquad b_2 \in \mathbb{R}^{1}$$

Egyszerűsített ábrázolás

$$\Theta = \{W_1, b_1, W_2, b_2\}$$

Multilayer Perceptron (MLP)



Egészen egyszerű ábrázolás...

A **teljesen összekötött réteg** egyik elterjedt jelölése

$$W_1 \in \mathbb{R}^{2\times n} \qquad W_2 \in \mathbb{R}^{1\times 2}$$

$$b_1 \in \mathbb{R}^{2} \qquad b_2 \in \mathbb{R}^{1}$$

$$\Theta = \{W_1, b_1, W_2, b_2\}$$

Multilayer Perceptron (MLP)

Kétrétegű neuronháló hipotézisfüggvénye:

$$h(x) = g_2(W_2 \underbrace{g_1(W_1 x + b_1)}_{\text{Az első réteg outputja}} + b_2) = \hat{y} \approx y$$

Költségfüggvények egyelőre maradnak:

- **Klasszifikáció:** logistic loss
- **Regresszió:** MSE

Multilayer Perceptron (MLP)

Kétrétegű neuronháló hipotézisfüggvénye:

$$h(x) = g_2(W_2\, \underbrace{g_1(W_1 x + b_1)}_{\text{Az első réteg outputja}} + b_2) = \hat{y} \approx y$$

Aktivációs függvények:

- **Klasszifikáció:** szigmoid, ahogy logisztikus regressziónál
- **Regresszió:** ???

Multilayer Perceptron (MLP)

Kétrétegű neuronháló hipotézisfüggvénye:

$$h(x) = g_2(W_2 \underbrace{g_1(W_1 x + b_1)}_{\text{Az első réteg outputja}} + b_2) = \hat{y} \approx y$$

Aktivációs függvények:

- **Klasszifikáció:** szigmoid, ahogy logisztikus regressziónál
- **Regresszió:** ??? **Lineáris regresszióban nem használtunk aktivációs függvényt** (nemlinearitást)... Ezt követve, regresszió esetén nem kellene aktivációs függvényt tenni a neuronjainkba...

Multilayer Perceptron (MLP)

Jó-e az alábbi hipotézisfüggvény regresszióhoz?

$$h(x) = W_2 \underbrace{\left(W_1 x + b_1\right)}_{\text{Az első réteg outputja}} + b_2) = \hat{y} \approx y$$

Multilayer Perceptron (MLP)

Jó-e az alábbi hipotézisfüggvény regresszióhoz?

$$h(x) = W_2\,\underbrace{(W_1 x + b_1)}_{\text{Az első réteg outputja}} + b_2) = \hat{y} \approx y$$

Sok értelme nincs, **a kifejezőereje egyetlen lineáris rétegnek felel meg:**

$$W_2(W_1 x + b_1) + b_2 = (W_2 W_1)x + (W_2 b_1 + b_2)$$

Multilayer Perceptron (MLP)

Jó-e az alábbi hipotézisfüggvény regresszióhoz?

$$h(x) = W_2 \, \underbrace{(W_1 x + b_1)}_{\text{Az első réteg outputja}} + b_2) = \hat{y} \approx y$$

Sok értelme nincs, **a kifejezőereje egyetlen lineáris rétegnek felel meg:**

$$W_2(W_1 x + b_1) + b_2 = \overbrace{(W_2 W_1)}^{\in \mathbb{R}^{S_2 \times S_0}} x + (W_2 b_1 + b_2)$$

Több lineáris leképezés kompozíciója lineáris → **nemlinearitás nélkül a neuronháló kifejezőereje azonos a lineáris regresszióéval...**

Multilayer Perceptron (MLP)

Kétrétegű neuronháló hipotézisfüggvénye:

$$h(x) = g_2(W_2 \underbrace{g_1(W_1 x + b_1)}_{\text{Az első réteg outputja}} + b_2) = \hat{y} \approx y$$

Regresszió esetén is a fenti hipotézisfüggvényt fogjuk használni.

Érdemes azonban g_2 -t (a legutolsó aktivációs függvényt) elhagyni.

Multilayer Perceptron (MLP)

Kétrétegű neuronháló hipotézisfüggvénye:

$$h(x) = g_2(W_2 \underbrace{g_1(W_1 x + b_1)}_{\text{Az első réteg outputja}} + b_2) = \hat{y} \approx y$$

Regresszió esetén is a fenti hipotézisfüggvényt fogjuk használni.

Érdemes azonban g_2 -t (a legutolsó aktivációs függvényt) elhagyni.

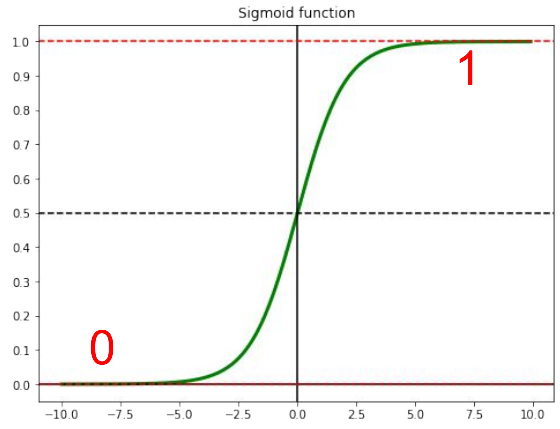
Hiszen, ha például g_2 szigmoid, a háló kimenete 0 és 1 közt lesz, ami pl. életkor becslésére alkalmatlan. **g_2 tehát regresszió esetén tipikusan identitásfüggvény.**

`(keras.activations.*)`

Aktivációs függvények

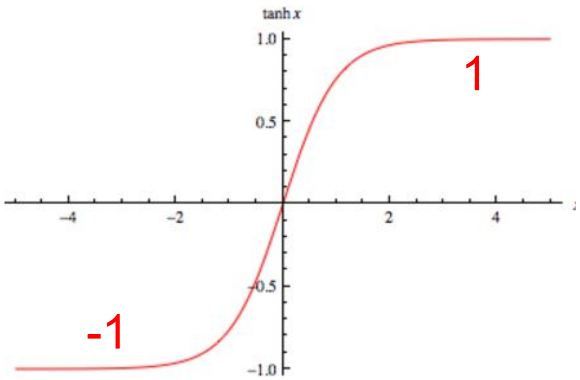
Gyakran használt aktivációs függvények:

sigmoid



$$g(z) = \frac{1}{1+e^{-z}}$$

tanh



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

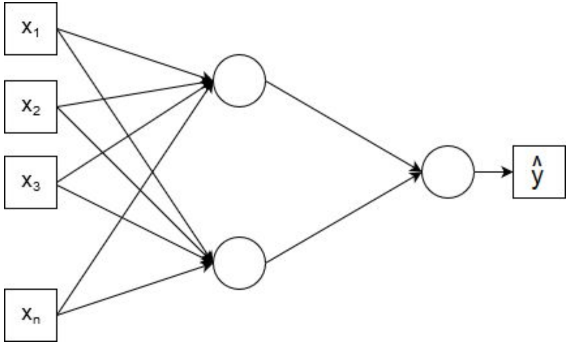
ReLU

(Rectified Linear Unit)

Gyorsan számolható és szinte mindig jól működik

$$g(z) = \mathrm{ReLU}(z) = \max(0, z)$$

Multilayer Perceptron (MLP)



Kétrétegű neuronháló hipotézisfüggvénye:

$$h(x) = g_2(W_2\, g_1(W_1 x + b_1) + b_2) = \hat{y} \approx y$$

Keras kód g_1 := ReLU, g_2 := identitás esetén

```
model = Sequential()
model.add(Dense(2, activation='relu', input_dim=n))
model.add(Dense(1, activation='linear'))
```

MLP betanítása

Gradiens módszert fogunk továbbra is használni…

$$
\begin{aligned}
&\text{repeat until convergence} \quad \{ \\
&\quad \text{for} \; \forall\, \theta \in \boldsymbol{\Theta} \quad \{ \\
&\qquad grad_\theta = \frac{\partial}{\partial \theta} J(\boldsymbol{\Theta}) \\
&\quad \} \\
&\quad \text{for} \; \forall\, \theta \in \boldsymbol{\Theta} \quad \{ \\
&\qquad \theta = \theta - \alpha \; grad_\theta \\
&\quad \} \\
&\}
\end{aligned}
$$

MLP betanítása

Az összes paraméter halmaza
(súlymátrixok, bias vektorok elemeit tartalmazza)

Gradiens módszert fogunk továbbra is használni…

$$
\begin{aligned}
&\text{repeat until convergence} \quad \{ \\
&\quad \text{for } \forall\, \theta \in \Theta \quad \{ \\
&\qquad grad_\theta = \frac{\partial}{\partial \theta} J(\Theta) \\
&\quad \} \\
&\quad \text{for } \forall\, \theta \in \Theta \quad \{ \\
&\qquad \theta = \theta - \alpha\, grad_\theta \\
&\quad \} \\
&\}
\end{aligned}
$$

Szerencsére nem kell kiszámolnunk kézzel a gradienseket. Ezt a Tensorflow **automatikus deriválási algoritmusa** elvégzi helyettünk...

