Szoftver mély neuronhálók alkalmazásához

6. előadás

Kovács Bálint, Varga Viktor
ELTE IK Mesterséges Intelligencia Tanszék

Előző órán - Felügyelt tanulás

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

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

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

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

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

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

Előző órán - A felügyelt tanulás két fő feladata

Regresszió: Folytonos értékű címke becslése

$|Y| = \infty$ **Példa:** Autók számának, vagy életkor becslése

Klasszifikáció: Diszkrét értékű címke becslése

$|Y| < \infty$ **Példa:** Mintaelemek kategorizálása

- A lakosság számából eldönteni, hogy város-e, vagy falu egy adott település
- Mi a foglalkozása a képeken szereplő személyeknek?

Előző órán - Bináris klasszifikáció

A legegyszerűbb eset: Bináris klasszifikáció (két kategória)

$$\hat{y} = h(x) = P(x\ egy\ macska)$$

$$\hat{y} = h(x) = 1 - P(x\ egy\ kutya)$$

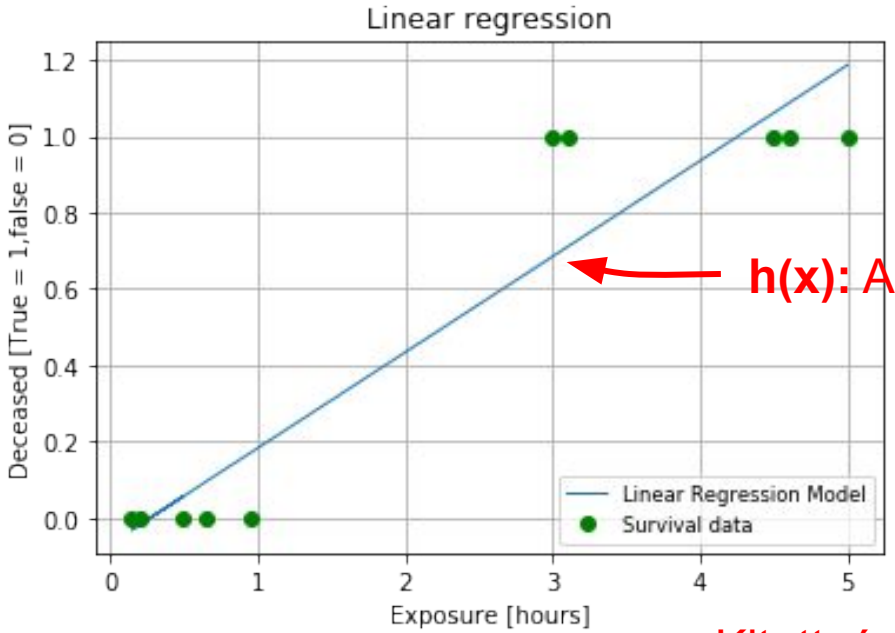
A becslés folytonos érték. Legyen **P = 0.5** a határ.

Ha a becslésünk (y^) nagyobb, mint **0.5**, azt mondjuk *“Ez egy macska”*.
Ha kisebb, azt mondjuk *“Ez egy kutya”*.

Előző órán - Klasszifikáció lin. reg.-gel - ellenpélda

Példa: Radioaktív sugárzásnak tettünk ki embereket adott ideig.
Túlélik-e?

y: Meghalt-e?
(1: igen, 0: nem)

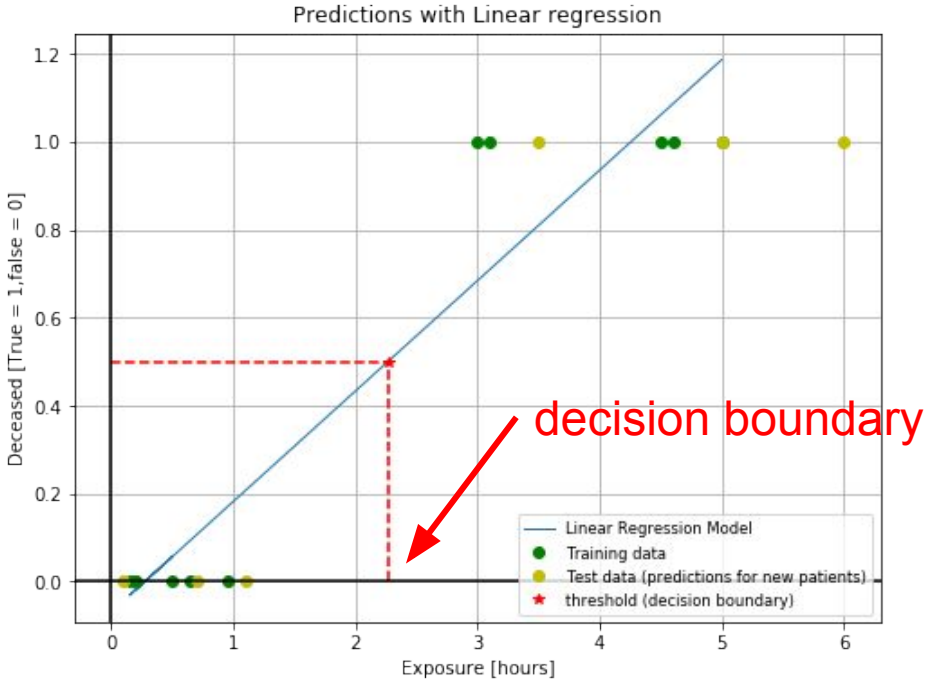


h(x): Az illesztett hipotézisfv.

x: Kitettség időtartama (óra)

Előző órán - Klasszifikáció lin. reg.-gel - ellenpélda

Döntési felület (decision boundary): $\{x \mid \hat{y} = h(x) = 0.5\}$

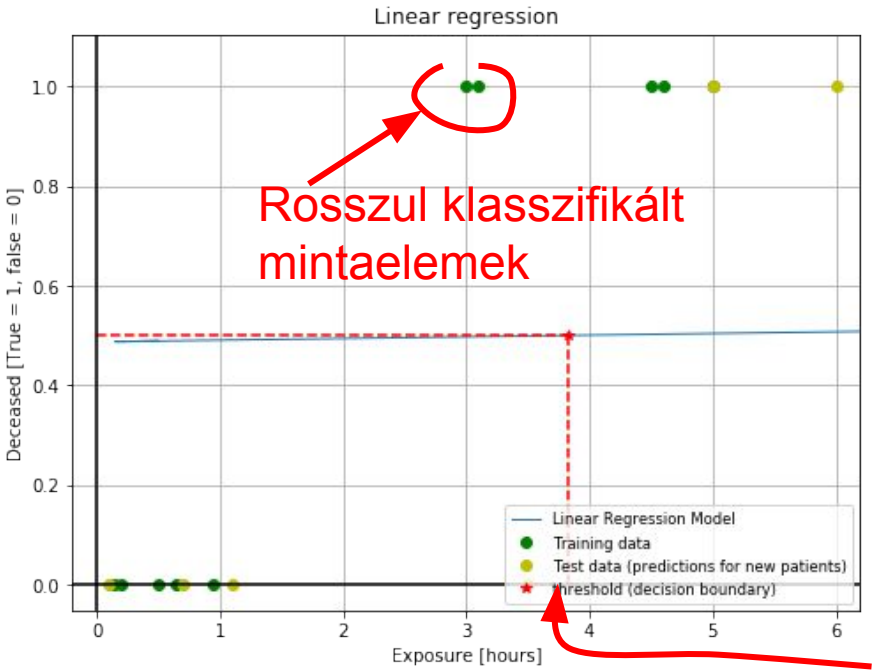


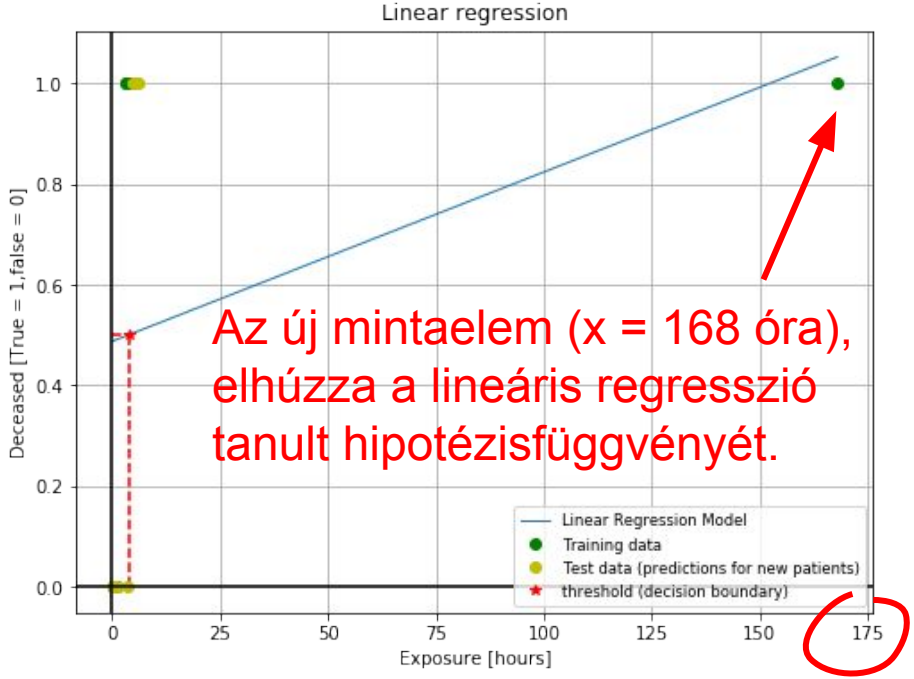
… **azon x-ek halmaza, ahol a hipotézisfv. becslése pontosan 0.5** (feltéve, hogy h folytonos)
→ tipikusan egy (hiper)felület

- **egy változó**: pont
- **két változó**: egyenes vagy görbe a síkon

…

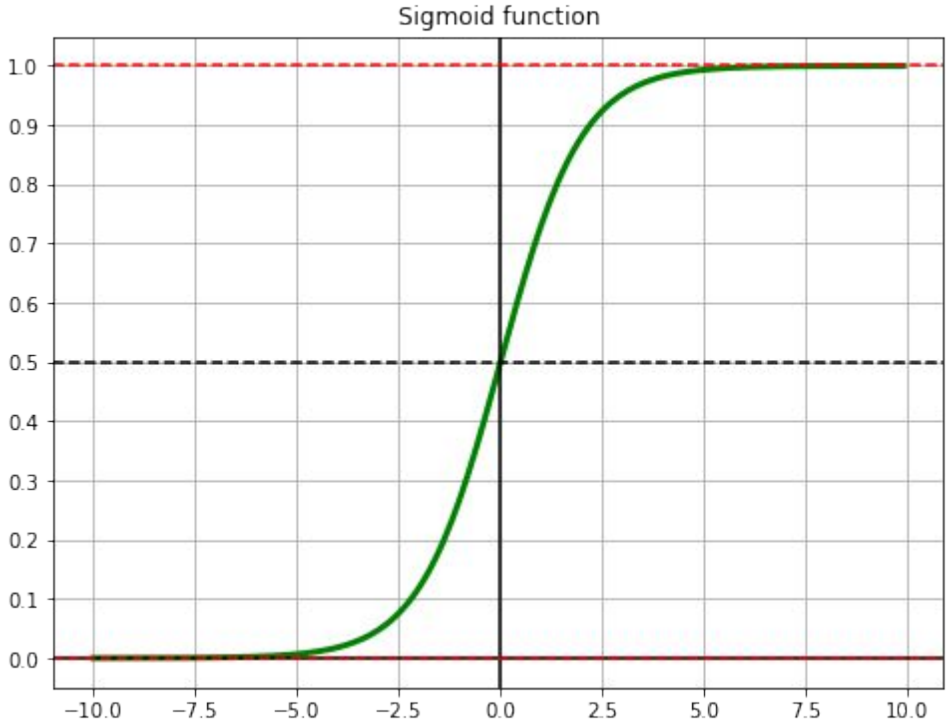
Előző órán - Klasszifikáció lin. reg.-gel - ellenpélda





Az új mintaelem jelentősen befolyásolta a döntési felületet.
Így az már 3 óra 48 percre esik.

Előző órán - Szigmoid függvény



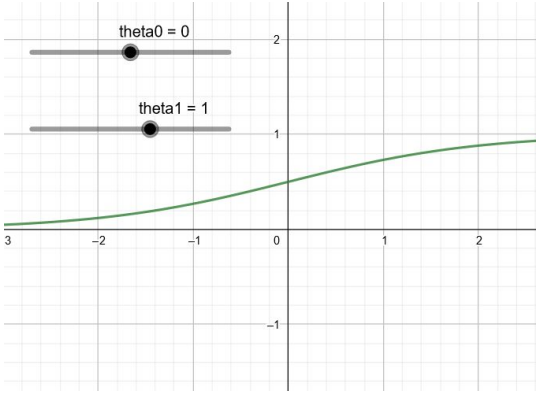
$$g(z) = \frac{1}{1+e^{-z}}$$

0 és 1 közé képez: **ideális valószínűségek becslésére!**

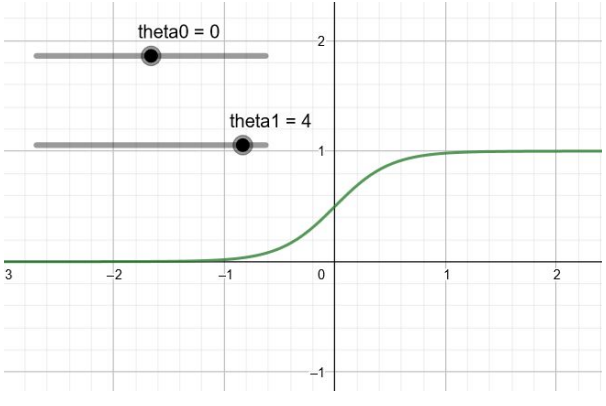
Előző órán - Klasszifikáció hipotézis függvény

$$h_\theta(x) = g(\theta_0 + \theta_1 x) = \frac{1}{1+e^{-(\theta_0+\theta_1 x)}} = \hat{y}$$

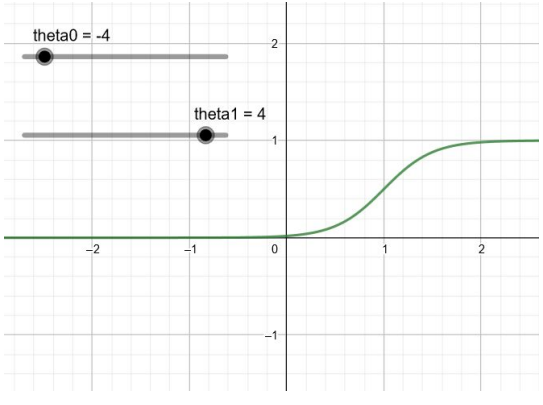
$\theta_0 = 0$
$\theta_1 = 1$



$\theta_0 = 0$
$\theta_1 = 4$



$\theta_0 = -4$
$\theta_1 = 4$



Előző órán - Logisztikus regresszió költség

Próbáljuk ki a lineáris regresszióhoz használt MSE költséget!

