MESTERSÉGES NEURONHÁLÓK ÉS ALKALMAZÁSAIK

Készítette: Dr. Tóth László

A kapcsolódó gyakorlati anyag itt található:
http://www.inf.u-szeged.hu/~groszt/index.php?id=DLmaterials
(Készítette: Dr. Grósz Tamás)

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

SZÉCHENYI 2020
MAGYARORSZÁG KORMÁNYA
Európai Unió
Európai Szociális Alap
BEFEKTETÉS A JÖVŐBE

GÉPI TANULÁSI ALAPFOGALMAK

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

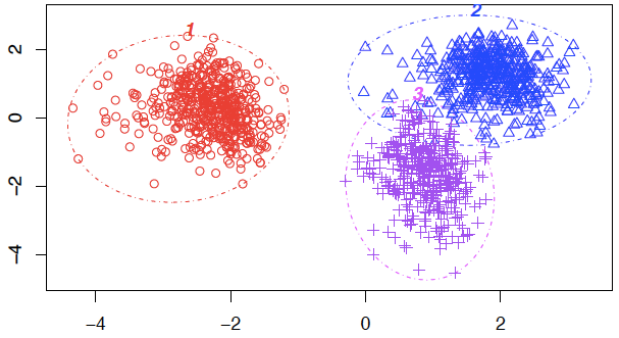
A gépi tanulás célja

- Cél: Olyan programok létrehozása, amelyek a működésük során szerzett tapasztalatok segítségével képesek javítani a saját hatékonyságukon
- Tanulóalgoritmus: Olyan algoritmusok, amelyek képesek szabályosságok, összefüggések megtalálására tanítópéldák egy halmaza alapján
  - 1. megj: A lényeg nem a konkrét tanulópéldák megtanulása, hanem a helyes általánosítás a tanulás során nem látott példákra is!
  - 2. megj: a feltételezett összefüggést „hipotézisnek” fogjuk hívni, ugyanis sosem lehetünk biztosak benne, hogy működni fognak a nem látott esetekre is
  - 3. megj: további példákat kapva a hipotézist javítjuk, az új példák alapján

Gépi tanulási feladatok típusai

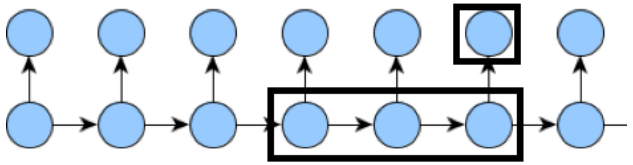
- Felügyelt tanulás: minden tanítópéldához meg van adva, hogy milyen választ várunk a géptől
  - Tipikus feladat: osztályozás
  - Példa: karakterfelismerés
  - Input: betűket ábrázoló képek

  - Output: (ezt kell a gépnek kitalálnia): milyen betű van a képen
  - Felügyelt tanítás esetén a tanítópéldákhoz (betűk képei) az elvárt választ is megadjuk (milyen betű van a képen)
- Felügyelet nélküli tanulás: az osztályokat is automatikusan kell megtalálnia a gépnek, pl. valamilyen hasonlóság alapján
  - Tipikus feladat: klaszterezés

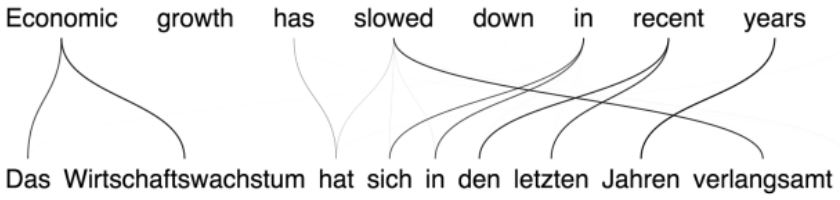


Feladattípusok (2)

- A neuronhálókat eredetileg az osztályozási feladat sémájára találták ki, azaz egy (fix méretű) inputra egy fix méretű outputot adnak
- Egyéb, fontos, speciális feladattípusok is léteznek, amelyek nem illenek ebbe a klasszikus sémába, pl:
- Időbeli folyamatok modellezése
  - Az aktuális kimenet a korábbi bemenetektől is függ(het), pl. videó felismerése, beszéd-felismerés, tőzsdei árfolyamok modellezése