Minden egyes paraméter szerint deriválnunk kell a költségfüggvényt.

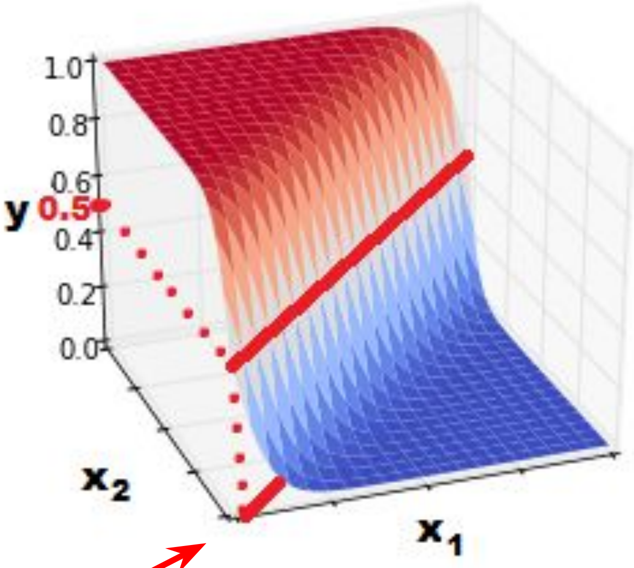
Mit tanul egy neuronháló?

Mit tanul egyetlen neuron (szigmoiddal)?

Mit tanul egy neuronháló?

Mit tanul egyetlen neuron (szigmoiddal)?

Egyetlen lineáris döntési felületet (hiszen egy logisztikus regresszióról van szó).



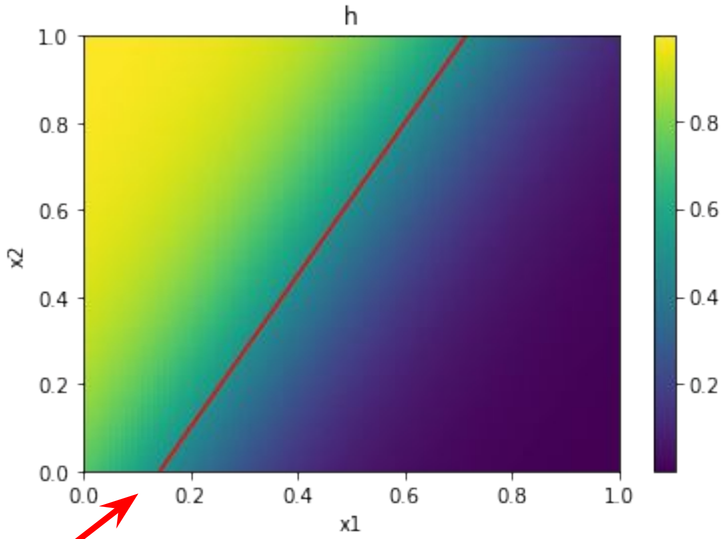
Döntési felület (decision boundary)

2 input változó (x_1, x_2) esetén ez egy egyenes az x_1, x_2 síkon.

Mit tanul egy neuronháló?

Mit tanul egyetlen neuron (szigmoiddal)?

Egyetlen lineáris döntési felületet (hiszen egy logisztikus regresszióról van szó).

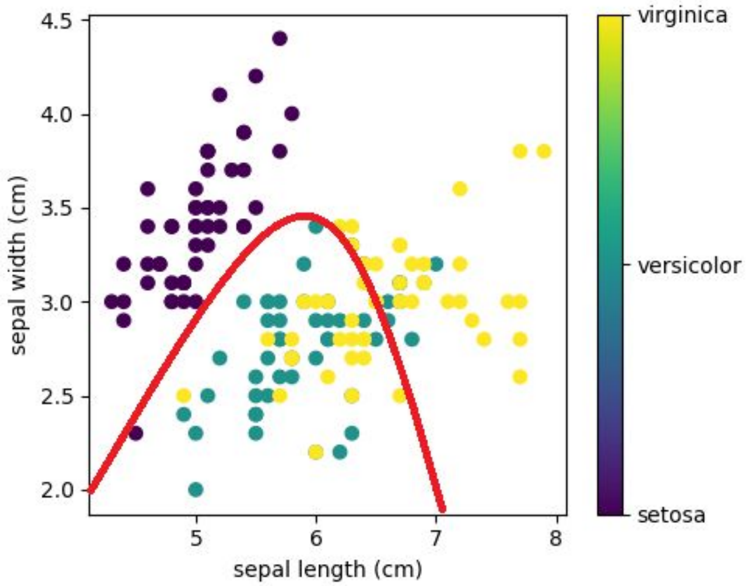


Ugyanez felülnézetből...

Döntési felület (decision boundary)

Mit tanul egy neuronháló?

Nem minden klasszifikációs problémát tudunk megoldani egy egyenes (hipersík) **tanulásával.**

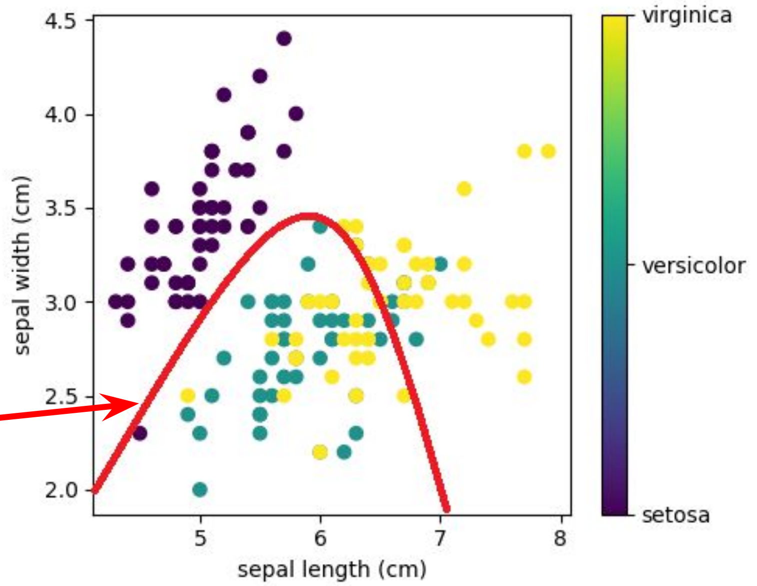


Mit tanul egy neuronháló?

Nem minden klasszifikációs problémát tudunk megoldani egy egyenes (hipersík) **tanulásával.**

IRIS adatbázis: háromfajta virág klasszifikációja a fajták szerint a szirmok hossza és szélessége alapján.

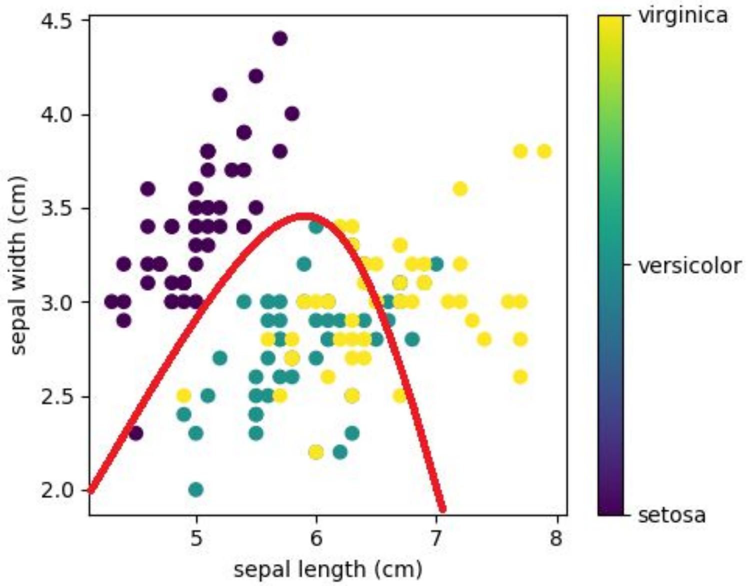
A "versicolor" fajta különválasztásához nem elég az egyenes…



Mit tanul egy neuronháló?

Nem minden klasszifikációs problémát tudunk megoldani egy egyenes (hipersík) **tanulásával.**

Hogyan választhatnánk mégis külön a "versicolor" kategóriát?