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

MSE loss
(átlagos négyzetes eltérés)

ahol, $h_\theta(x) = g(x\theta) = \frac{1}{1+e^{-x\theta}} = \hat{y}$

Probléma: A szigmoid (g) nem konvex, a négyzete sem lesz az…

→ Több lokális szélsőérték lehet

→ A gradiens módszer nem feltétlenül találja meg az optimális megoldást

Előző órán - Logisztikus regresszió költség

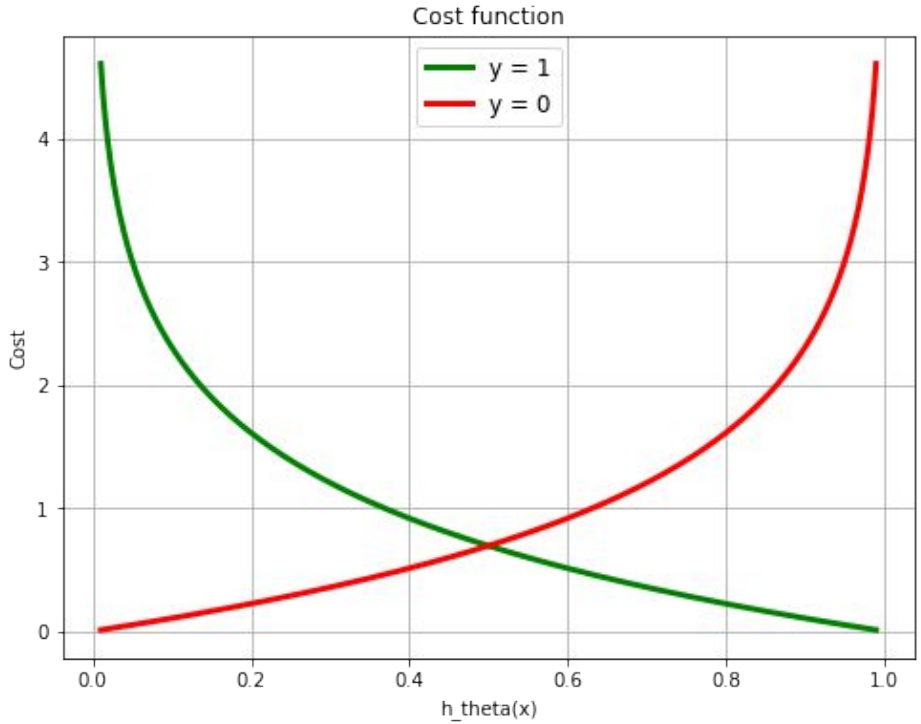
Költségfüggvény logisztikus regresszióhoz:

$$J(\theta) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Levezethető:

a költségfüggvény konvex

(a második deriváltja mindenhol pozitív)



Előző órán - logisztikus regresszió költség

Költségfüggvény, összevont alak (logistic loss, binary crossentropy)**:**

$$J(\theta) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$$J(\theta) = \frac{1}{m} \sum_{j=1}^{m} [-y^{(j)} \; log(\underbrace{h_\theta(x^{(j)})}_{\hat{y}^{(j)}}) - (1 - y^{(j)}) \; log(1 - \underbrace{h_\theta(x^{(j)})}_{\hat{y}^{(j)}})]$$

= 0, ha a true y címke 0

= 0, ha a true y címke 1

Előző órán - Gradiens módszer (log.reg.)

Gradiens módszer általánosan (többváltozós):

repeat until convergence {

  for i $\leftarrow 1 \ldots n$ {

    $grad_i = \frac{\partial}{\partial \theta_i} J(\theta)$

  }

  for i $\leftarrow 1 \ldots n$ {

    $\theta_i = \theta_i - \alpha \, grad_i$

  }

}

$$\frac{\partial}{\partial \theta_i} J(\theta) = \frac{1}{m} \sum_{j=1}^{m} (h_\theta(x^{(j)}) - y^{(j)}) x_i^{(j)}$$

A derivált ránézésre ugyanaz, de ne feledjük: itt a hipotézisfüggvény más

$$h_\theta(x) = \frac{1}{1+e^{-(\theta_0 x_0 + \theta_1 x_1 + \ldots)}}$$

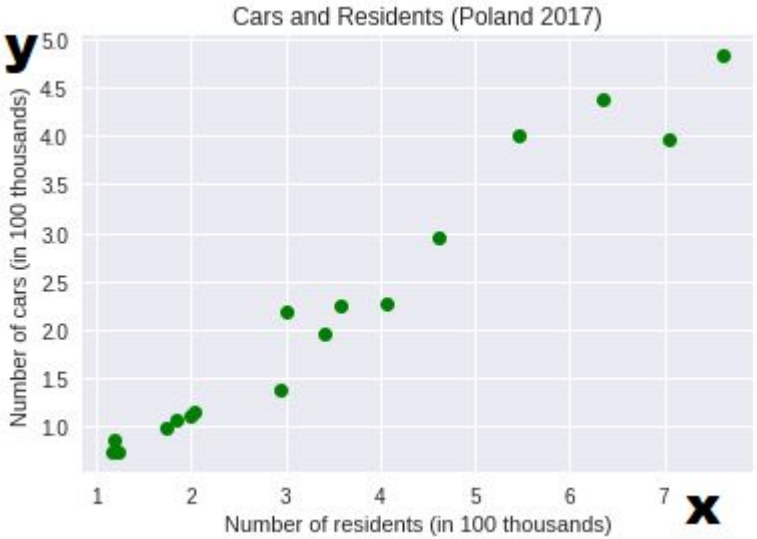
Előző órán - logisztikus regresszió

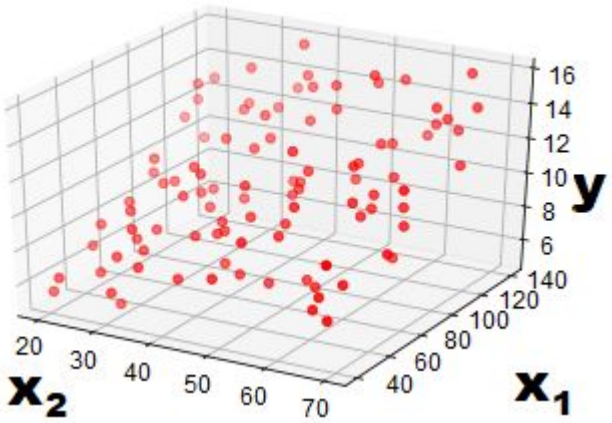
Tehát,

bár **klasszifikáció esetén a címke diszkrét**, logisztikus regresszióval oldjuk meg, ami a mintaelemek adott kategóriákba tartozásának valószínűségeit becsli (melyek **folytonos értékek**).

Vissza a regresszióhoz...

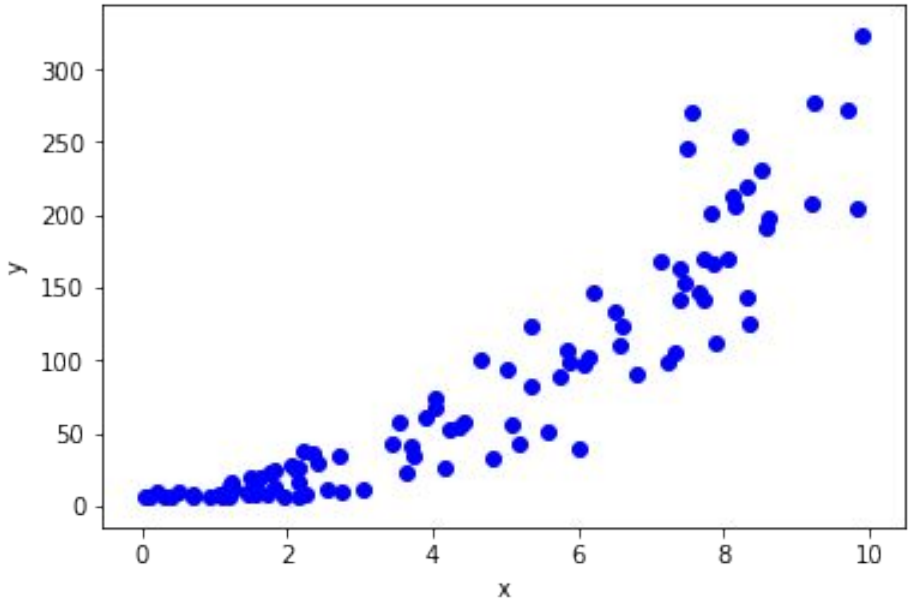
A lineáris regresszió jól működött olyan mintán, ahol a változók lineáris kombinációja jól közelíti a címkét.





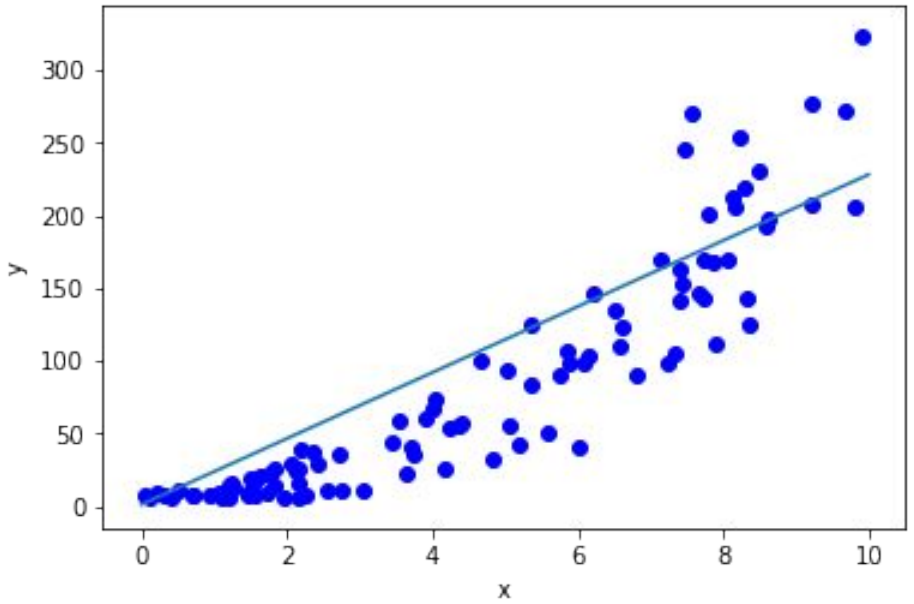
Vissza a regresszióhoz...

Azonban, mit tehetünk akkor, ha ez nem áll fenn?

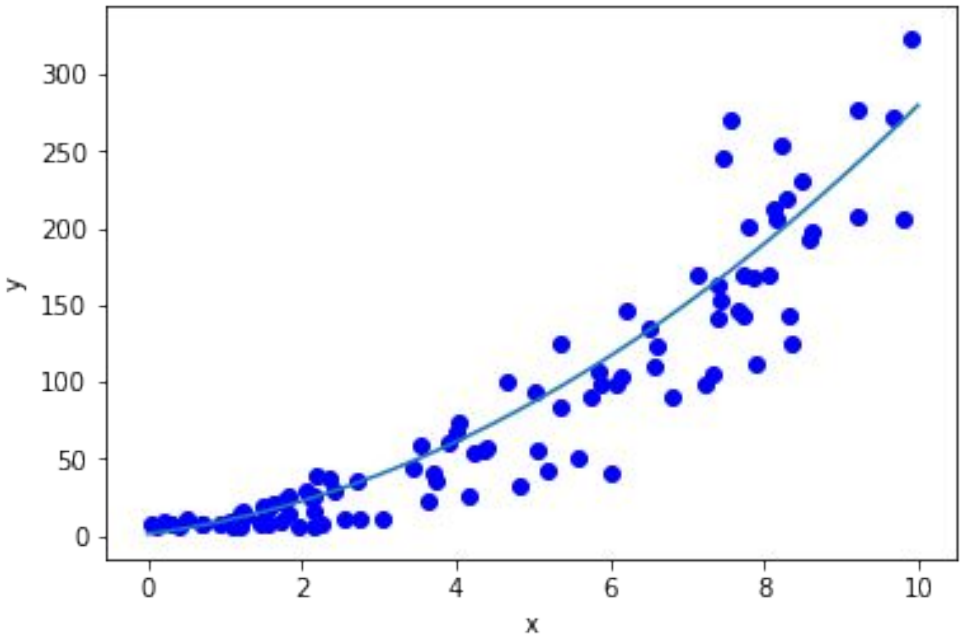


Vissza a regresszióhoz...