- Sorozat→sorozat leképezés, pl. gépi fordítás
  - A sorozat hossza változó
  - Input-output hossza eltérhet
  - Nincsenek egyértelmű szó-szó párok, sorrend is keveredhet



Feladattípusok (3)

- Megerősítéses tanulás (reinforcement learning)
  - Pl. mesterséges „élőlények” létrehozása
  - Interakcióban van a környezetével, tapasztalatokat gyűjt, azokra reagál
  - Az osztályozással szemben itt nincs egyértelmű tanítócímke minden egyes eseményhez, csak egy hosszú távú cél (minél tovább életben maradni)
  - Kézzelfoghatóbb példa: sakk (vagy go) játszásanak megtanulása
  - Cél: a parti megnyerése, az egyes lépések nem egyértelműen minősíthetők jónak vagy rossznak
- Összegzés: a neuronhálós modellt alapvetően az osztályozási feladat megoldására találták ki, de újabban próbálják alkalmazni más jellegű, bonyolultabb feladatok megoldására is (önmagában, vagy más algoritmusokkal kombinálva)

Az osztályozási feladat

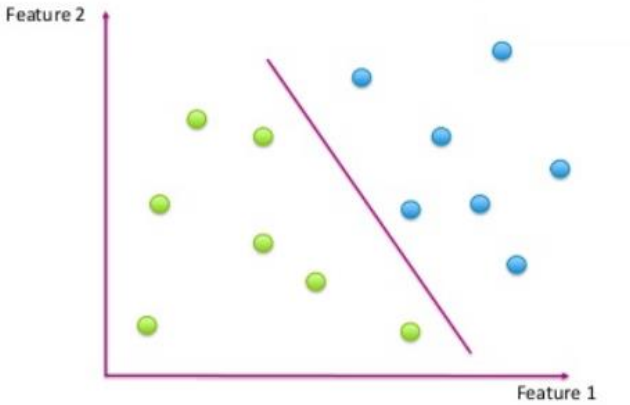
- A leggyakoribb gépi tanulási feladat
- Cél: objektum-példányok besorolása előre adott $c_1,\ldots,c_M$ osztályok valamelyikébe
- Input: valamilyen mérési adatokból álló (fix méretű) vektor
  - Jellemzővektor, feature vector, attributumvektor
- Tanítópéldák halmaza: jellemzővektorokból és hozzájuk tartozó osztálycímkékből álló párosok egy halmaza
  - Példa: Influenzás-e a beteg?

Feature vector			Osztálycímke (I/N)
Láz	Ízületi_fájdalom	Köhögés	Influenzás
38,2	Van	Nincs	Igen
36,7	Van	Száraz	Nem
41,2	Nincs	Nyákos	Igen
38,5	Van	Száraz	Igen
37,2	Nincs	Nincs	Nem

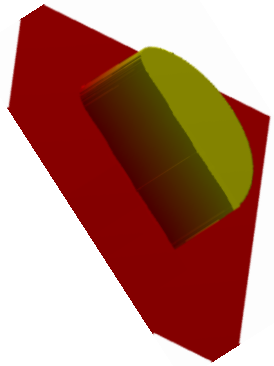
tanító-példányok

A jellemzőtér

- Ha a jellemzővektorunk N komponensből áll, akkor a tanítópéldáink egy N-dimenziós tér pontjaiként jeleníthetők meg
  - Példa:
    - két jellemző –> 2 tengely (x_1, x_2)
    - Osztálycímke (c): színekkel jelölve