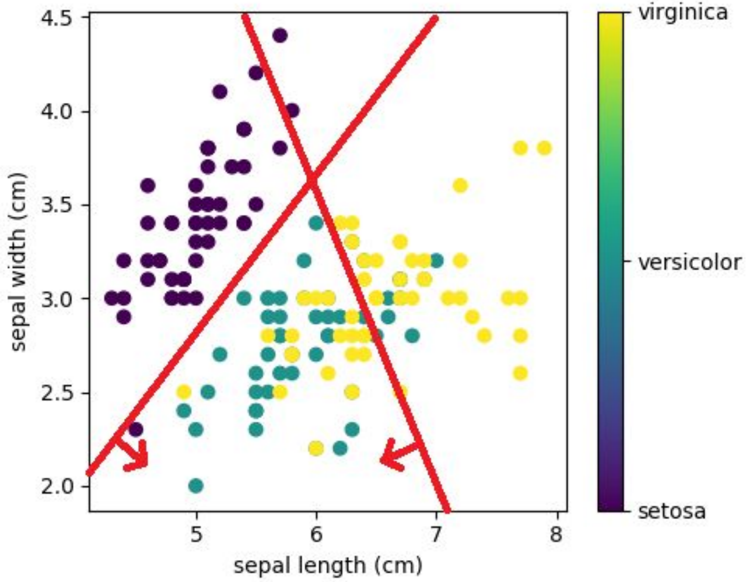
Mit tanul egy neuronháló?

Nem minden klasszifikációs problémát tudunk megoldani egy egyenes (hipersík) **tanulásával.**

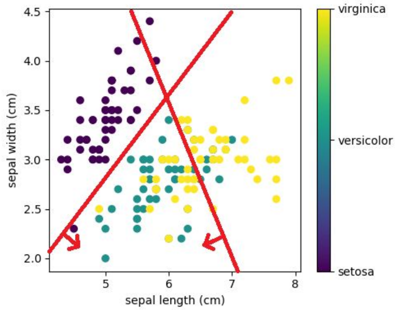
Hogyan választhatnánk mégis külön a "versicolor" kategóriát?

Esetleg két lineáris döntési felületettel és azok "össze-ÉSelésével" (AND)

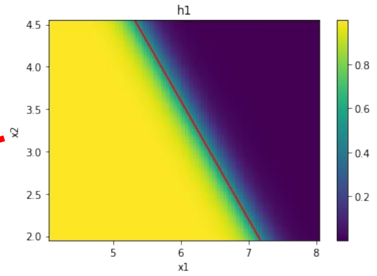
Hogyan reprezentálhatja ezt egy neuronháló?

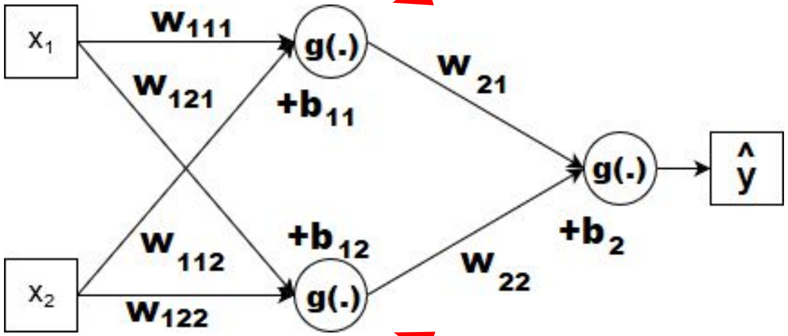


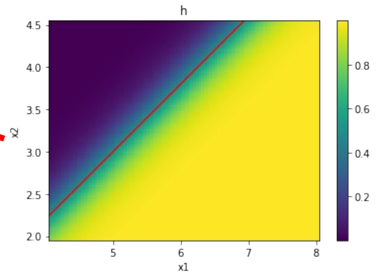
Neuronháló kifejezőereje - példa



$$w_{111} = -7,\ w_{121} = -5,\ b_{11} = 60$$

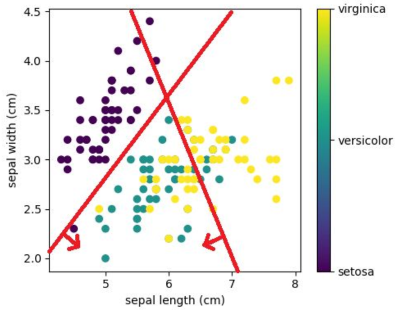




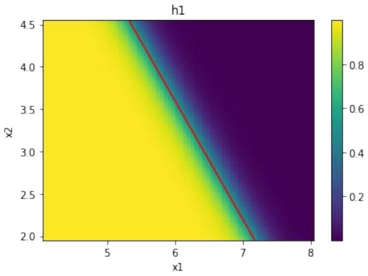


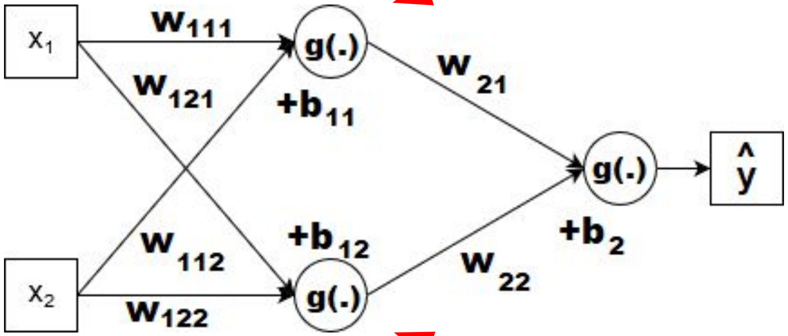
$$w_{112} = 4,\ w_{122} = -5,\ b_{12} = -5$$

Neuronháló kifejezőereje - példa

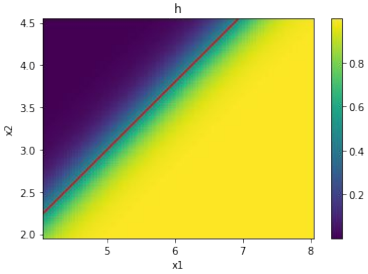


$$w_{111} = -7,\ w_{121} = -5,\ b_{11} = 60$$





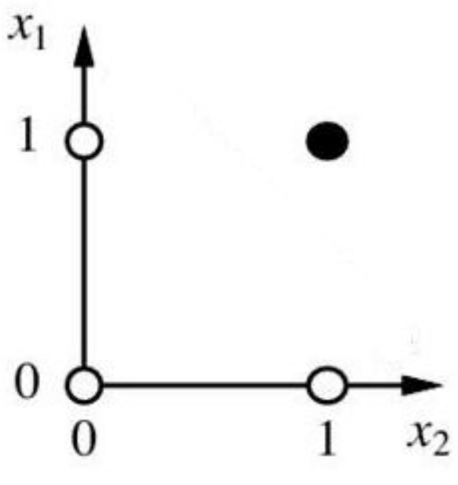
Azt szeretnénk, ha a második rétegbeli egyetlen neuronunk ott adna 1-et, ahol mindkét első rétegbeli neuron 1-et ad kimenetként. **Azaz, megközelítőleg egy AND műveletre lenne szükségünk...**



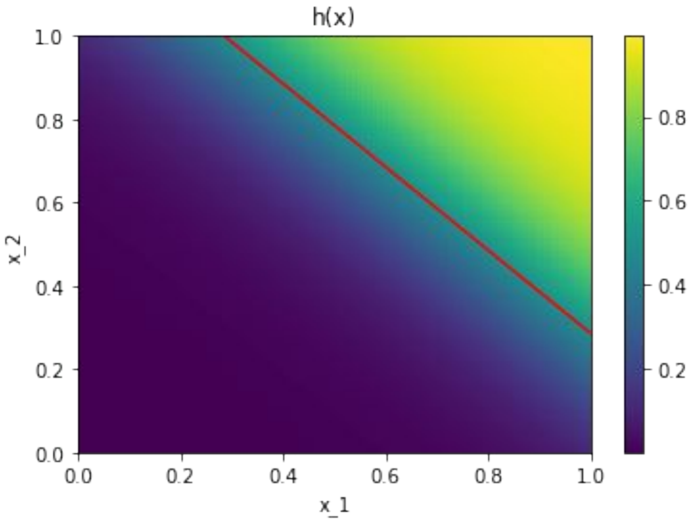
$$w_{112} = 4,\ w_{122} = -5,\ b_{12} = -5$$

Neuronháló kifejezőereje - példa

Bináris logikai függvények közelítése: x1 AND x2



x_1 and x_2



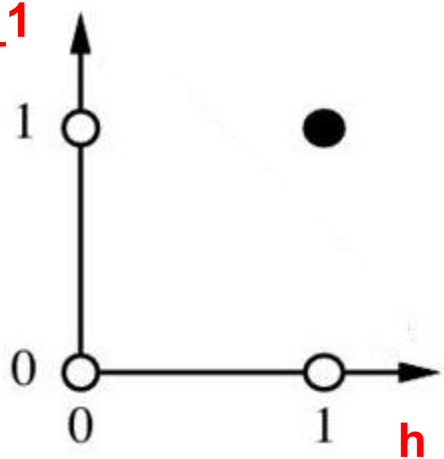
$$w_1 = 7,\ w_2 = 7,\ b = -9$$

Neuronháló kifejezőereje - példa

x_1, x_2 helyett itt a két első rétegbeli neuron kimenete lesz az input, (pl. h_1, h_2).