Pl. $y \approx ax^2 + bx + c$ áll fenn. Ebben az esetben a lineáris regresszió nem ad túl jó eredményt. **Mit tehetünk?**



Polinomiális regresszió



Polinomiális regresszió

Hipotézisfüggvény:

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

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

Költésgfüggvény (MSE marad)**:**

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

Polinomiális regresszió

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

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

Megoldás gradiens módszerrel:

A költségfüggvény deriválható.

→ A gradiens módszer alkalmazható jó **θ** együtthatók megtalálására.

Polinomiális regresszió

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

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

Megoldás gradiens módszerrel:

A költségfüggvény deriválható.

→ A gradiens módszer alkalmazható jó **θ** együtthatók megtalálására.

Nem a **θ** paramétereket hatványozzuk.

→ J továbbra is kvadratikus a **θ** paraméterek szerint!

Polinomiális regresszió

Megoldás a gyakorlatban:

- A hatványok miatt a magasabbrendű együtthatókhoz tartozó gradiensek extrém nagyok/kicsik.
- Ahogy többváltozós lin. reg.-nél is láttuk, erre a gradiens módszer érzékeny.

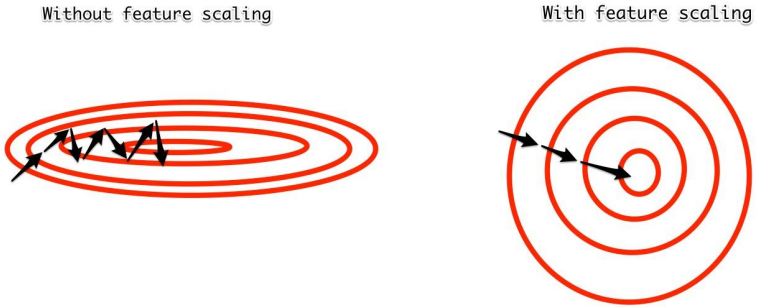
Megoldás?

Polinomiális regresszió

Megoldás a gyakorlatban:

- A hatványok miatt a magasabbrendű együtthatókhoz tartozó gradiensek extrém nagyok/kicsik.
- Ahogy többváltozós lin. reg.-nél is láttuk, erre a gradiens módszer érzékeny.

Megoldás?



Polinomiális regresszió

Megoldás:

Kezeljük a polinomiális regressziót úgy, mintha lineáris regresszió lenne:

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

helyett

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

Polinomiális regresszió

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

helyett

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

… helyettesítünk és előre kiszámoljuk a változók hatványait.

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

Végül az új változókat azonos nagyságrendre hozzuk feature scaling-gel és a feladatot **lineáris regresszióként** oldjuk meg.

Polinomiális regresszió - kitérő

Egyváltozós eset: Vandermonde mátrix

$$x = \begin{bmatrix} 1 & x^{(1)} & (x^{(1)})^2 & \dots & (x^{(1)})^n \\ \dots & \dots & \dots & \dots & \\ 1 & x^{(m)} & (x^{(m)})^2 & \dots & (x^{(m)})^n \end{bmatrix} \in \mathbb{R}^{m \times n}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \dots \\ \theta_n \end{bmatrix} \in \mathbb{R}^n \qquad y = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix} \in \mathbb{R}^m$$

Feature scaling:
x, x^2, …, x^n -ekre külön változóként tekintünk és egymástól függetlenül skálázzuk őket

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

Polinomiális regresszió - kitérő

Többváltozós eset, például:

$$x = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & (x_1^{(1)})^2 & (x_1^{(1)})^3 (x_2^{(1)})^2 & \dots \\ \dots & \dots & \dots & \dots & & \\ 1 & x_1^{(m)} & x_2^{(m)} & (x_1^{(m)})^2 & (x_1^{(m)})^3 (x_2^{(m)})^2 & \dots \end{bmatrix} \in \mathbb{R}^{m \times n}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \dots \\ \theta_n \end{bmatrix} \in \mathbb{R}^n \qquad y = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix} \in \mathbb{R}^m$$

Feature scaling:
x_1, x_2, (x_1)^2, …
-ekre külön változóként tekintünk és egymástól függetlenül skálázzuk őket

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

A minta felosztása

Modell életciklusa:

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

A két halmaz diszjunkt!

A minta felosztása

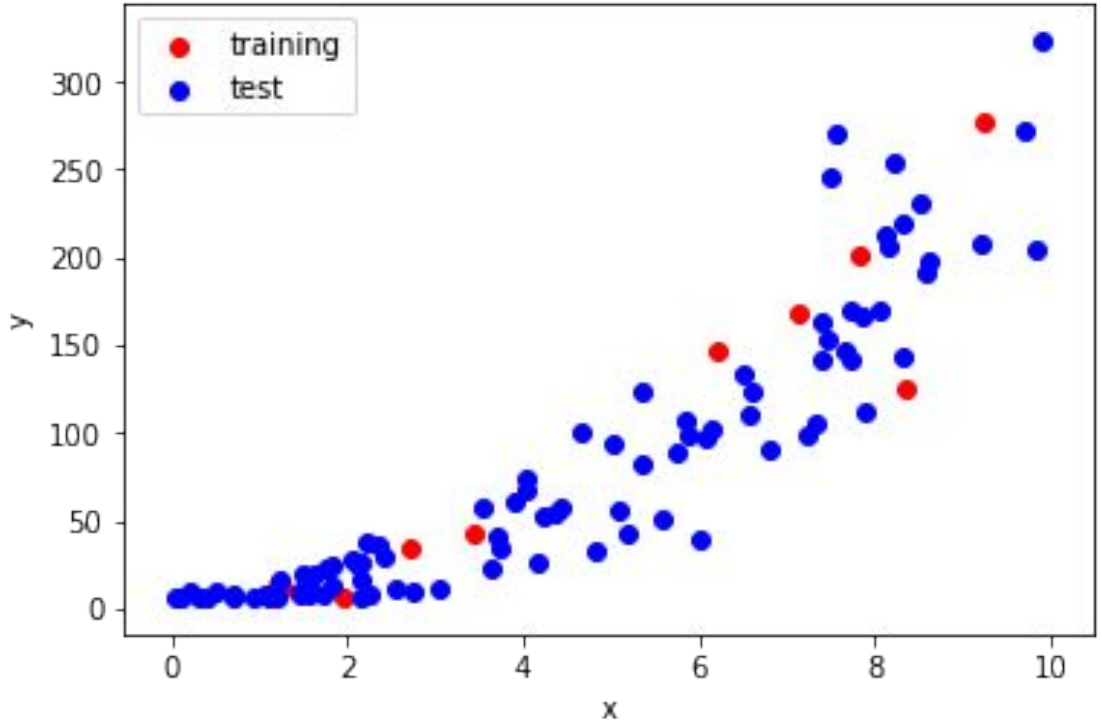
Modell életciklusa:

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

A két halmaz diszjunkt!

A teszthalmazt ne használjuk tanulásra! A teszthalmaz segítségével megpróbáljuk megbecsülni, hogyan fog teljesíteni a betanított modellünk tanításkor nem látott mintaelemeken.

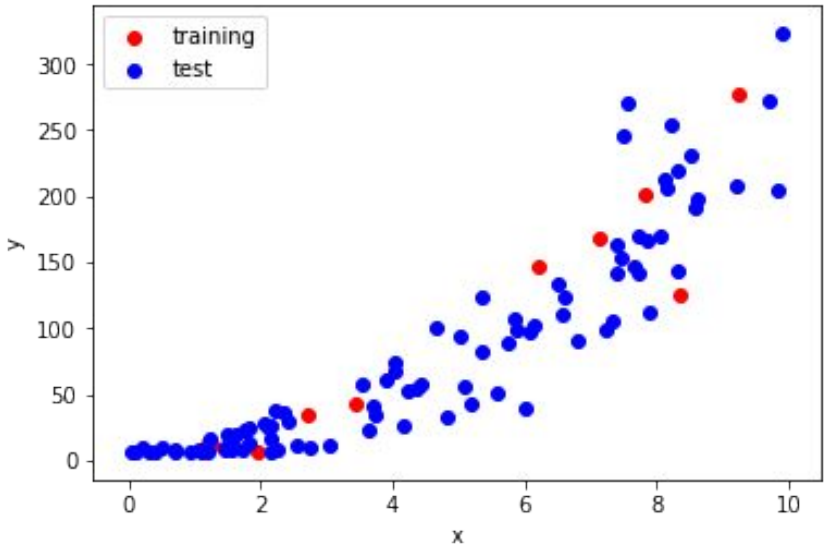
A minta felosztása



A költség alakulása a betanítás során

Végezzünk polinomiális regressziót az alábbi mintán!

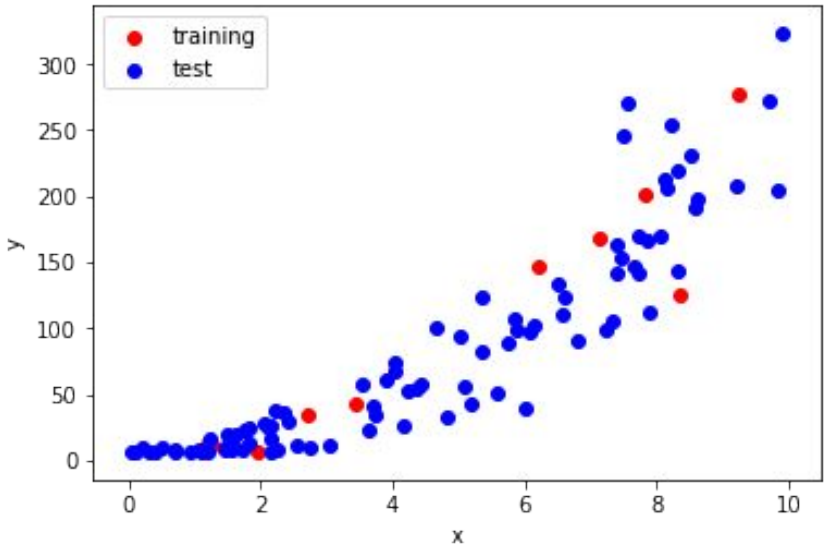
Hányadfokú polinomot illesszünk?



A költség alakulása a betanítás során

Végezzünk polinomiális regressziót az alábbi mintán!

Hányadfokú polinomot illesszünk?



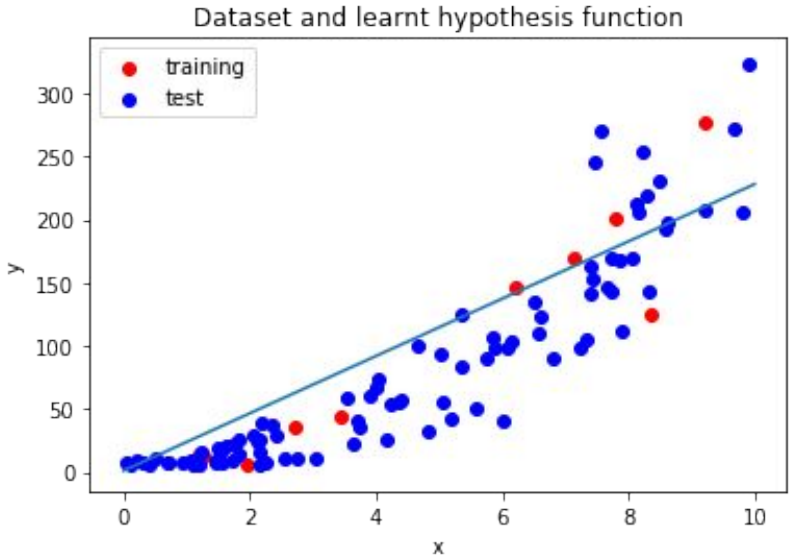
Egyetlen változó esetén jól látszik az ábrán, hogy egy másodfokú polinom illeszkedne jól a mintára.

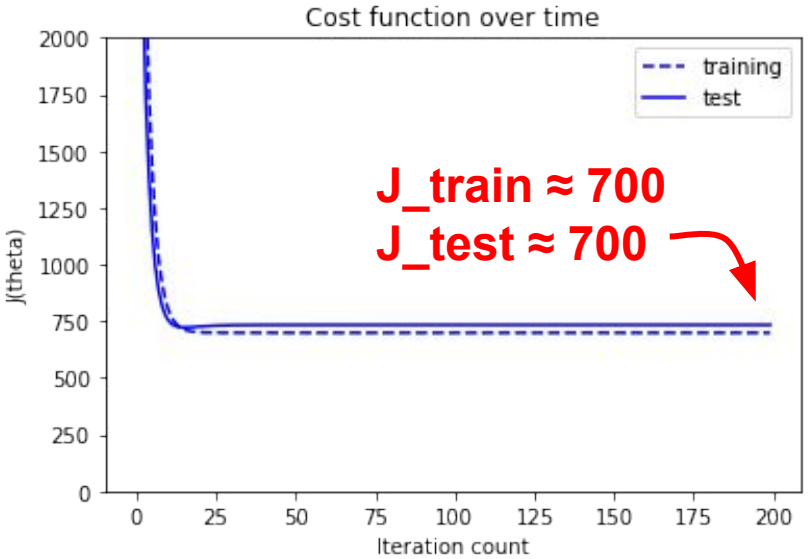
Sajnos sok változó esetén az ábra is sokdimenziós lenne, nem tudnánk értelmezni, így kísérletezés nélkül nem tudjuk megmondani hányadfokú poinom lenne az ideális.

A költség alakulása a betanítás során: **alultanulás**

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

A training és teszt hiba is (hasonlóan) nagy.