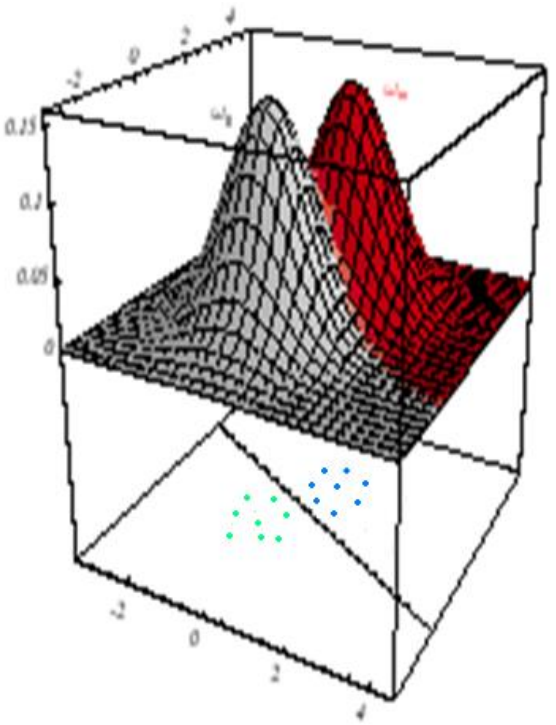


- A tanulási feladat tehát: becslést adni az (x1,x2)→c függvényre a tanítópéldák alapján
- M osztály esetén ez ekvivalens M db. (x1,x2)→{0,1} karakterisztikus függvény megtanulásával
  - A karakterisztikus függvény szakadásos, folytonos jellemzőtér felett nehezen reprezentálható
  - Kétféle módon tudjuk megoldani, hogy folytonos modellel tudjuk reprezentálni a fenti szakadásos függvényt



A döntési felület reprezentálása

- Direkt (geometriai) szemlélet: közvetlen módon a határoló felületeket reprezentáljuk
  - Valamilyen egyszerű módon, pl. egyenesekkel
- Indirekt (döntéselméleti) szemlélet:
  - Minden osztályhoz rendelünk egy függvényt, amely megmondja, hogy a tér egyes pontjai mennyire tartoznak az osztályhoz (tkp. olyan, mint a karakterisztikus fgv., de ez folytonos is lehet)
  - A diszkriminánsfüggvényekből úgy kapunk osztály-címkét, hogy a tér adott pontját az ott *maximális* értéket adó függvény osztályához soroljuk
  - Az osztályok határait indirekt módon a diszkriminánsfüggvények metszési görbéi határozzák meg



- A neuronhálók működése mindkét módon interpretálható!

A statisztikai alakfelismerés alapjai

- Az osztályozási feladat statisztikai alapú megközelítése
  - A gyakorlatban a gépi tanulás egyik legnépszerűbb irányzata
    - Szilárd matematikai háttér (valószínűségszámítás/statisztika)
    - Gyakorlatban is implementálható/használható megoldásokat ad
- Lényege a Bayes döntési szabály, amely matematikai formalizmust rak a döntéselméleti szemlélet mögé
  - Osztályozandó objektumok → **x** jellemzővektor
  - Feladat: az objektumok $\{c_1,..,c_M\}$ osztályok valamelyikébe sorolása
  - Cél: a téves besorolások számának minimalizálása hosszú távon
  - Bayes döntési szabály: adott **x** vektor esetén azt az *i* osztályt kell választani, amelyre $P(c_i|\mathbf{x})$ maximális
  - Azaz a döntéselméleti módszer optimális megoldást ad, ha diszkriminánsfüggvényként $P(c_i|\mathbf{x})$-et használjuk!
- Innentől alakfelismerés = $P(c_i|\mathbf{x})$ minél pontosabb becslése a tanítópéldák alapján!

Statisztikai alakfelismerés és neuronhálók

- A neuronháló működését értelmezhetjük az egyszerűbb „geometriai” szemlélet alapján
  - Azaz tekinthetjük úgy, hogy a háló az egyes osztályok közötti határvonalat igyekszik megkeresni
  - Ez az egyszerűbb, szemléletesebb magyarázat
- A neuronháló statisztikai alakfelismerési modellként is értelmezhető
  - Bebizonyítható, hogy bizonyos feltételek teljesülése esetén a neuronháló kimeneteit értelmezhetjük úgy is, mint az egyes osztályok $P(c_i|\mathbf{x})$ valószínűségére adott becslések
  - Ezek pedig megfelelnek a korábban döntéselméleti módszert által elvárt osztályonkénti diszkriminánsfüggvényeknek.
  - Ez a bonyolultabb értelmezés, viszont matematikailag sokkal hasznosabb (lehetővé teszi a modell matematikai értelmezését, elemzését…)
- A továbbiakban mindkét értelmezést megnézzük majd részletesebben