Bináris logikai függvények közelítése: x1 AND x2

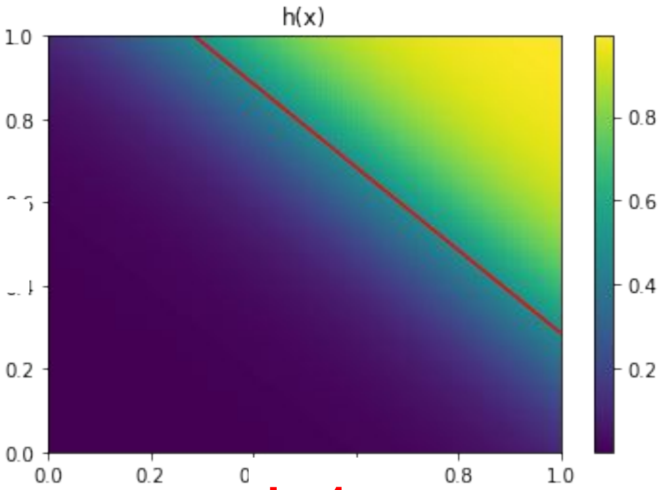
h_1



h_2

h_1 and h_2

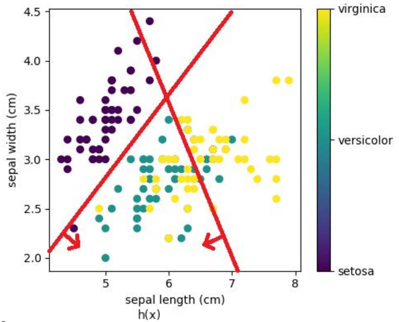
h_2



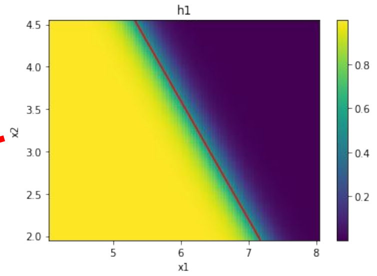
h_1

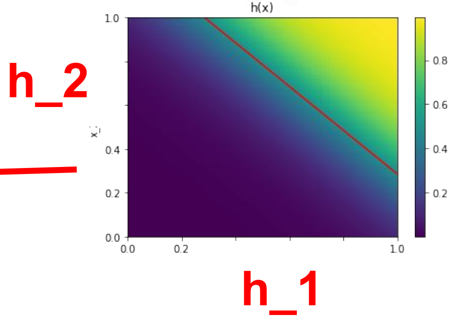
$$w_1 = 7,\ w_2 = 7,\ b = -9$$

Neuronháló kifejezőereje - példa

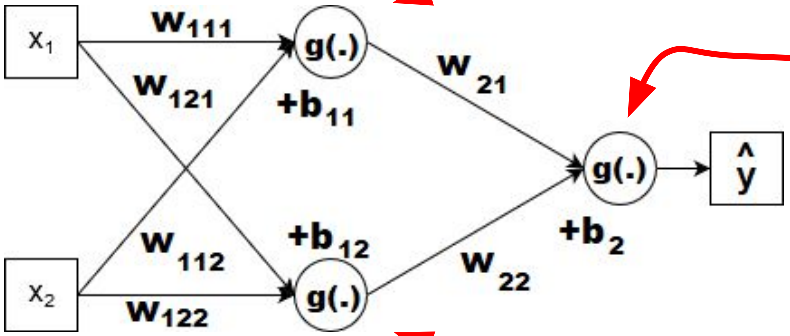


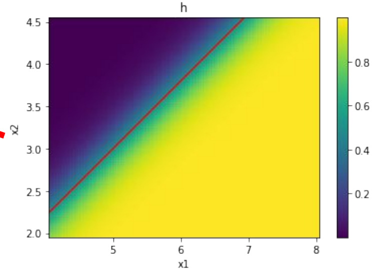
$$w_{111} = -7,\ w_{121} = -5,\ b_{11} = 60$$





$$w_{21} = 7,\ w_{22} = 7,\ b_2 = -9$$

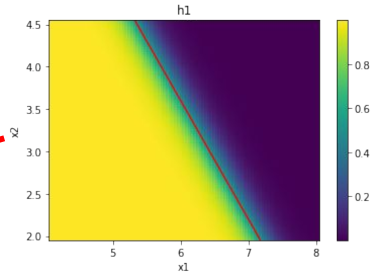


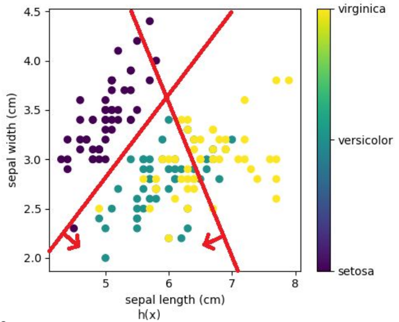


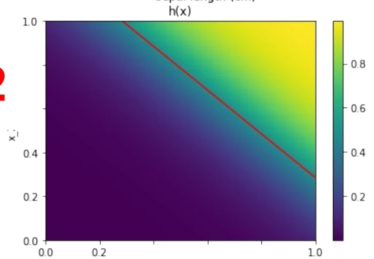
$$w_{112} = 4,\ w_{122} = -5,\ b_{12} = -5$$

Neuronháló kifejezőereje - példa

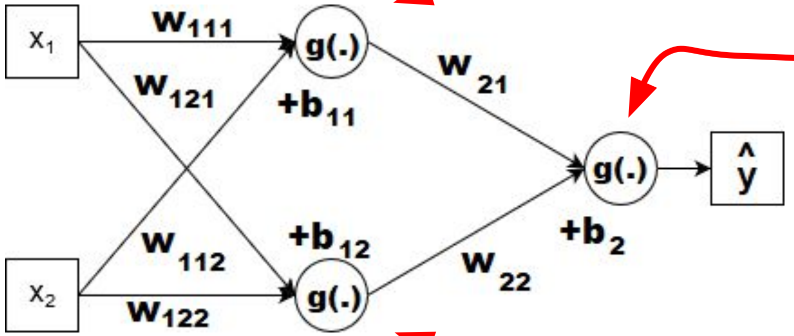
$$w_{111} = -7,\ w_{121} = -5,\ b_{11} = 60$$