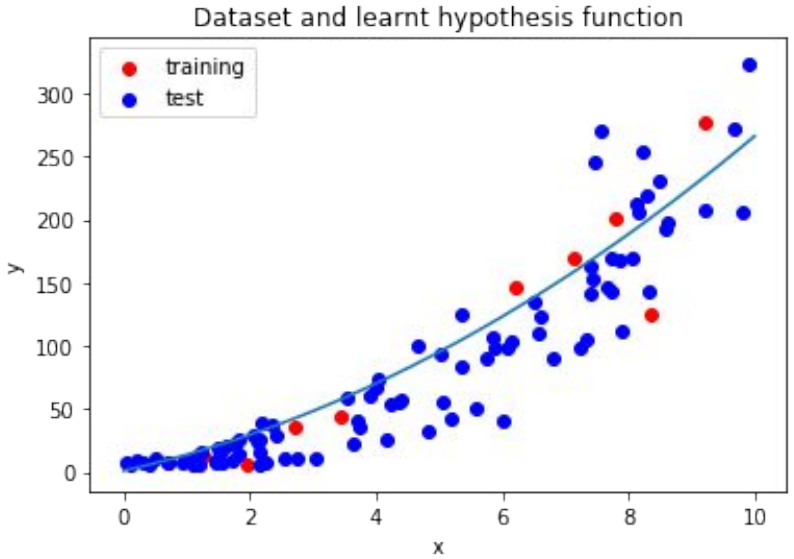


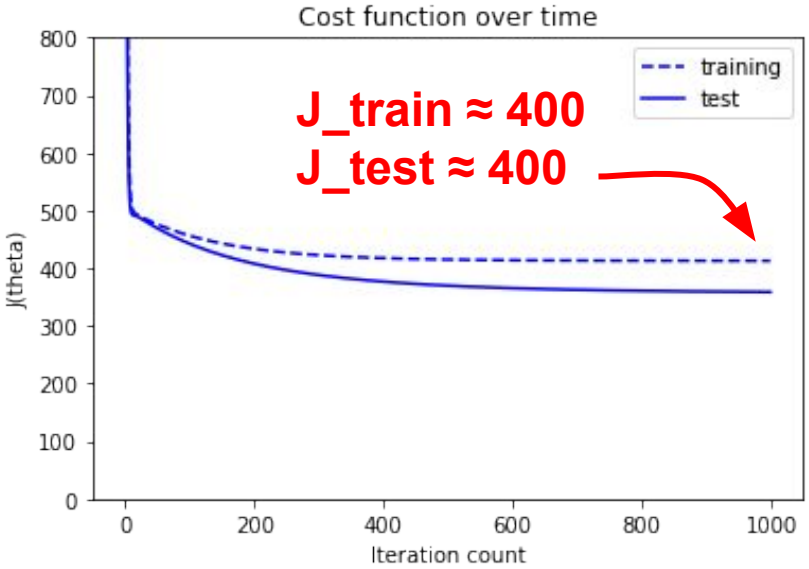
“high bias, low variance”

A költség alakulása a betanítás során: **“éppen jó”**

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

A training és teszt hiba is kicsi és mindkettő sokáig csökken.



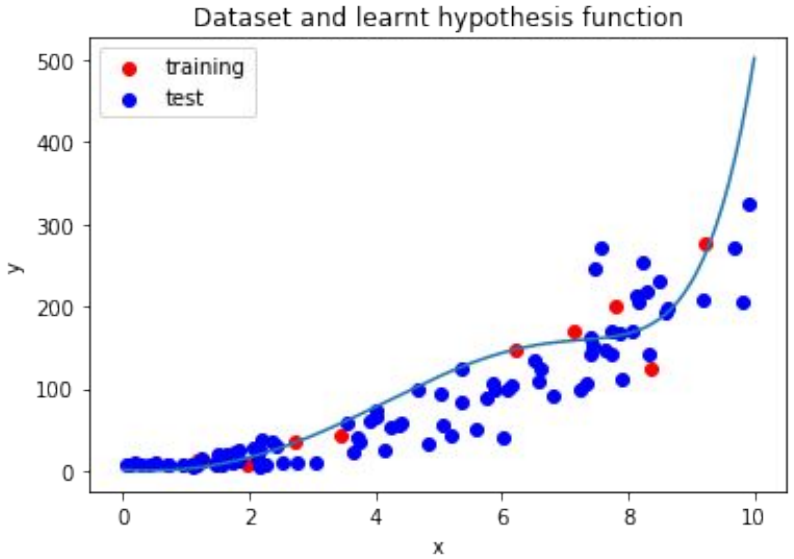


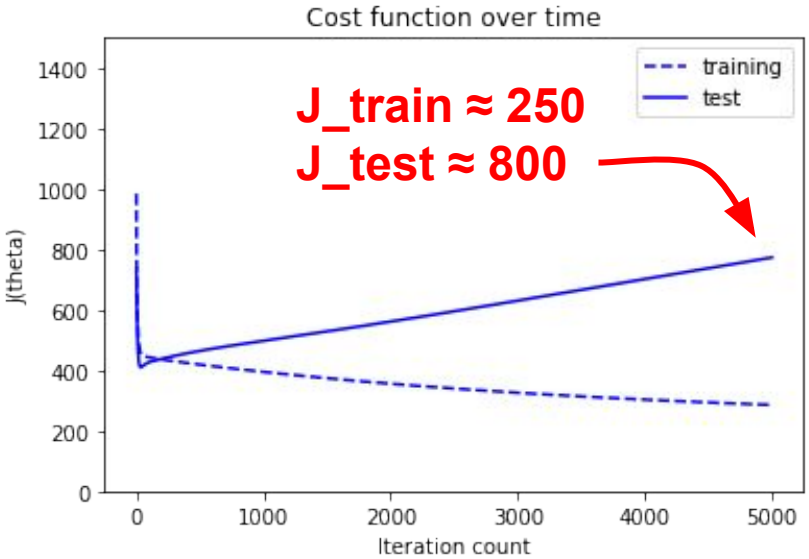
“balancing bias & variance”

A költség alakulása a betanítás során: **túltanulás**

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

A training hiba csökken, de a teszt hiba nő.

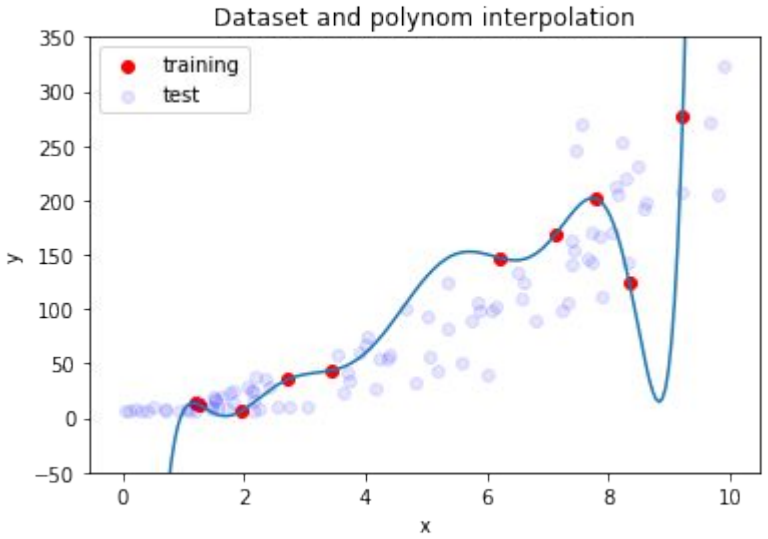




"low bias, high variance"

Kitérő: polinom interpoláció

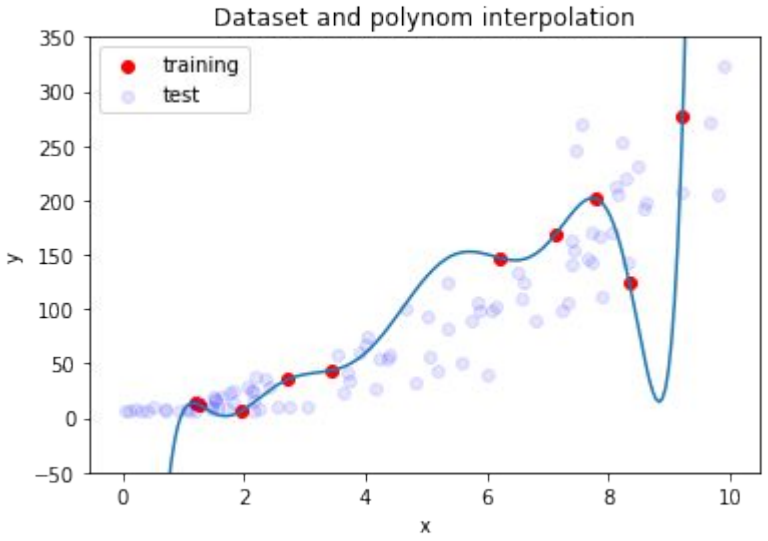
Polinom interpolációval akár pontosan is illeszthetünk polinomot a tanítóhalmazra.



J_train = 0
J_test ≈ 252507.67

Kitérő: polinom interpoláció

Polinom interpolációval akár pontosan is illeszthetünk polinomot a tanítóhalmazra. Gépi tanulásban ez azonban nem hasznos!



J_train = 0
J_test ≈ 252507.67

10 pontra kilencedfokú polinom hiba nélkül illeszthető.
Ha elég kevés a tanítóadat és elég nagy a hipotézisfüggvény kifejezőereje, pontos illeszkedés is elérhető, de a tesztadattól ekkor nagyon eltávolodunk.
→ **Extrém túltanulás**

Túltanulás mély neuronhálókban

Példa: Fényképek kategorizálása

Ha a mélyháló kifejezőereje elég nagy, illetve elég sok tanulható paraméterrel rendelkezik, elképzelhető, hogy nem kívánt módon tanulja meg a feladatot.

Kép forrása: sastanin @ Stack Exchange

Túltanulás mély neuronhálókban

Példa: Fényképek kategorizálása

Ha a mélyháló kifejezőereje elég nagy, illetve elég sok tanulható paraméterrel rendelkezik, elképzelhető, hogy nem kívánt módon tanulja meg a feladatot.

Például, **“megjegyez” minden egyes tanítóhalmazbeli képet valamilyen speciális mintázatról.** ...akár egy egyedi JPEG tömörítési hibáról (artifact-ról)!

Kép forrása: sastanin @ Stack Exchange

Alultanulás és túltanulás elkerülése

A célunk, hogy a modellünk ne tanuljon alul és ne tanuljon túl. Ehelyett, találjuk meg az “éppen jó” modellt!

Hiszen, akár alul-, akár túltanulás esetén a betanult modell gyengén fog teljesíteni, amikor a betanítás során nem látott, címkézetlen mintaelemekhez fogunk címkéket becsülni.

Hogyan kerülhetjük el az alul-, illetve a túltanulás előfordulását?

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.

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.

Példa: A lineáris regresszió alkalmazható bonyolult feladatokra is, például fényképekről celebek életkorának becslésére. Azonban a modell **súlyosan alultanul**, hiszen a lineáris regresszió mindössze a pixelek fényerejének lineáris kombinációjából tudja az életkorbecslést előállítani, ami nem egy jó megközelítés az életkor becslésére. Egy bonyolultabb modell, például egy konvolúciós neuronháló alkalmasabb lehet a feladatra.

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.

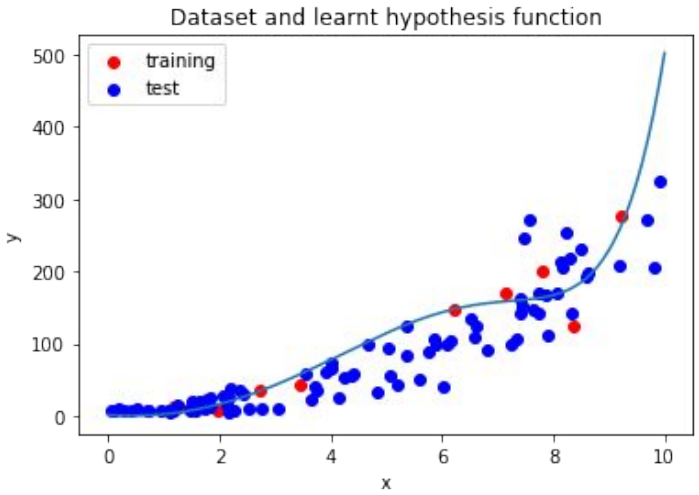
Kezelése:

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

Túltanulás (overfitting) kezelése

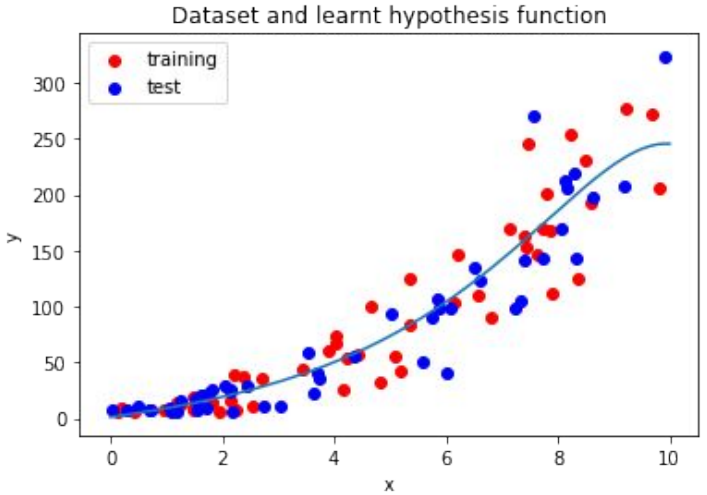
Több tanítóadat

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



$|X_{train}| = 10$

J_train ≈ 250
J_test ≈ 800



$|X_{train}| = 50$

J_train ≈ 400
J_test ≈ 400

Túltanulás (overfitting) kezelése

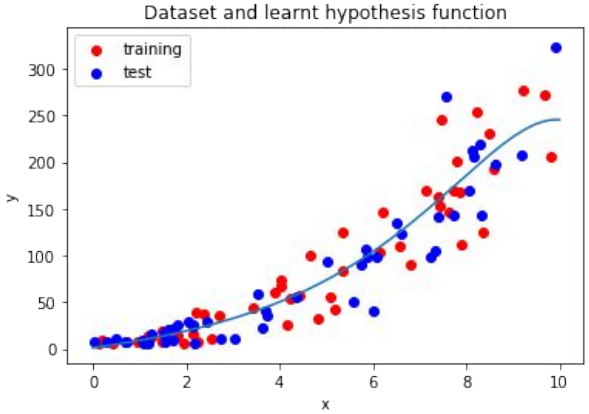
Több tanítóadat

Probléma: Tipikusan nem áll rendelkezésre elegendő címkézett tanítópélda, hogy egy százmillió paraméteres mély neuronháló túltanulását megakadályozzuk.