Kiértékelés

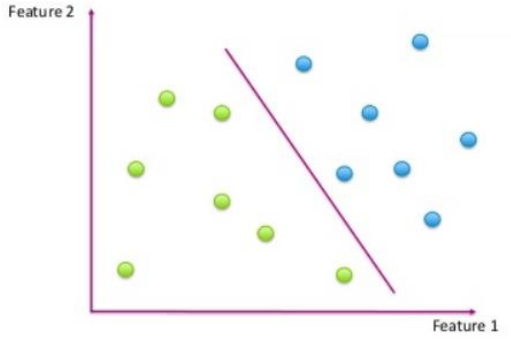
- A tanítás eredménye: a tanítópéldák alapján a tanítóalgoritmus készít egy modellt, azaz egy hipotézist a (x1,x2)→c függvényre
  - Ez a tér tetszőleges (x1, x2) pontjára meg tudja tippelni, hogy az melyik osztályba esik
- Honnan tudjuk, hogy ez a modell mennyire jó vagy rossz?
  - Megmérhetnénk, hogy a tanítópéldák hány százalékát osztályozza helyesen
- Fő célunk azonban nem a tanítópéldák címkéjének hibátlan megtanulása, hanem a tanítás során nem látott példákra való általánosítás!
  - A tanítópéldákon mért pontosság becsapós: kellően rugalmas modell jól be tudja tanulni a tanítópéldákat, azaz kicsi hibát ad rajtuk, mégis lehet, hogy rosszul általánosít a tanulás során látottaktól kicsit eltérő példákra

Kiértékelés (2)

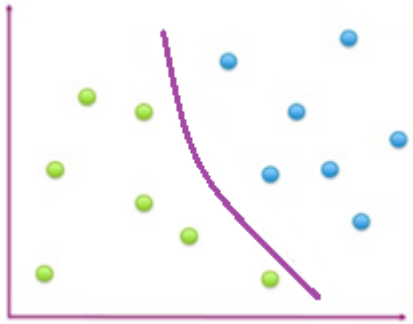
- Hogyan tudunk becslést adni az általánosító képességre?
  - A tanítóadatokat nem használhatjuk a fentiek miatt
  - Viszont mindenképp címkézett adat kell a tippelt és a valódi címkék összehasonlításához
- Megoldás: a tanítópéldáink egy részét félretesszük, nem használjuk fel a tanítás során
  - Az lesz az ún. teszthalmaz
- A tanítás+kiértékelés lépései tehát:
  - A tanító és –tesztadatok szétválasztása
  - Háló betanítása a tanító halmazon
  - A háló kiértékelése a teszthalmazon → tippelt címkék
  - A tippelt és a valódi címkék összehasonlítása a teszthalmazon
  - Osztályozási hibaszázalék = elrontott címkék száma/összes tesztpélda száma

A gépi tanulási modell paraméterei és metaparaméterei

- Minden gépi tanulási algoritmusnak vannak beállítandó paraméterei
  - Tekintsük a korábbi példát, ahol egy egyenessel igyekeztünk elválasztani két osztályt
  - A modell paraméterei ekkor az egyenes együtthatói
  - Ezeket optimalizáljuk a tanítás során, ezzel tudjuk az egyenes optimális irányba és pozícióba igazítani



- És mik azok a meta-paraméterek (más szóval hiperparaméterek)?
  - Általánosítsuk a fenti modellt úgy, hogy egyenes helyett polinomot használjon
  - A polinom fokszáma lesz az algoritmus metaparamétere
  - A paraméterekkel általában a modell adatokra való illeszkedését, a meta-paraméterekkel pedig a modell flexibilitását, reprezentációs erejét tudjuk állítani



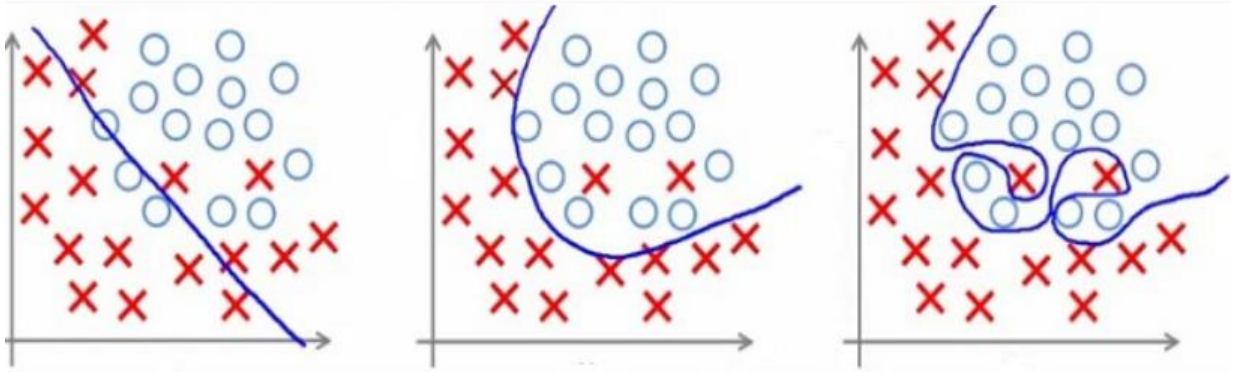
  - A paramétereket a tanítóalgoritmus hangolja be, a metaparamétereket általában mi választjuk meg kézileg, valamilyen heurisztika alapján
  - Neuronhálók esetén meta-paraméter: neuronok száma, rétegek száma,…