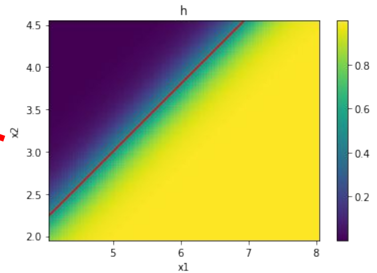


h_2

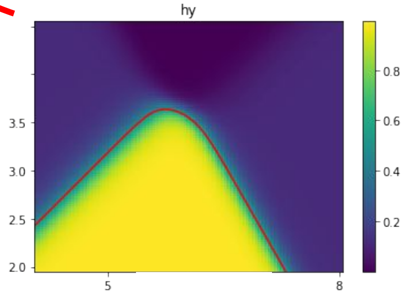


$$w_{21} = 7,\ w_{22} = 7,\ b_2 = -9$$

h_1



x_2

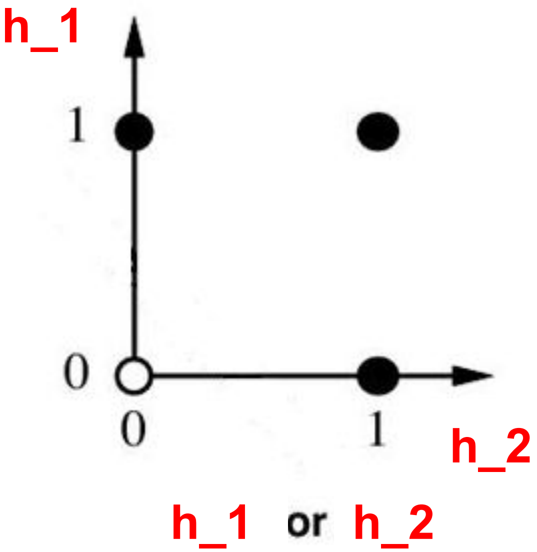


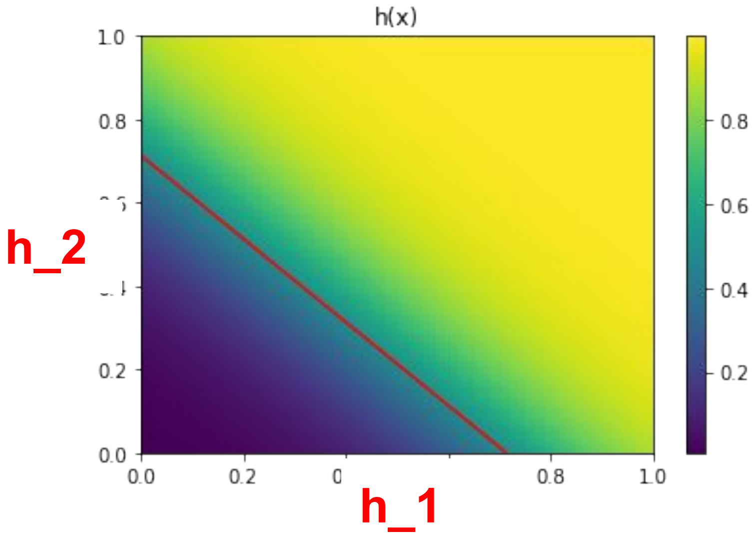
x_1

$$w_{112} = 4,\ w_{122} = -5,\ b_{12} = -5$$

Neuronháló kifejezőereje - példa

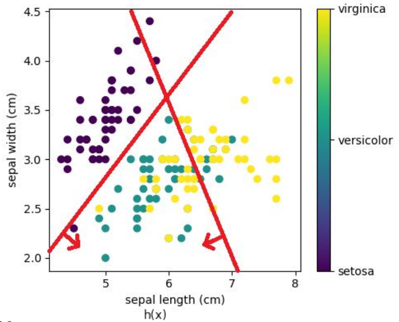
Bináris logikai függvények közelítése: x1 OR x2



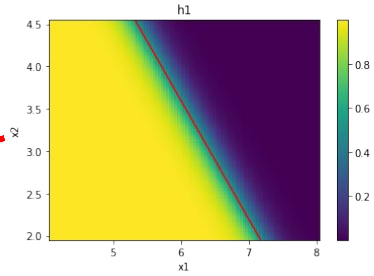


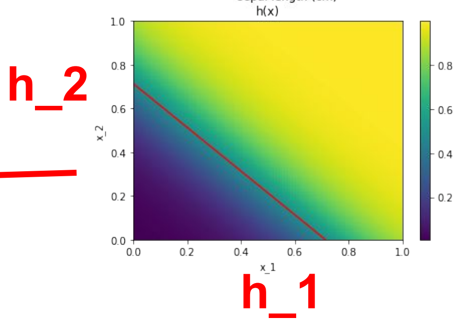
$$w_1 = 7,\ w_2 = 7,\ b = -5$$

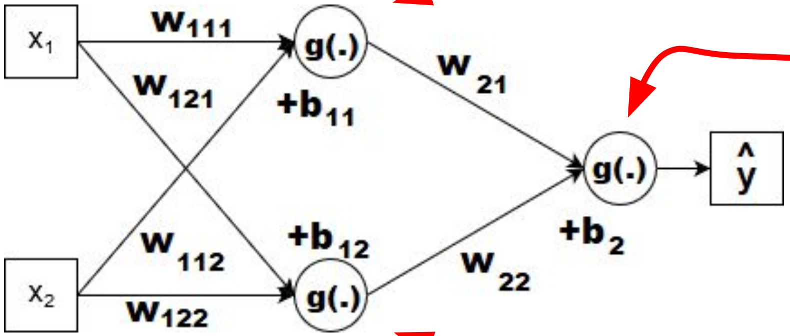
Neuronháló kifejezőereje - példa



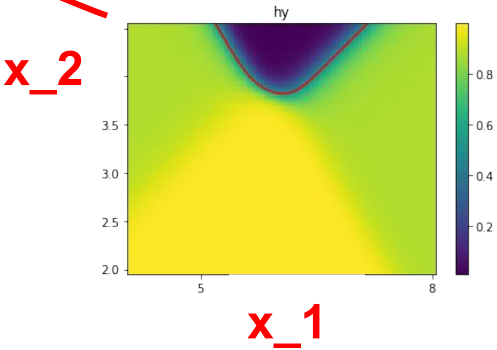
$$w_{111} = -7,\ w_{121} = -5,\ b_{11} = 60$$

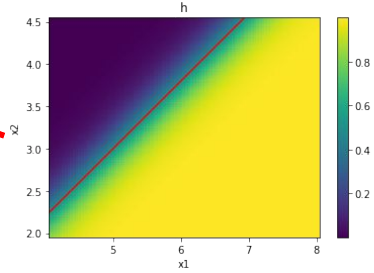






$$w_{21} = 7,\ w_{22} = 7,\ b_2 = -5$$





$$w_{112} = 4,\ w_{122} = -5,\ b_{12} = -5$$

Neuronháló kifejezőereje - példa

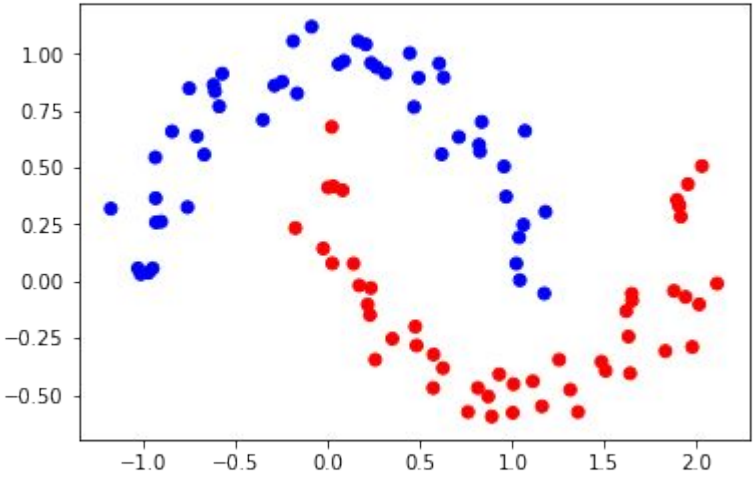
Ezeket a súlyokat kézzel állítottuk be...

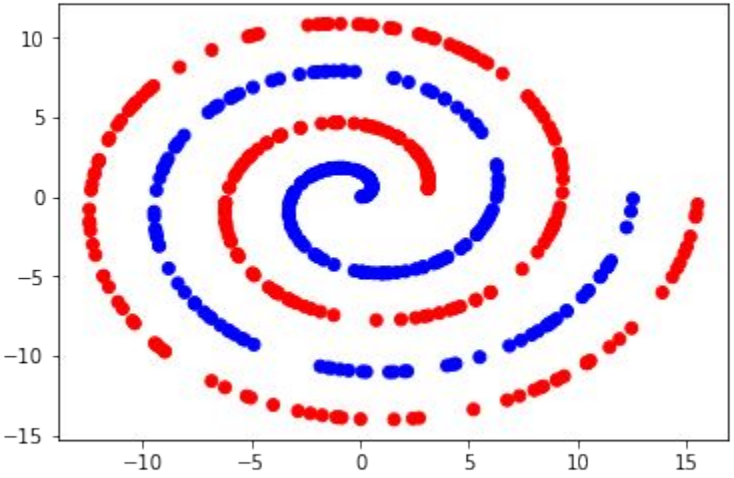
Most nézzük meg, mit **tanulhat** a neuronháló…

Colab notebook:
https://colab.research.google.com/drive/1L0gD0Zzq7dcW7QfqavdI0jXzrWW_5H1b

Neuronháló kifejezőereje - példa

Két bonyolultabb klasszifikációs probléma:





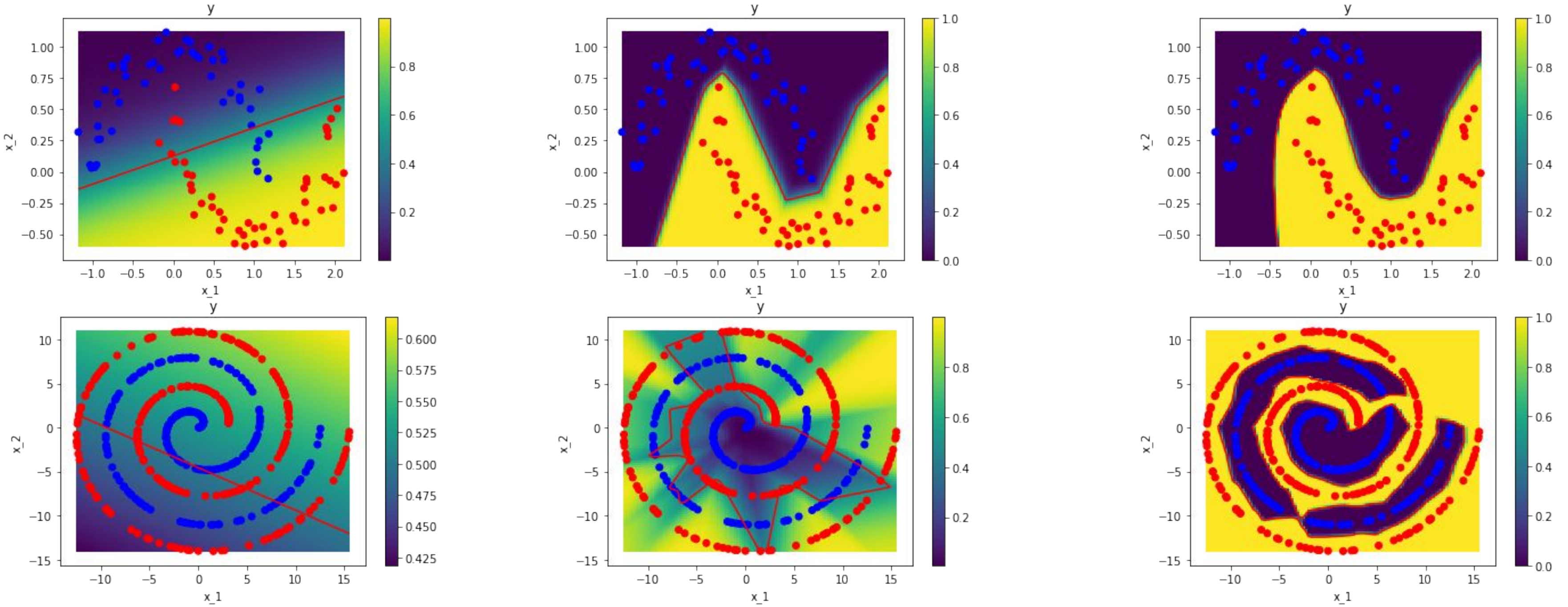
Bináris klasszifikáció: tanuljuk meg döntési felülettel elválasztani a két kategóriájú mintaelemeket!

Neuronháló kifejezőereje - példa

log.reg.

2 réteg, 20+1 neuron

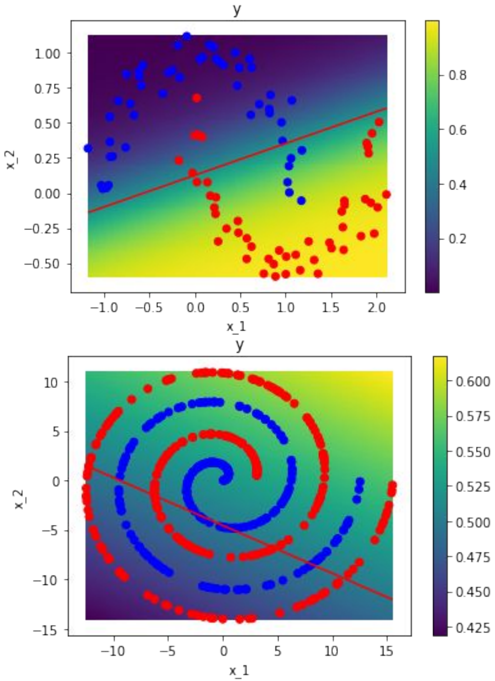
4 réteg, 20+20+20+1 neuron



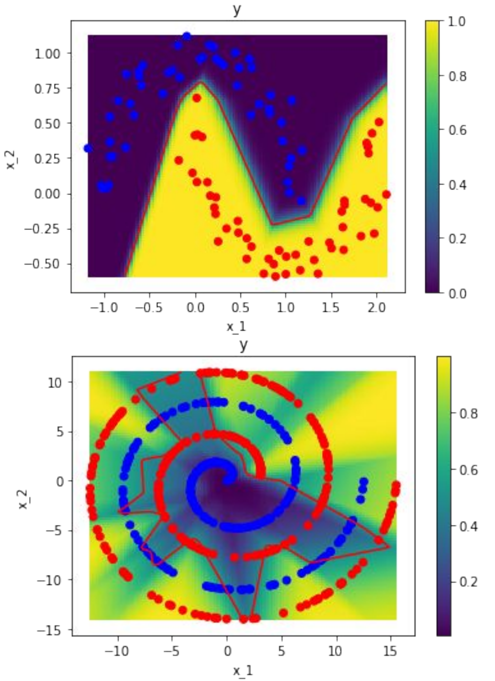
Neuronháló kifejezőereje - példa

Túltanulás (overfitting)

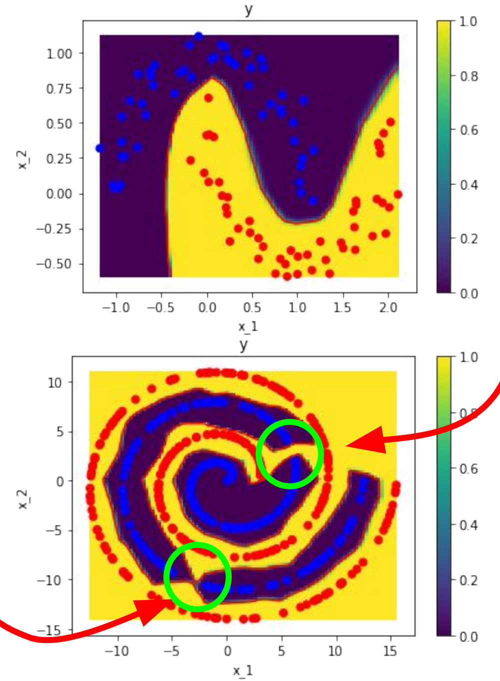
log.reg.



2 réteg, 20+1 neuron



4 réteg, 20+20+20+1 neuron

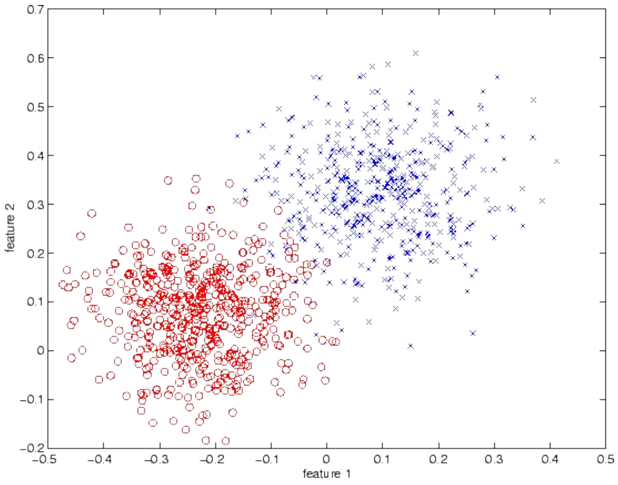


Alul- és túltanulás neuronhálók esetén

Interaktív neuronháló szimuláció: https://playground.tensorflow.org/

A szimulációs felülettel vizsgálható a túltanulás hatása különböző architektúrájú neuronhálók esetén.

Beállítások:
Data: gaussian
Noise: >25



Klasszifikáció: Döntési felületet tanulunk, mely elválasztja a két kategóriából való címkével rendelkező mintaelemeket.

Alul- és túltanulás neuronhálók esetén

Interaktív neuronháló szimuláció: https://playground.tensorflow.org/

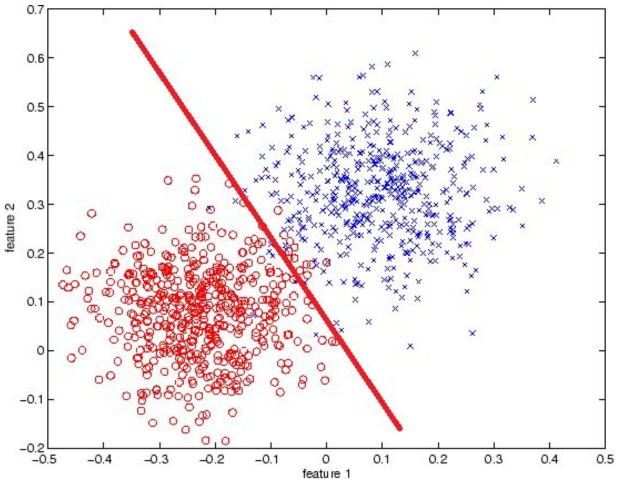
A szimulációs felülettel vizsgálható a túltanulás hatása különböző architektúrájú neuronhálók esetén.

Beállítások:

Data: gaussian

Noise: >25

Az ideális döntési felület egy egyenes lenne, azonban nagymértékű zajjal generáltuk a mintát, hogy túltanulást könnyű legyen előidézni.



Klasszifikáció: Döntési felületet tanulunk, mely elválasztja a két kategóriából való címkével rendelkező mintaelemeket.