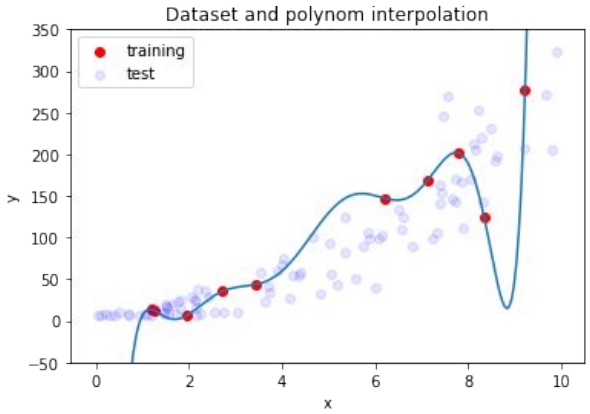
Más módszerekre is szükség van.

Túltanulás (overfitting) kezelése

(L2) regularizáció: Magasabb fokú polinomok illesztésénél az együtthatók (a θ paraméterek) jellemzően megnőnek, ahogy egyre pontosabb az illeszkedés.



θ = [1., 7.46, 0.82, 0.06, 0.005, 0.0004, 0., 0., 0., 0., 0.]



θ = [[-2229., 7234., -9545., 6756., -2843., 743., -121., 12.05, -0.66, 0.015]

Túltanulás (overfitting) kezelése

(L2) regularizáció: Magasabb fokú polinomok illesztésénél az együtthatók (a θ paraméterek) jellemzően megnőnek, ahogy egyre pontosabb az illeszkedés.

A nagyméretű együtthatók (paraméterek) büntetése segíthet!

Hogyan?

Túltanulás (overfitting) kezelése

(L2) regularizáció: Magasabb fokú polinomok illesztésénél az együtthatók (a θ paraméterek) jellemzően megnőnek, ahogy egyre pontosabb az illeszkedés.

Tegyük bele a költségfüggvénybe a büntetést:

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

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

Túltanulás (overfitting) kezelése

(L2) regularizáció: Magasabb fokú polinomok illesztésénél az együtthatók (a θ paraméterek) jellemzően megnőnek, ahogy egyre pontosabb az illeszkedés.

Tegyük bele a költségfüggvénybe a büntetést:

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

MSE költség: A becslés hibájának büntetése

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

(L2) regularizációs tag: A nagyméretű paraméterek büntetése

λ = 0: Regularizáció nélkül, eredeti modell

Túl nagy λ: Érdemesebb 0 paramétereket tanulni, mint jó becslést adni...

Túltanulás (overfitting) kezelése

(L2) regularizáció: Magasabb fokú polinomok illesztésénél az együtthatók (a θ paraméterek) jellemzően megnőnek, ahogy egyre pontosabb az illeszkedés.

Tegyük bele a költségfüggvénybe a büntetést:

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

MSE költség: A becslés hibájának büntetése

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

A konstans tagot (θ_0) ne büntessük!

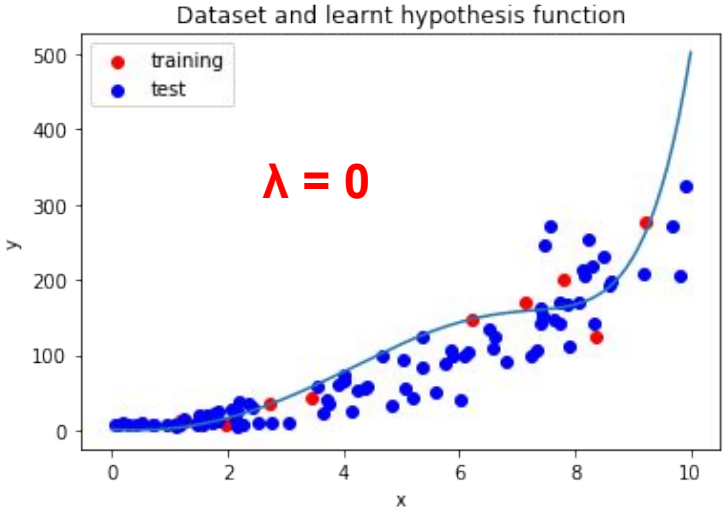
Túltanulás (overfitting) kezelése

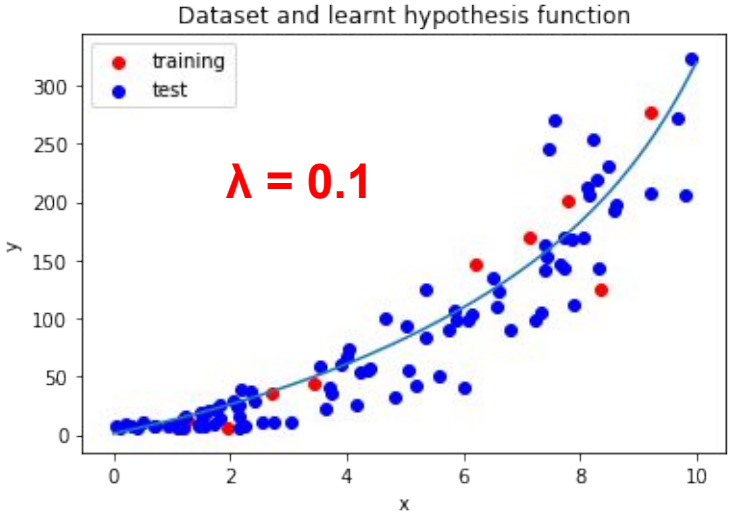
(L2) regularizáció:

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

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

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





Túltanulás (overfitting) kezelése

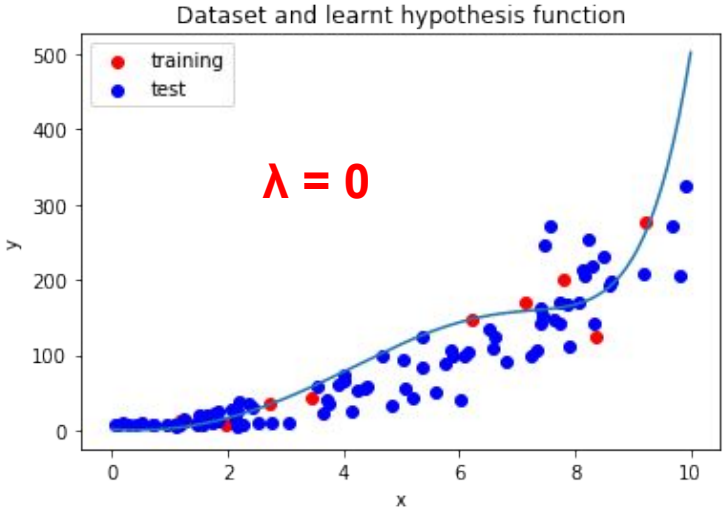
(L2) regularizáció:

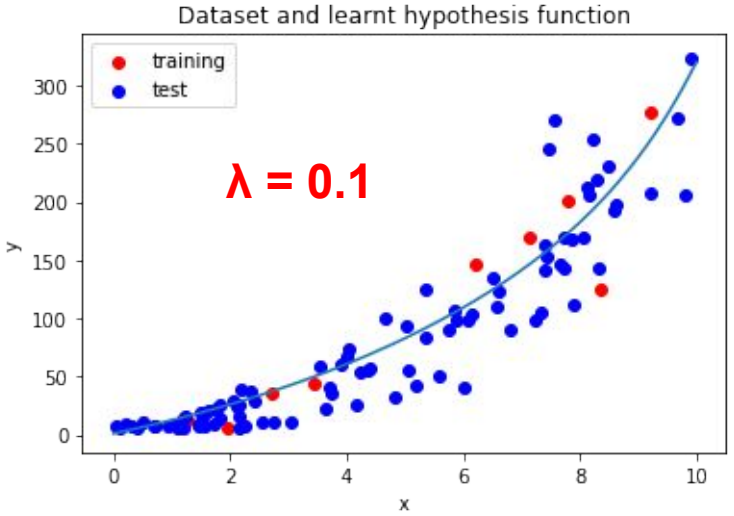
$$h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_9 x^9$$

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

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

Kevés tanítóadat esetén is!





Túltanulás (overfitting) kezelése

(L2) regularizáció:

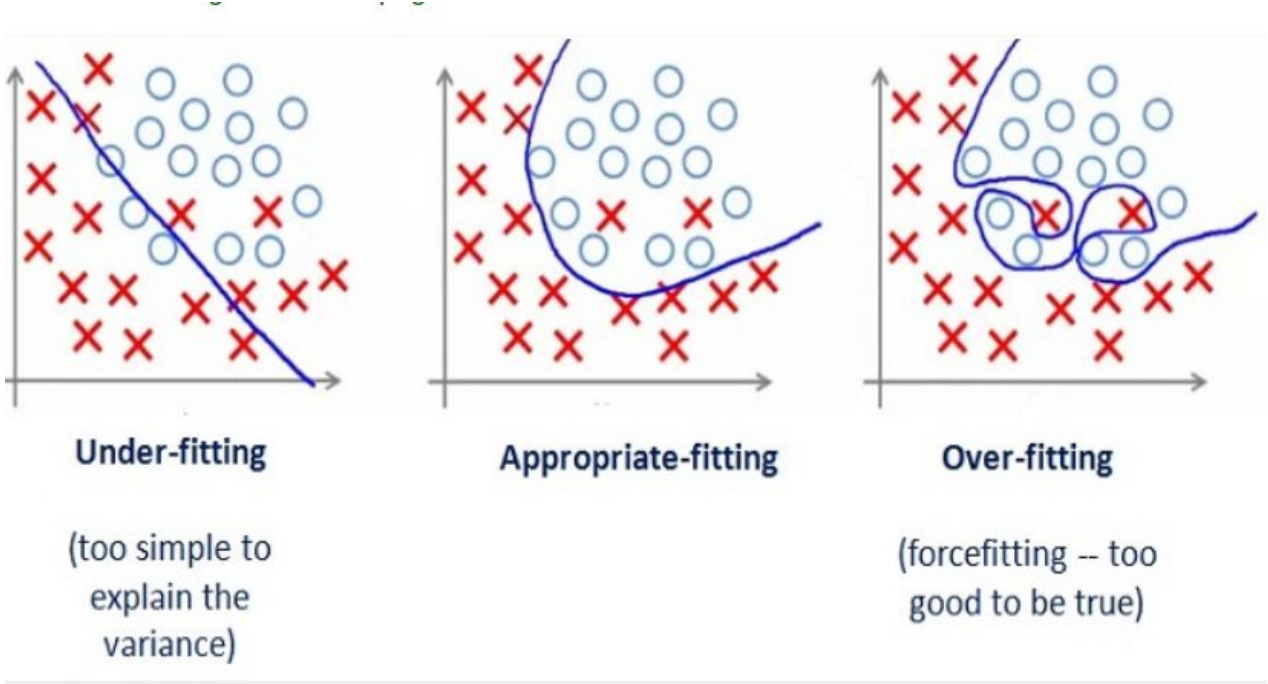
$$h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_9 x^9$$

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

Az L2 regularizáció nem csak polinomiális regressziónál működik.

A tapasztalat azt mutatja, hogy általában, neuronhálók esetén is megnőnek a súlyok (paraméterek) túltanulás esetén.

Alul- és túltanulás klasszifikáció esetén



Kép forrása: Andrew Ng Machine learning video series

Alul- és túltanulás klasszifikáció esetén

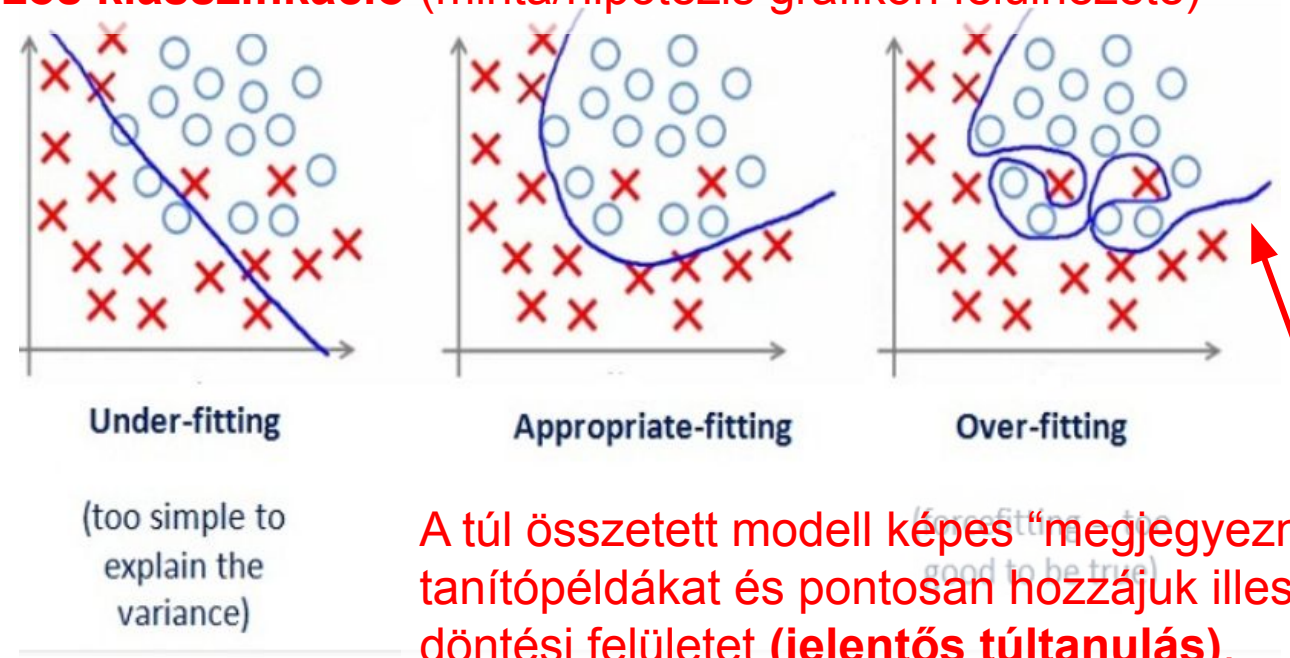
2 változós klasszifikáció (minta/hipotézis grafikon felülnézete)

A tanult döntési felület a három esetben:
Túl egyszerű, megfelelő kifejezőerejű és túl összetett modellek esetén.