A gépi tanulási modell paraméterei és metaparaméterei

- Különböző paraméterértékek mellett kicsit különböző modelleket kapunk
  - Hogy tudjuk megtalálni az optimális meta-paramétereket?
- A tanítóhalmazból lecsípünk egy kisebb validációs (vagy development) példahalmazt
  - A példákat tehát train-dev-test halmazokra osztjuk
  - A tanítást a train halmazon többször elvégezzük, különböző metaparaméter-értékek mellett
  - A kapott modelleket kiértékeljük a development halmazon
  - Végül a development halmazon legjobbnak bizonyuló modellt értékeljük csak ki a teszthalmazon

A metaparaméterek szerepe

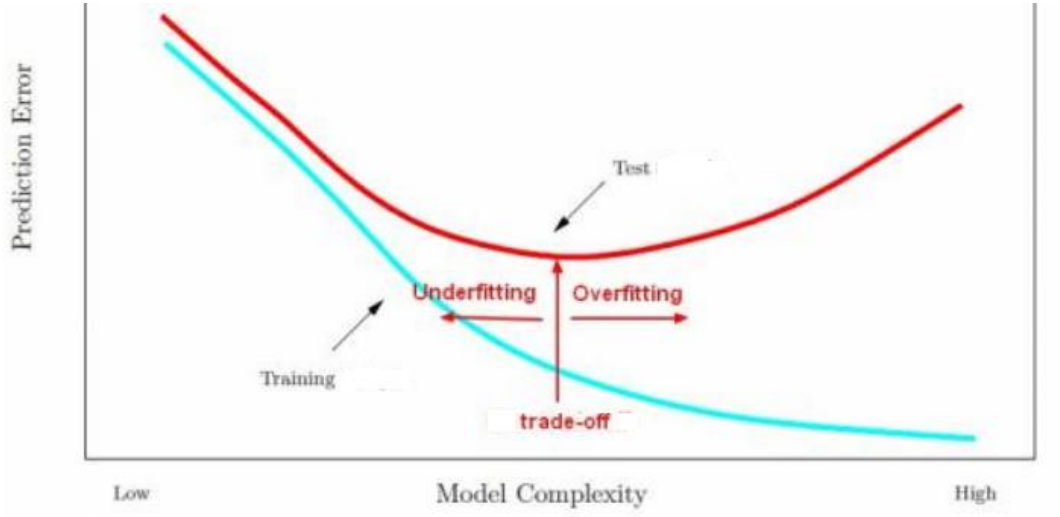
- Neuronháló esetén a paraméterek: a háló súlyai
- A fő metaparaméterek pedig: neuronok száma, rétegek száma,…
  - Ezekkel a háló méretét állítjuk
  - Nagyobb háló bonyolultabb döntési felületeket képes kialakítani



- Különböző metaparaméter-értékek mellett különböző modelleket kapunk
  - Hogy tudjuk megtalálni az optimális meta-paramétereket?
  - Használjunk nagyon rugalmas modellt?
  - Sajnos nem feltétlenül jó, mert a tanulópéldák félrevezetőek is lehetnek (véges mintát vettünk egy végtelen eloszlásból), vagy pl. hibásak (zajosak)

Általánosítás és túltanulás

- Végtelen jellemzőtér, véges számú tanító példa → sosem lehetünk biztosak benne, hogy a tanult modell jól általánosít a tanulás során nem látott esetekre
- A modell méretének növelésével a modell rugalmassága nő, egyre pontosabb lesz a tanítópéldákon
  - Viszont a tesztpéldákon egy ponton túl romlani fog a teljesítménye
  - Ezt hívják túltanulásnak (overfitting): a modell az általános tulajdonságok után már az aktuális véges mintahalmaz egyedi furcsaságait kezdi megtanulni