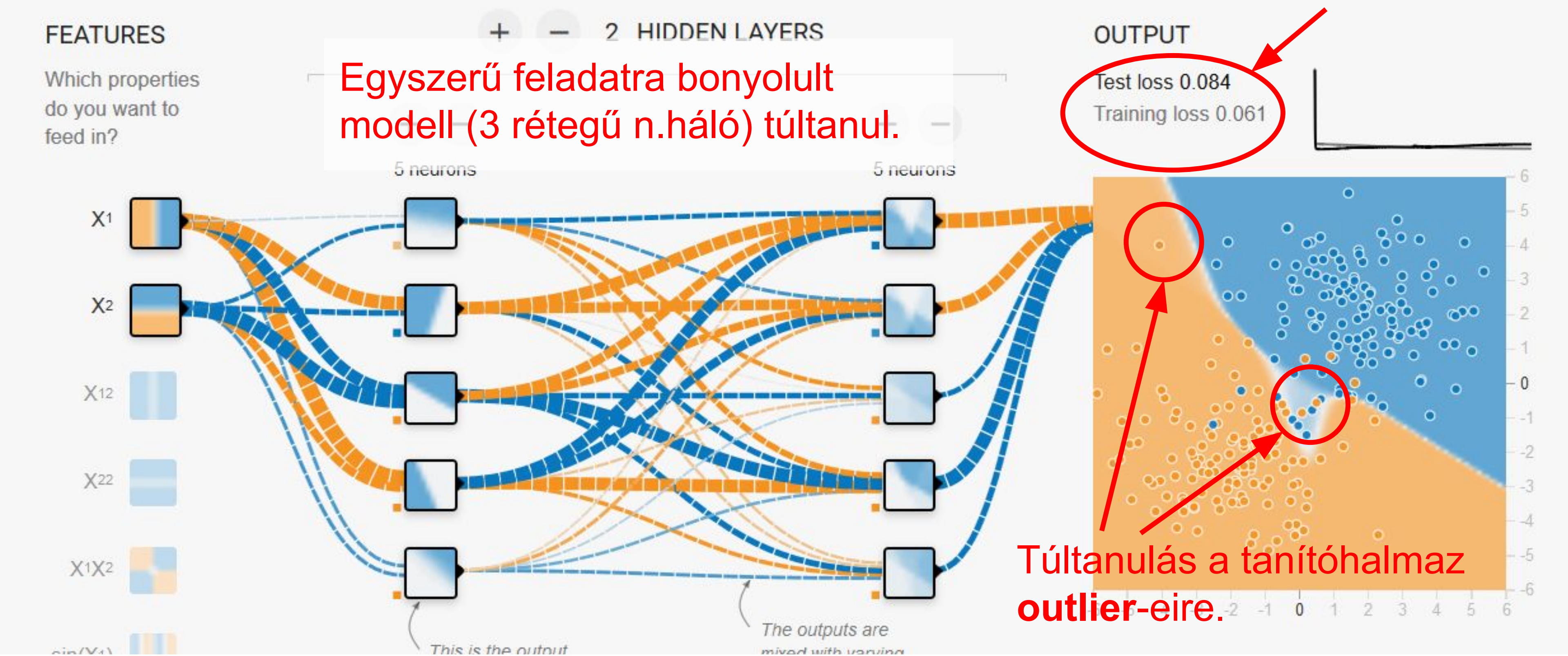
Alul- és túltanulás neuronhálók esetén

A **training loss magas**,
de a **test loss alacsony.**

Egyszerű feladatra egyszerű modell
(log.reg.) megfelelő

Alul- és túltanulás neuronhálók esetén

A **training loss alacsony**,
de a **test loss magas**.



Egyszerű feladatra bonyolult modell (3 rétegű n.háló) túltanul.

Túltanulás a tanítóhalmaz **outlier**-eire.

MLP alkalmazása kézírás felismerésére

MNIST adatbázis

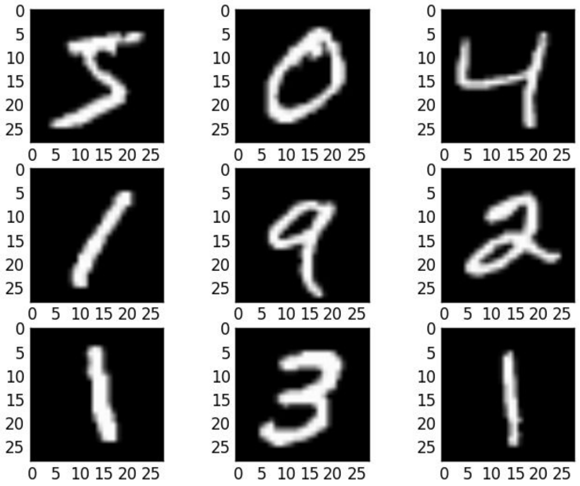
- Kézzel írt számjegyek
- 28 × 28-as méretű képek
- 10 kategória (számjegyek: 0 .. 9)
- 60 ezer tanítópélda,
  10 ezer tesztpélda

MLP alkalmazása kézírás felismerésére

MNIST adatbázis

- Kézzel írt számjegyek
- 28 × 28-as méretű képek
- 10 kategória (számjegyek: 0 .. 9)
- 60 ezer tanítópélda,
  10 ezer tesztpélda

784 input változó: Minden pixel fényereje egy változó

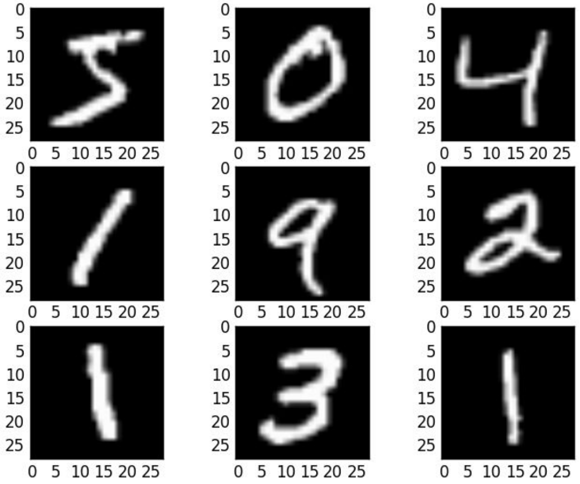


MLP alkalmazása kézírás felismerésére

MNIST adatbázis

- Kézzel írt számjegyek
- 28 × 28-as méretű képek
- 10 kategória (számjegyek: 0 .. 9)
- 60 ezer tanítópélda,
  10 ezer tesztpélda

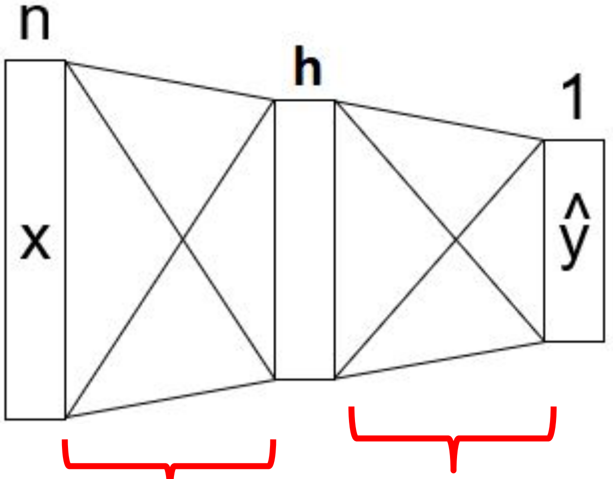
784 input változó: Minden pixel fényereje egy változó



Hogyan tanulunk 10 kategóriába klasszifikálni?

MLP - vektor alakú címke becslése

$$h(x) = g_2(W_2\, g_1(W_1 x + b_1) + b_2) = \hat{y} \approx y$$



$W_1 \in \mathbb{R}^{h \times n}$ $\quad W_2 \in \mathbb{R}^{1 \times h}$

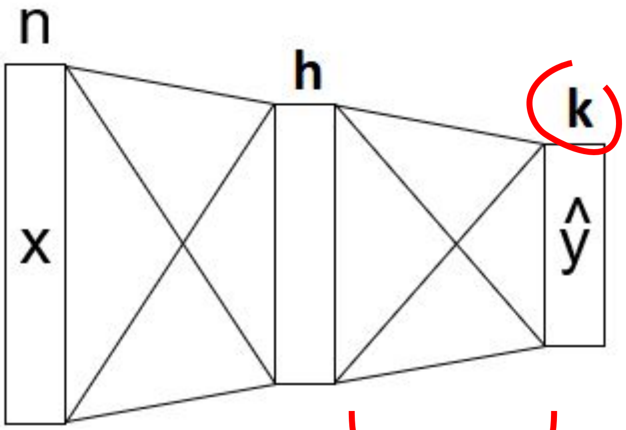
$b_1 \in \mathbb{R}^h$ $\quad b_2 \in \mathbb{R}$

Eddig y skalár.
Regressziónál egy érték becslésére,
klasszifikációnál egy valószínűség
becslésére (2 kategória) limitál minket...

$$\Theta = \{W_1, b_1, W_2, b_2\}$$

MLP - vektor alakú címke becslése

$$h(x) = g_2(W_2\ g_1(W_1 x + b_1) + b_2) = \hat{y} \approx y$$



Legyen y is vektor, hasonlóan x-hez.