Kép forrása: Andrew Ng Machine learning video series

Alul- és túltanulás klasszifikáció esetén

2 változós klasszifikáció (minta/hipotézis grafikon felülnézete)

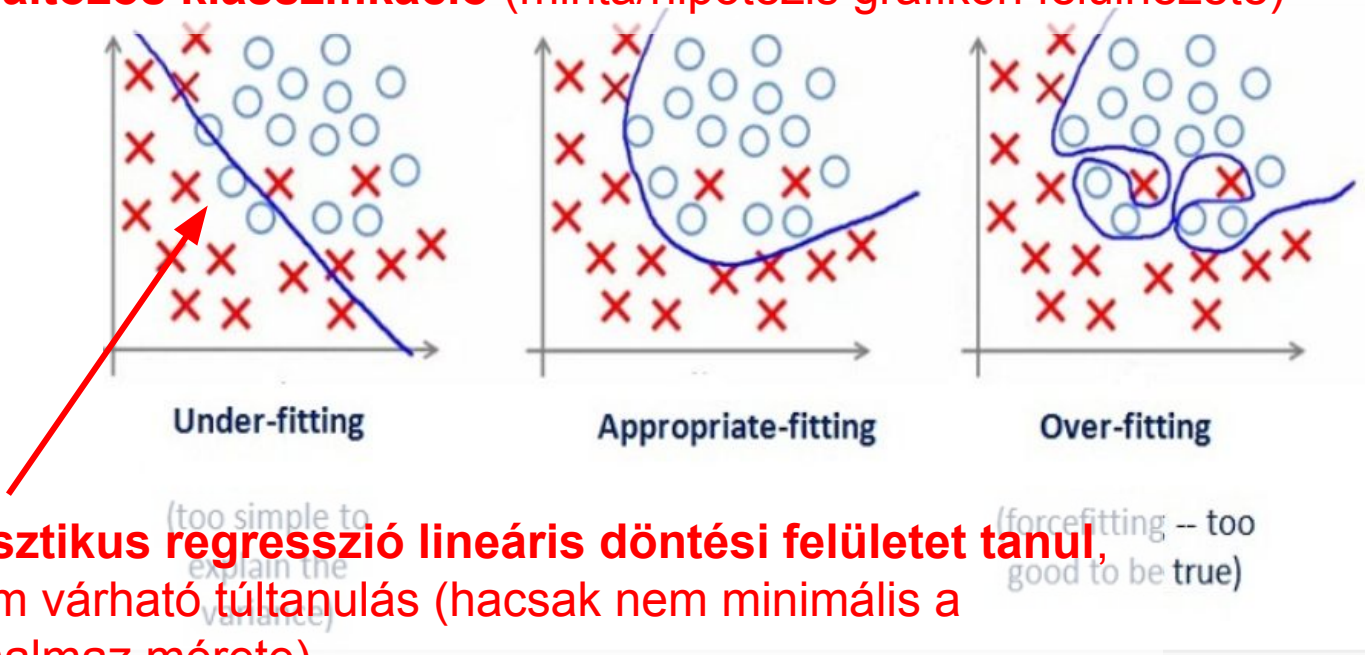


A túl összetett modell képes "megjegyezni" az egyes tanítópéldákat és pontosan hozzájuk illeszteni a döntési felületet **(jelentős túltanulás)**.

Kép forrása: Andrew Ng Machine learning video series

Alul- és túltanulás klasszifikáció esetén

2 változós klasszifikáció (minta/hipotézis grafikon felülnézete)

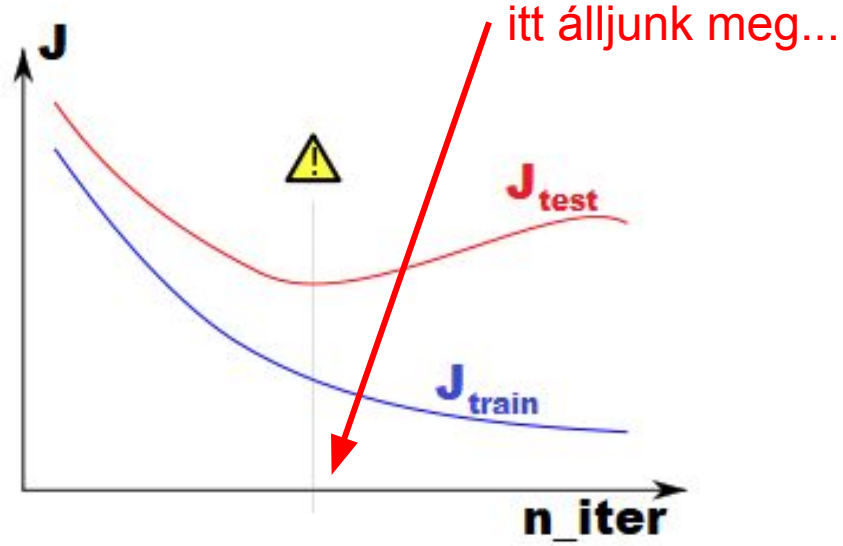


A logisztikus regresszió lineáris döntési felületet tanul, így nem várható túltanulás (hacsak nem minimális a tanítóhalmaz mérete).

Kép forrása: Andrew Ng Machine learning video series

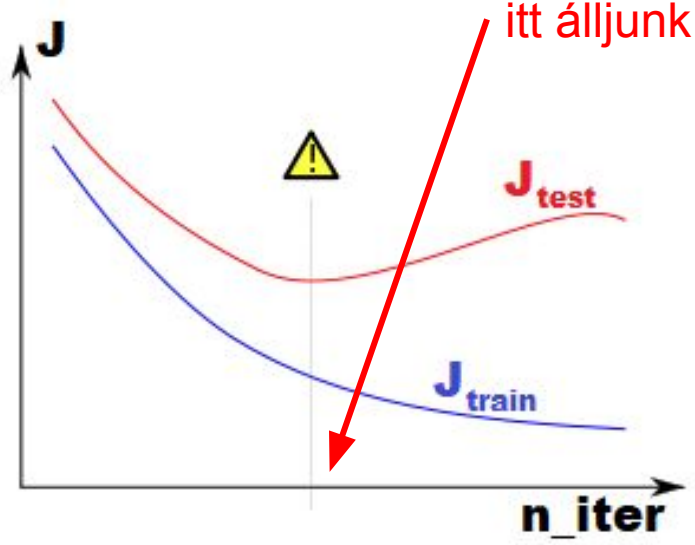
Túltanulás (overfitting) kezelése

Early stopping: A túltanulás sokszor elkerülhetetlenül megjelenik mély neuronhálók betanításakor, adott számú iteráció után. Viszonylag hatékony megoldás az *early stopping:*



Túltanulás (overfitting) kezelése

Early stopping: A túltanulás sokszor elkerülhetetlenül megjelenik mély neuronhálók betanításakor, adott számú iteráció után. Viszonylag hatékony megoldás az *early stopping:*



Álljunk meg a gradiens módszer iterációval, **amikor a hiba a teszthalmazon elkezd nőni!**

Túltanulás (overfitting) kezelése

Early stopping:

break loop when J_test stopped decreasing

repeat until convergence {

  for i ← 1 . . . n {

    $grad_i = \frac{\partial}{\partial \theta_i} J(\theta)$

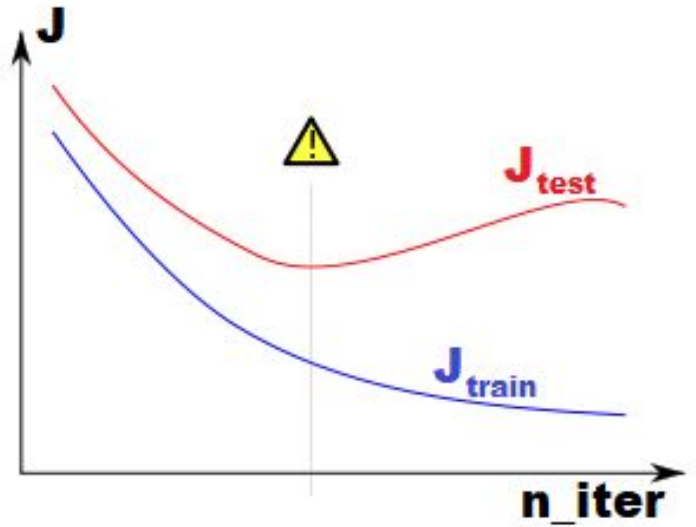
  }

  for i ← 1 . . . n {

    $\theta_i = \theta_i - \alpha \, grad_i$

  }

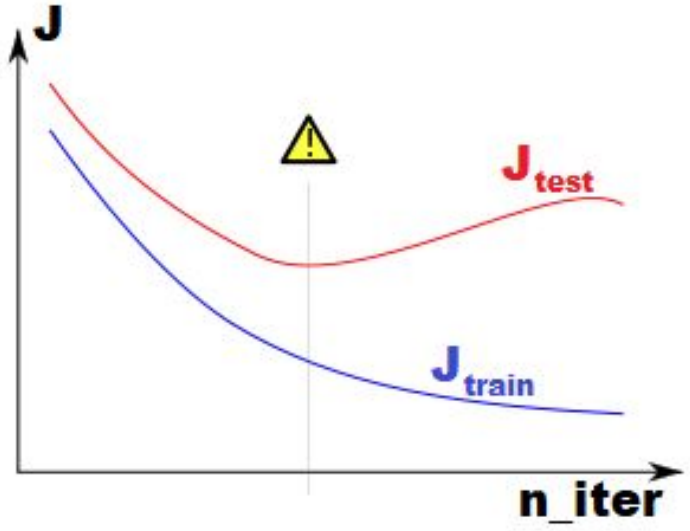
}



Early stopping

Kikötöttük, hogy a teszthalmaz elemeire nem tanítunk.

Mi történik mégis?

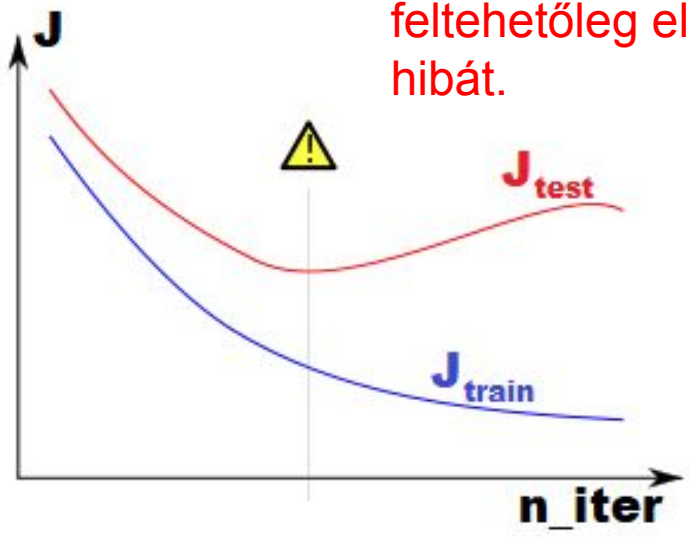


Early stopping

Kikötöttük, hogy a teszthalmaz elemeire nem tanítunk.

Mi történik mégis?

A tanított modellt hozzáigazítjuk a teszthalmazhoz, hiszen a tanítással akkor állunk le, ha a teszthalmazon feltehetőleg elértük a lehető legkisebb hibát.



A minta felosztása

Új felosztás:

Tanítóhalmaz, **validációs halmaz,** teszthalmaz

Például 50%, 25%, 25%

A minta felosztása

Új felosztás:

Tanítóhalmaz, **validációs halmaz,** teszthalmaz

Például 50%, 25%, 25%

Kritikus alkalamzásokban például 20%, 10%, 70%.
A teszthalmaztól elvárjuk, hogy a lehető legjobban megbecsülhessük a segítségével a betanított modell jövőbeli teljesítményét még nem látott adaton.

Validációs halmaz, hiperparaméterek

A validációs halmazt fogjuk használni a következő paraméterek optimalizálására:

- Tanulási ráta (alfa)
- Polinom fokszáma, vagy a neuronháló architektúrája (rétegek, neuronok száma, stb.)
- Gradiens módszer iterációs lépéseinek száma (early stopping)
- …

Az ilyen paramétereket **hiperparamétereknek** nevezzük.

A validációs halmazt a hiperparaméterek optimalizálására használjuk.