Optimális modellméret megtalálása

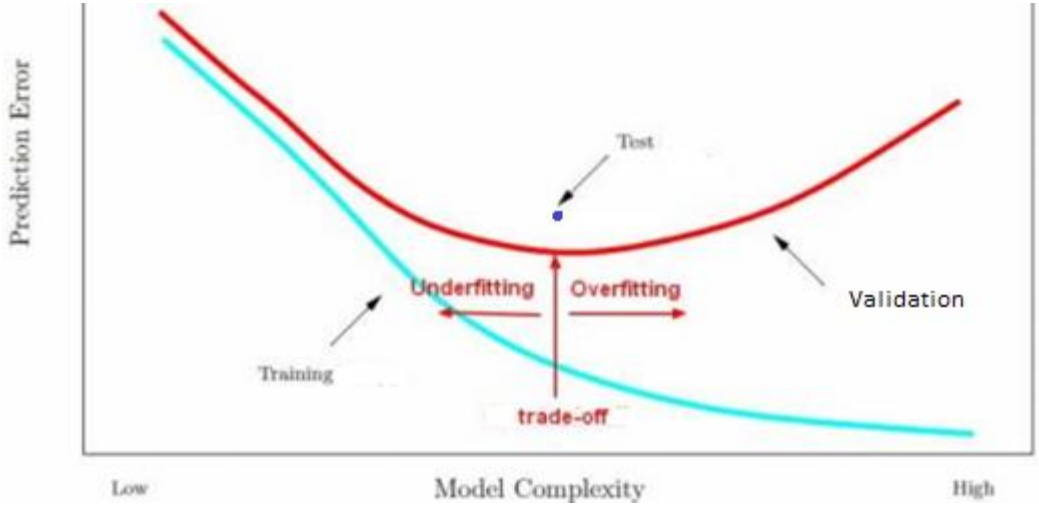
- Az optimális modellméret feladatonként nagyon eltérő lehet
  - A feladat jellege mellett függ a példaszámtól, jellemzők számától,…
  - Bonyolult elméleti háttere van: „No free lunch” tétel
- Gépi tanulási tapasztalatot igényel a belövése
  - bár egyre inkább vannak automatikus megoldások erre is…
- Néhány „hüvelykszabály”:
  - „There is no data like more data” - Minél több tanítópéldánk van, annál nagyobb hálózatot építhetünk a túltanulás veszélye nélkül
  - Érdemes kis jellemzőszámra törekedni („curse of dimensionality”)
    - Az (adatmennyiséghez képest) nagyon sok jellemző esetén szintén nő a túltanulás veszélye
  - Kevés adat és/vagy nagy jellemzőszám esetén érdemes regularizációs technikákat használni (ld. később)
  - Megjegyzés: a „deep learning”-nek csak nagyon nagy adatmennyiség mellett van igazán értelme…

A metaparaméterek belövése a gyakorlatban

- Különböző paraméterértékek mellett kicsit különböző modelleket kapunk
  - Hogy tudjuk megtalálni az optimális metaparamétereket?
  - Le kellene mérnünk a modell pontosságát különböző metaparaméter-értékek mellett egy teszthalmazon
  - A végső teszthalmaznak függetlennek kell lennie, mind a paraméterek, mind a metaparaméterek optimalizálása során használt adatoktól
- A tanítóhalmazból lecsípünk egy kisebb validációs (más szóval development) példahalmazt
  - A példákat tehát train-dev-test halmazokra osztjuk
  - A tanítást a train halmazon többször elvégezzük, különböző metaparaméter-értékek mellett
  - A kapott modelleket kiértékeljük a development halmazon
  - Végül a development halmazon legjobbnak bizonyuló modellt értékeljük csak ki a teszthalmazon

A metaparaméterek belövése a gyakorlatban

- Ábrával szemléltetve:
  - A tanítást a train halmazon többször elvégezzük, különböző metaparaméter-értékek mellett – megkapjuk a kék görbét
  - A kapott modelleket kiértékeljük a validációs halmazon – piros görbe
  - Végül a development halmazon legjobbnak bizonyuló modellt értékeljük csak ki a teszthalmazon – kék pötty



KÖSZÖNÖM A FIGYELMET!

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.