$W_1 \in \mathbb{R}^{h \times n}$ $\quad W_2 \in \mathbb{R}^{k \times h}$

$b_1 \in \mathbb{R}^{h}$ $\quad b_2 \in \mathbb{R}^{k}$

$$\Theta = \{W_1, b_1, W_2, b_2\}$$

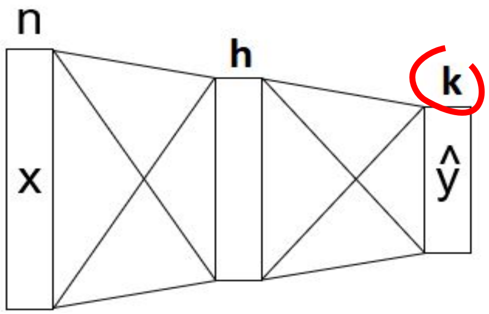
MLP - vektor alakú címke becslése

Regresszió

y^, y vektor

$$h(x) = g_2(W_2\, g_1(W_1 x + b_1) + b_2) = \hat{y} \approx y$$

Mivel regresszió esetén a címkéink tetszőleges számokat tartalmazhatnak, g_2 tipikusan identitásfüggvény (azaz elhagyható).



MLP - vektor alakú címke becslése

Regresszió

y^, y vektor

$$h(x) = g_2(W_2\, g_1(W_1 x + b_1) + b_2) = \hat{y} \approx y$$

Költség: A címkevektor elemei szerinti négyzetes költségeket átlagoljuk.

Eddig (y skalár): $$J(\Theta) = \frac{1}{2m} \sum_{j=1}^{m} (\hat{y}^{(j)} - y^{(j)})^2$$

y vektor: $$J(\Theta) = \frac{1}{2mk} \sum_{j=1}^{m} ||\hat{y}^{(j)} - y^{(j)}||_2^2 = \frac{1}{2mk} \sum_{j=1}^{m} \sum_{i=1}^{k} (\hat{y}_i^{(j)} - y_i^{(j)})^2$$

MLP - vektor alakú címke becslése

Regresszió

y^, y vektor

$$h(x) = g_2(W_2\, g_1(W_1 x + b_1) + b_2) = \hat{y} \approx y$$

Költség: A címkevektor elemei szerinti négyzetes költségeket átlagoljuk

Eddig (y skalár): $$J(\Theta) = \frac{1}{2m} \sum_{j=1}^{m} (\hat{y}^{(j)} - y^{(j)})^2$$

y vektor: $$J(\Theta) = \frac{1}{2mk} \sum_{j=1}^{m} ||\hat{y}^{(j)} - y^{(j)}||_2^2 = \boxed{\frac{1}{2mk} \sum_{j=1}^{m} \sum_{i=1}^{k} (\hat{y}_i^{(j)} - y_i^{(j)})^2}$$

MSE ahogy eddig, **de most a címkevektor elemei felett is átlagolunk.**

MLP - vektor alakú címke becslése

Klasszifikáció

y^, y vektor

$$h(x) = g_2(W_2\, g_1(W_1 x + b_1) + b_2) = \hat{y} \approx y$$

Eddig (y skalár): A szigmoid 0 és 1 közé becsült, amit valószínűségként értelmezhettünk → Legfeljebb 2 kategóriára elég

Több kategóriát hogyan?

MLP - vektor alakú címke becslése

Klasszifikáció

y^, y vektor

$$h(x) = g_2(W_2\, g_1(W_1 x + b_1) + b_2) = \hat{y} \approx y$$

Eddig (y skalár): A szigmoid 0 és 1 közé becsült, amit valószínűségként értelmezhettünk → Legfeljebb 2 kategóriára elég

Több kategóriát hogyan?

Becsüljünk minden kategóriához egy-egy valószínűséget.

A becsült valószínűségek összege 1 kellene, hogy legyen, hiszen pontosan 1 kategóriába tartozik egy mintaelem.

MLP - vektor alakú címke

Softmax függvény:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

$$\boldsymbol{\sigma} : \mathbb{R}^k \rightarrow \mathbb{R}^k$$

$$\sigma\left(\begin{bmatrix} 2.6 \\ 1.5 \\ 0.2 \\ 0.6 \end{bmatrix}\right) = \begin{bmatrix} 0.64 \\ 0.21 \\ 0.06 \\ 0.09 \end{bmatrix}$$

MLP - vektor alakú címke

Softmax függvény:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

e^{z_i} Az input vektor i-edik eleme az exponenciálisra emelve.

$\sum_{j=1}^{k} e^{z_j}$ Az exponenciálisra emelt vektorelemek összege.

$$\sigma : \mathbb{R}^k \to \mathbb{R}^k$$

$$\sigma\left(\begin{array}{|c|}\hline 2.6 \\ \hline 1.5 \\ \hline 0.2 \\ \hline 0.6 \\ \hline\end{array}\right) = \begin{array}{|c|}\hline 0.64 \\ \hline 0.21 \\ \hline 0.06 \\ \hline 0.09 \\ \hline\end{array}$$

Az eredmény vektor elemeinek összege 1, így értelmezhető valószínűségi eloszlás tömegfv.-eként

MLP - vektor alakú címke becslése

Klasszifikáció

y^, y vektor

$$h(x) = g_2(W_2\, g_1(W_1 x + b_1) + b_2) = \hat{y} \approx y$$

Eddig (y skalár): A szigmoid 0 és 1 közé becsült, amit valószínűségként értelmezhettünk → Legfeljebb 2 kategóriára elég

y vektor: g_2 a softmax függvény lesz.

MLP - vektor alakú címke becslése

Klasszifikáció

y^, y vektor

$$h(x) = g_2(W_2\ g_1(W_1 x + b_1) + b_2) = \hat{y} \approx y$$

Költség: kereszt-entrópia (cross-entropy), a bináris eset általánosítása

Eddig (y skalár): $$J(\theta) = \frac{1}{m} \sum_{j=1}^{m} [-y^{(j)} \log(\hat{y}^{(j)}) - (1 - y^{(j)})\ log(1 - \hat{y}^{(j)})]$$

y vektor: $$J(\theta) = -\frac{1}{m} \sum_{j=1}^{m} \sum_{i=1}^{k} y_i log(\hat{y}_i)$$

MLP - vektor alakú címke becslése

Klasszifikáció

y^, y vektor

$$h(x) = g_2(W_2\, g_1(W_1 x + b_1) + b_2) = \hat{y} \approx y$$

Költség: kereszt-entrópia (cross-entropy), a bináris eset általánosítása

Eddig (y skalár): $$J(\theta) = \frac{1}{m}\sum_{j=1}^{m}[-y^{(j)}\log(\hat{y}^{(j)}) - (1-y^{(j)})\, log(1-\hat{y}^{(j)})]$$

y vektor: $$J(\theta) = -\frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{k} y_i\, log(\hat{y}_i)$$

Bináris esetre (k=2) azonos a fentivel.

Hogy néz ki az (igazi) **y címke?**

MLP - vektor alakú címke becslése

Kereszt-entrópia (crossentropy):

$$J(\theta) = -\frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{k} y_i log(\hat{y}_i)$$

Igazi (true / target / ground truth) **címke, one-hot kódolással:** az a vektorelem 1, amelyik a mintaelem igazi osztályát reprezentálja, a többi 0.

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} 0.64 \\ 0.21 \\ 0.06 \\ 0.09 \end{bmatrix}$$

Becsült címke: az egyes osztályokba tartozás becsült valószínűségei

MLP - Keras, regresszió

```
model = Sequential()
model.add(Dense(h, activation='relu', input_dim=n))
model.add(Dense(1, activation='linear'))
model.compile(loss='mse',optimizer=sgd)
```

y skalár

```
model = Sequential()
model.add(Dense(h, activation='relu', input_dim=n))
model.add(Dense(k, activation='linear'))
model.compile(loss='mse',optimizer=sgd)
```

y vektor

MLP - Keras, klasszifikáció

```
model = Sequential()
model.add(Dense(h, activation='relu', input_dim=n))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer=sgd)
```

y skalár

```
model = Sequential()
model.add(Dense(h, activation='relu', input_dim=n))
model.add(Dense(k, activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer=sgd)
```

y vektor