Hiperparaméterek

A legkisebb hibájú modell megtalálása így már két optimalizációs feladatból áll:

Eddig: $\theta^* = argmin_\theta \, J(\theta)$

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

ψ*: optimális hiperparaméterek

ψ: hiperparaméterek

A betanítás folyamata

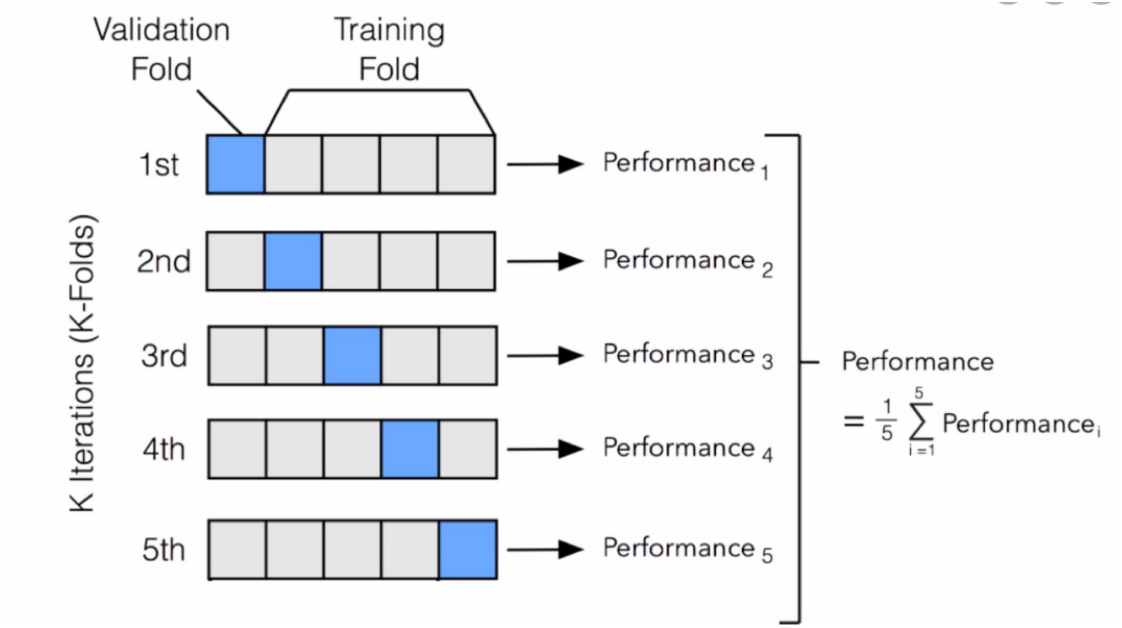
Feladat: $\psi^*, \theta^* = argmin_\psi \; argmin_\theta \; J_\psi(\theta)$

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**

A validációs halmaz választása

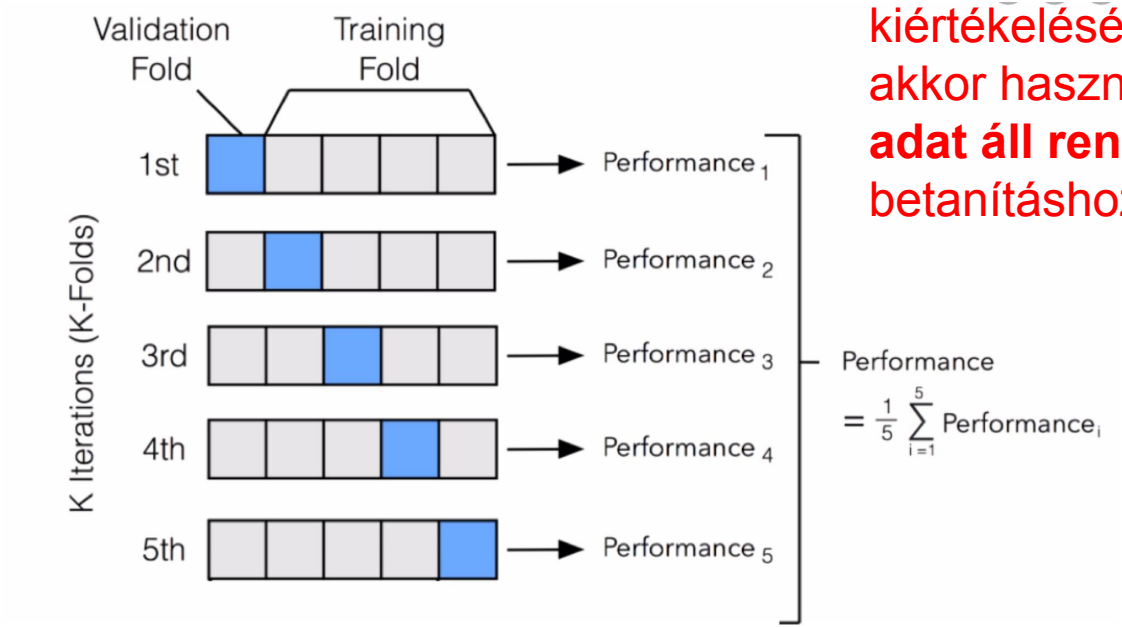
Keresztvalidáció (cross validation):



Kép forrása: https://medium.com/analytics-vidhya/

A validációs halmaz választása

Keresztvalidáció (cross validation):



Egy népszerű technika hiperparaméterek kiértékeléséhez. Leginkább akkor hasznos, **ha kevés adat áll rendelkezésünkre** a betanításhoz / validációhoz.

Kép forrása: https://medium.com/analytics-vidhya/

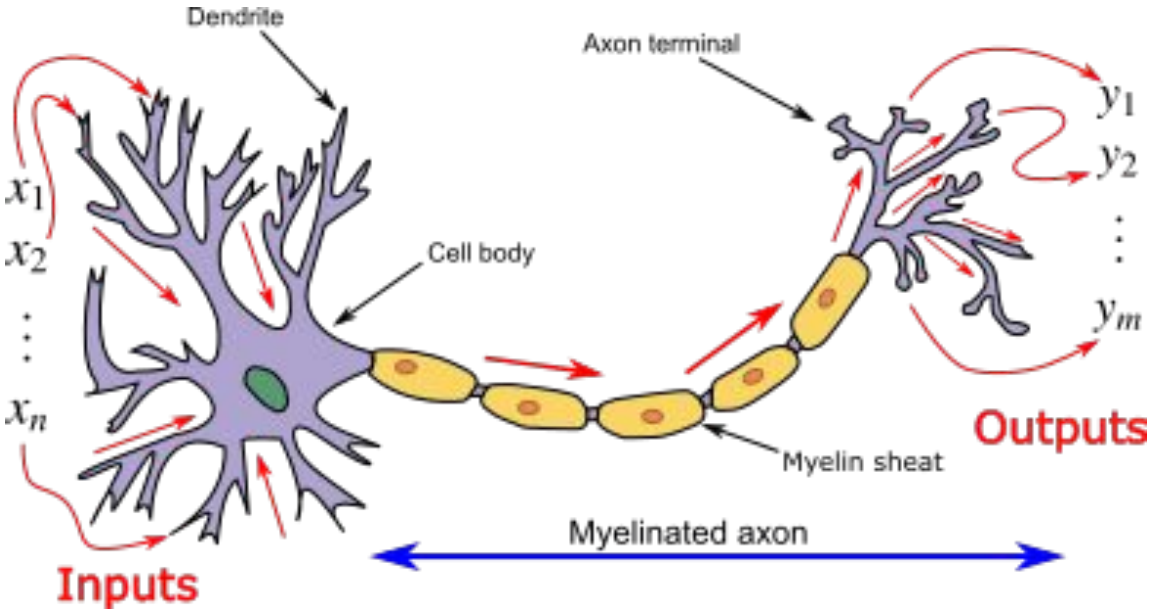
Hiperparaméterek keresése

Gradiensmódszert általában nem tudunk erre a célra használni (nem feltétlenül tudnánk deriválható költségfüggvényt megadni).

Elterjedt technikák:

- Manuális próbálgatás
- Keresés rácson (grid search)
- Véletlenszerű keresés
- Bayes optimalizáció
- Evolúciós algoritmus

A biológiai neuron modell



Kép forrása: Egm4313.s12 at English Wikipedia

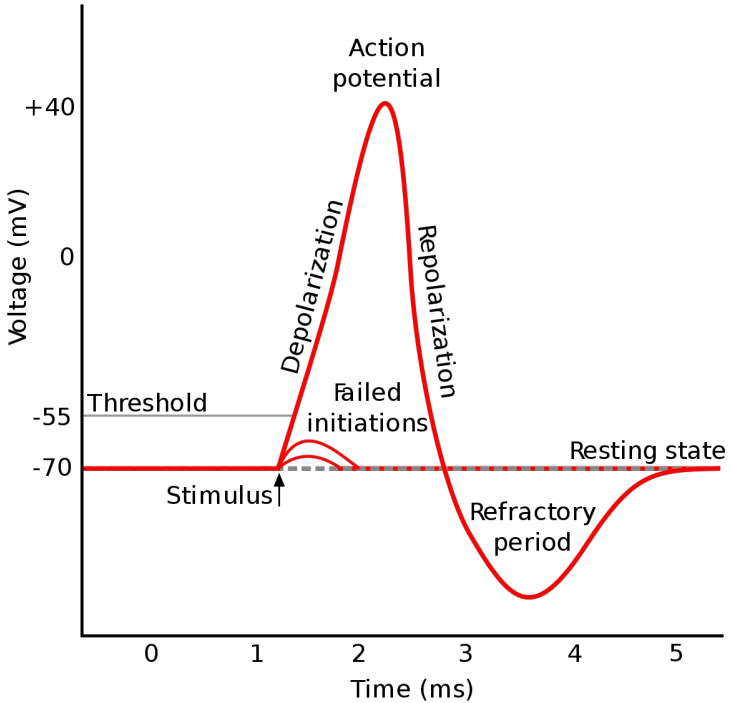
A biológiai neuron működése leegyszerűsítve

Kondenzátor-szerű működés:

- **Membránpotenciál:** A sejt belső és külső fala közti feszültségkülönbség
- Bemenet hiányában a membránpotenciál folyamatosan csökken egy nyugalmi szintig.
- A membránpotenciál input hatására nő.
- A különböző ágakon (dendriteken) bejövő input különböző **súlyokkal** lesz erősítve, vagy gyengítve (negatív súly is lehet).
- Ha a membránpotenciál elér egy (adott neuronra jellemző) **küszöböt**, akkor a neuron “tüzel”, a töltés továbbhalad a kimeneten keresztül

A biológiai neuron működése leegyszerűsítve

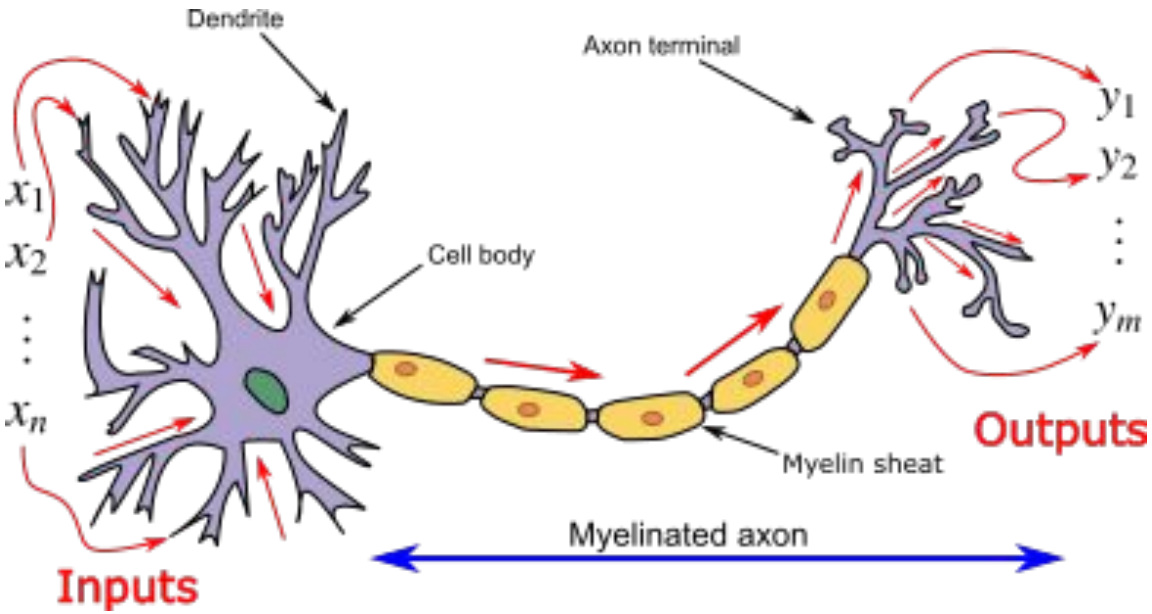
Kondenzátor-szerű működés:



Kép forrása: en:User:Chris 73, updated by en:User:Diberri at English Wikipedia

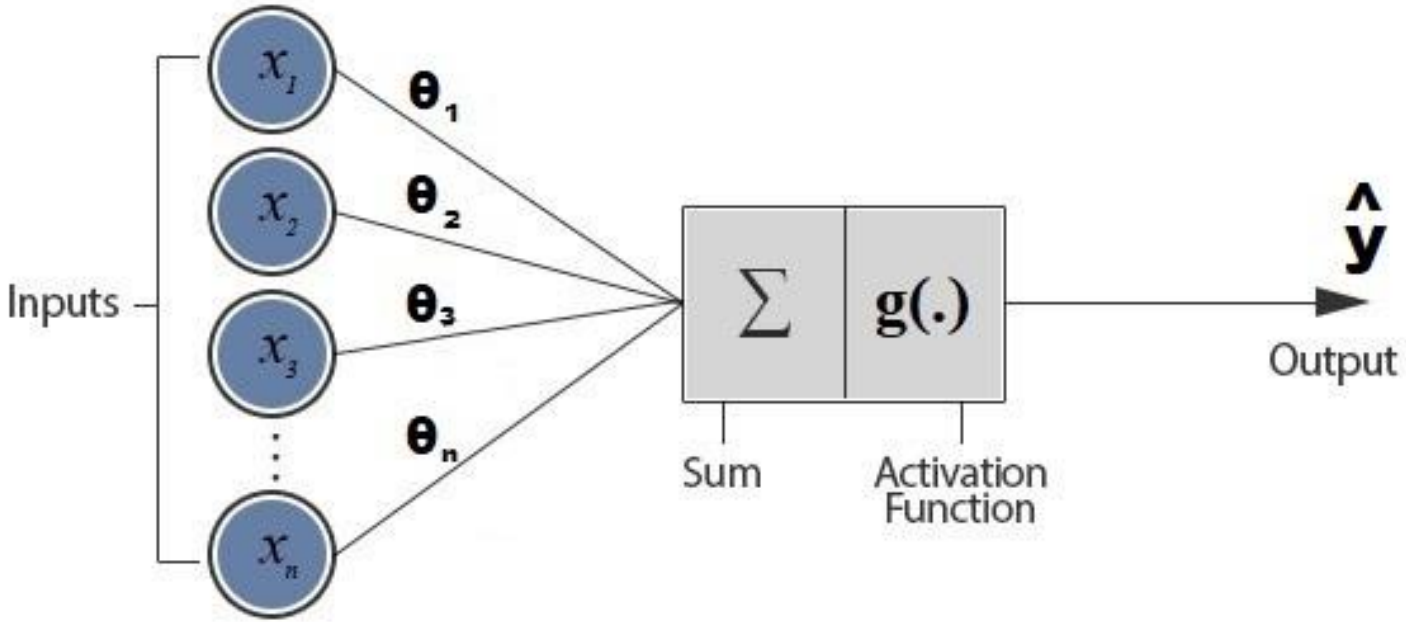
A biológiai neuron modell

Mire hasonlít a működése?



Kép forrása: Egm4313.s12 at English Wikipedia

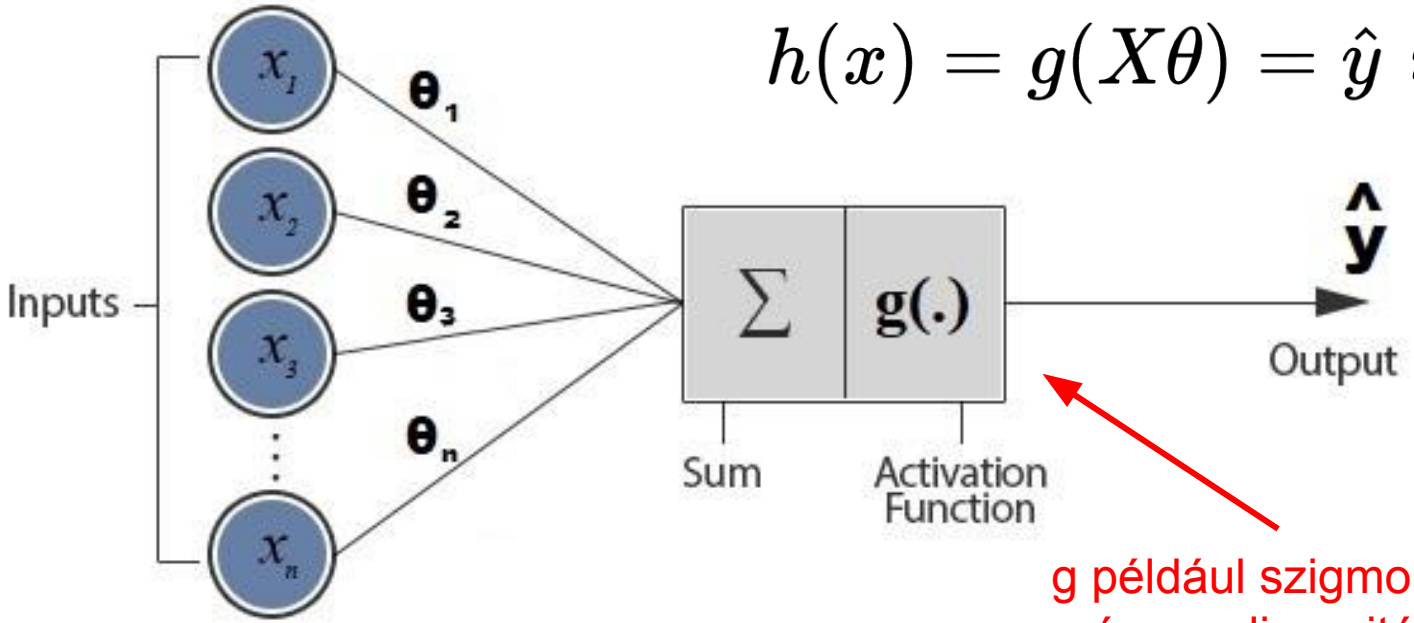
A mesterséges neuron modell (Rosenblatt, 1958)



A biológiai neuron, leegyszerűsített, diszkretizált modellje. **Mire hasonlít?**

A mesterséges neuron modell (Rosenblatt, 1958)

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

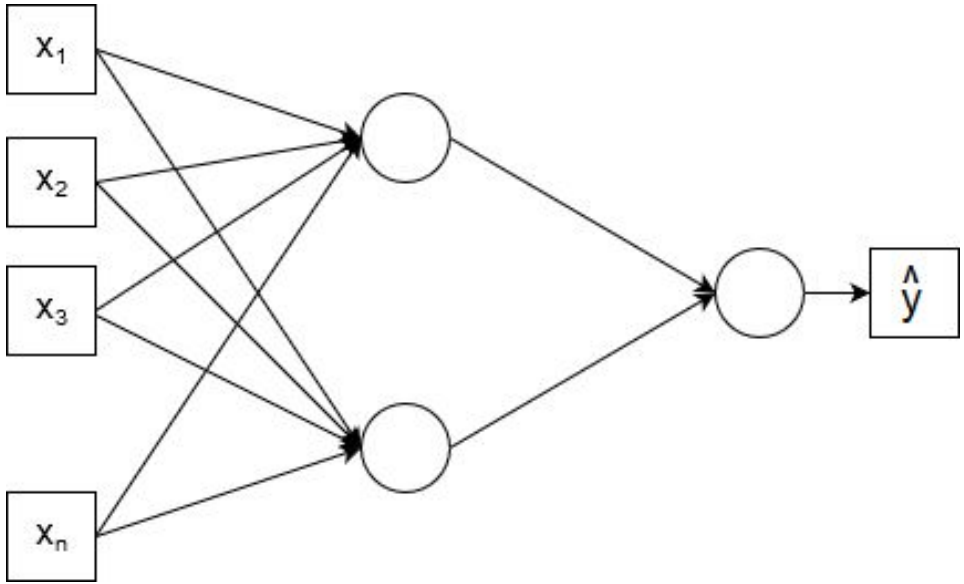


g például szigmoid, de más nemlinearitás is lehet

A mesterséges neuron egy (többváltozós) logisztikus regresszió, ha **g** szigmoid**!**

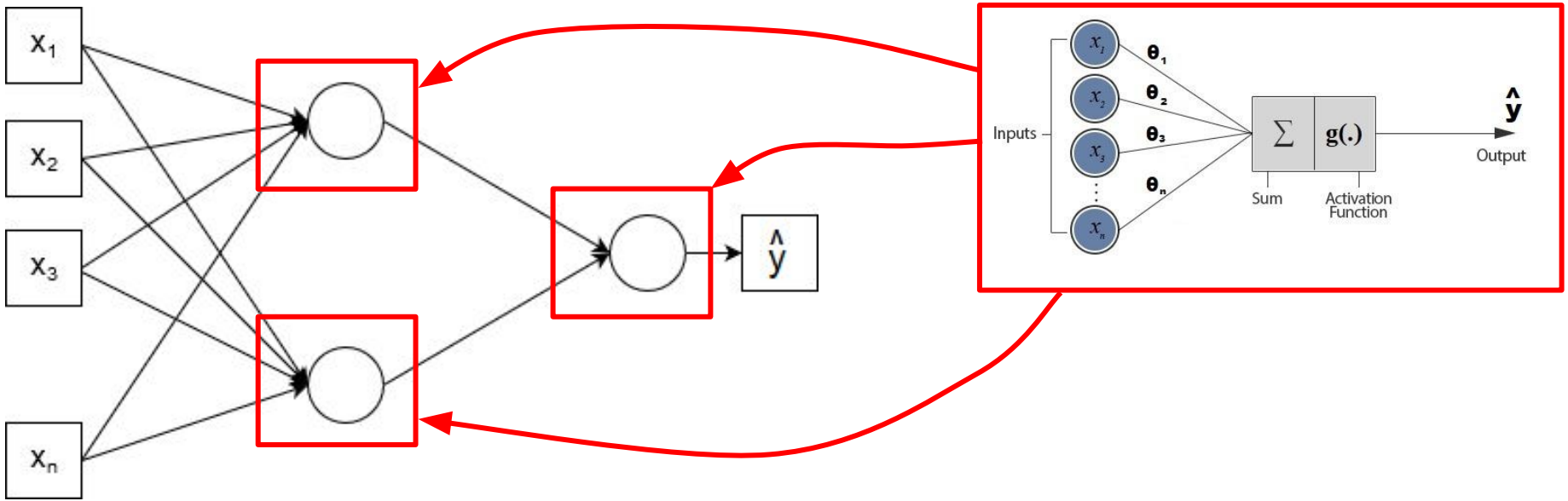
Mesterséges neuronháló felépítése

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



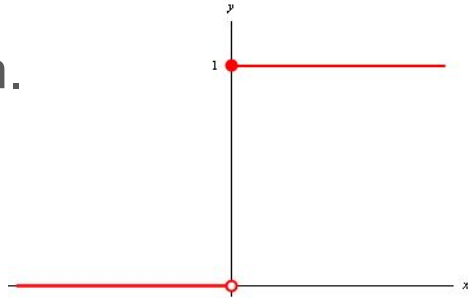
Mesterséges neuronháló felépítése

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



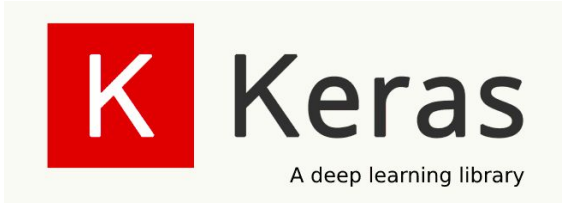
A biológiai neuron működése leegyszerűsítve

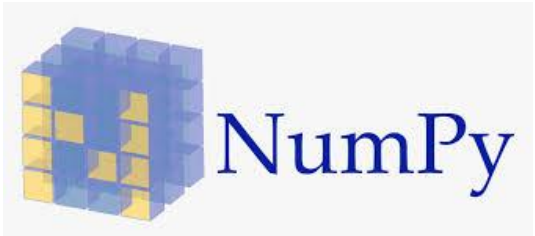
Különbségek a mesterséges neuron modellel:



- Diszkrét lépések helyett folytonos jel az idegsejtben.
  → **Mesterséges neuronban szumma,** idegsejtben integrál.
- Az idegsejtben a nemlinearitás nem kell, hogy folytonos legyen (Heaviside step function).
  → **Mest. neuronban folytonos, deriválható nemlinearitás kell.**
- Egy idegsejt súlya vagy mindig negatív, vagy mindig pozitív, az előjel nem változik meg (excitor vs. inhibitor)
  → **Mesterséges neuronban változhat a súly előjele.**

Szoftver neuronhálókhoz





csak CPU

CPU & GPU, számítási gráfok, automatikus deriválás

Logisztikus regresszió - NumPy v. Keras

```
def __sigmoid(self, z):
        return 1 / (1 + np.exp(-z) + self.eps)
```

hipotézisfüggvény NumPy-ban

```
h = self.__sigmoid(np.dot(X, self.theta))
```

```
model = Sequential()
model.add(Dense(1, activation='sigmoid'))
```

hipotézisfüggvény Keras-ban

Logisztikus regresszió - NumPy v. Keras

```
def __sigmoid(self, z):
        return 1 / (1 + np.exp(-z) + self.eps)
```

hipotézisfüggvény NumPy-ban

```
h = self.__sigmoid(np.dot(X, self.theta))
```

```
model = Sequential()
model.add(Dense(1, activation='sigmoid'))
```

hipotézisfüggvény Keras-ban

lineáris regresszióhoz **‘linear’**

Logisztikus regresszió - NumPy v. Keras

```
loss = np.mean(-y * np.log(h + self.eps) - \
               (1 - y) * np.log(1 - h + self.eps))
```

költség NumPy-ban

```
model.compile(loss='binary_crossentropy',\
              optimizer=sgd)
```

költség Keras-ban

Logisztikus regresszió - NumPy v. Keras

```
loss = np.mean(-y * np.log(h + self.eps) - \
               (1 - y) * np.log(1 - h + self.eps))
```

költség NumPy-ban

```
model.compile(loss='binary_crossentropy',\
              optimizer=sgd)
```

költség Keras-ban

költség lineáris regresszióhoz **‘mse’**

Logisztikus regresszió - NumPy v. Keras

```
intercept = np.ones((X.shape[0], 1))
X = np.concatenate((intercept, X), axis=1)
self.theta = np.zeros(X.shape[1])

for i in range(self.num_iter):
    h = self.__sigmoid(np.dot(X, self.theta))
    gradient = np.dot(X.T, (h - y)) / y.size
    self.theta -= self.lr * gradient
```

tanulás NumPy-ban

```
model.fit(X, y, epochs=num_iter)
```

tanulás Keras-ban

Összefoglalás

Számonkérhető:

- Alul- és túltanulás felismerése és kezelése, hiperparaméterek, validációs halmaz
- Mesterséges neuron modell

Nem lesz számonkérve a félévben:

- Polinomiális regresszió
- Polinom interpoláció
- Biológiai neuron modell és különbségek a mesterséges modellel