# REGULARIZÁCIÓ

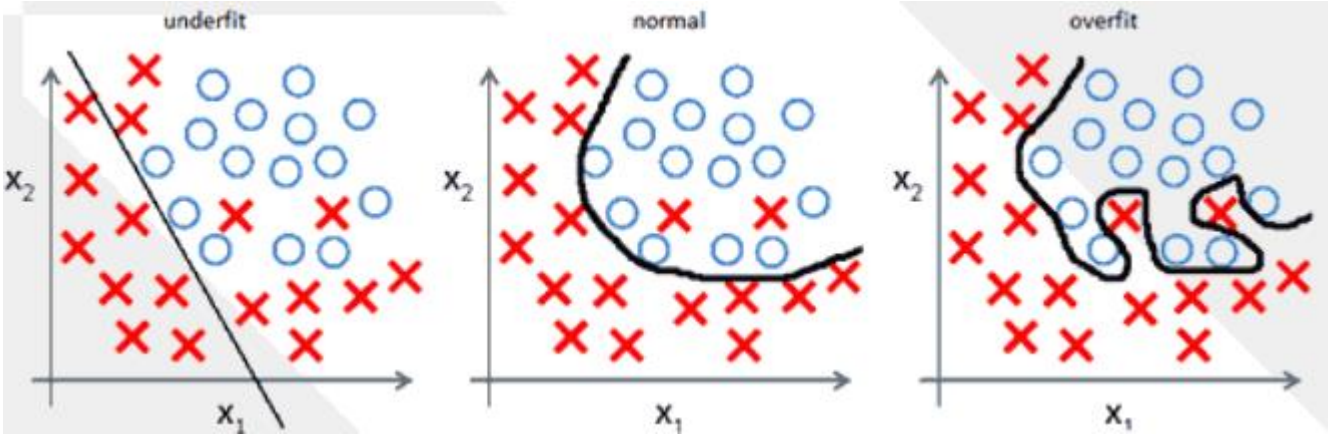
A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

# Mi az a túltanulás?

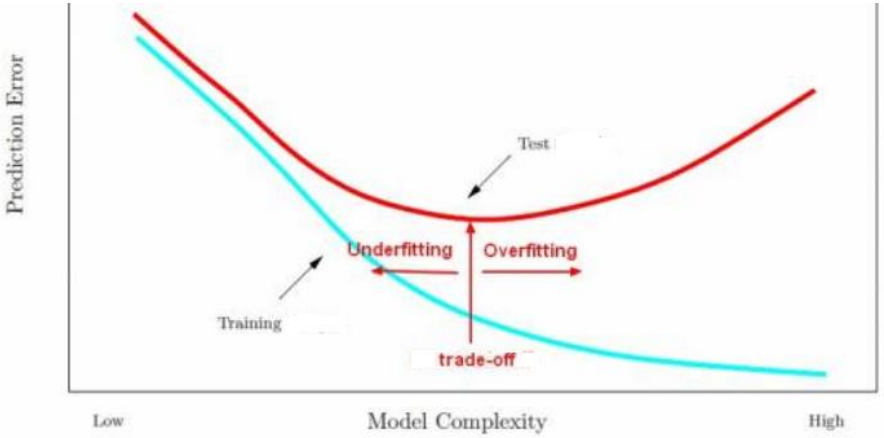
- Példa: leszáll a földön egy ufó, azzal a feladattal, hogy térképezze fel, milyenek az emberek
  - Lát 10 embert, és ebből levonja a következtetést: az embereknek két lábuk van, két kezük, egy fejük, és barna a szemük
  - Míg az eső három valóban igaz, a negyedik tulajdonság csak az adott 10 emberre teljesül → túltanulás
- Túltanulás (overfitting): a modell az általános tulajdonságok mellett az aktuális véges mintahalmaz egyedi furcsaságait is megtanulni
  - Nyilván csökkenthető az esélye a példaszám növelésével, de végtelen tér esetén véges számú példa sosem lesz tökéletesen reprezentatív
  - A gépi tanulási modell rugalmasságának növelésével nő a túltanulás esélye (a modell jobban tud az apró részletekre figyelni)
  - A túltanulás csökkentése miatt kulcsfontosságú, hogy a modellt a tanítóhalmaztól független teszthalmazon is kiértékeljük

# A túltanulás jellemzői

- A modell méretének (komplexitásának) növelésével a modell rugalmassága nő, így egyre pontosabb lesz a tanítópéldákon




- Viszont a tesztpéldákon egy ponton túl romlani fog a teljesítménye
- Ezért lehet egy független teszt (vagy validációs) halmazon való kiértékeléssel megkeresni az optimális modell-komplexitást



# A túltanulás neuronhálóknál

- Neuronhálók esetén a modell rugalmasságát alapvetően a neuronok és a rétegek számának növelésével növelhetjük
- Emellett backpropagation tanítás esetén a túltanulás jellemzően a tanítás vége felé jelentkezik


- Ezért a túltanulás ellen a legalapvetőbb eszközök:
  - A tanítás korai leállítása
  - A kis paraméterszámra való törekvés

# „Early stopping”

- Ezt legkönnyebben egy validációs példahalmaz segítségével tudjuk megvalósítani
- Erről már beszéltünk a learning rate állításának ismertetésekor
  - Kicsit most frissítjük
- A learning rate frissítése validációs példahalmaz segítségével
  - A tanítás előtt tegyük féle az adatok egy részét → validációs halmaz
  - **Valamennyi tanítási lépés előtt mentsük el az aktuális súlyokat**
  - (pl. 1 epoch) után értékeljük ki a hibát a validációs halmazon
  - Ha az előző kiértékeléshez képest a hibacsökkenés kisebb, mint egy küszöb → learning rate felezése, új iteráció
  - **Ha hiba nőtt volna, visszaállítjuk az előző súlyokat**
  - Teljes leállás: ha a learning rate lement egy küszöb alá **vagy a validációs hiba több lépés során is nőtt**

# Regularizáció

- A kis paraméterszámra való törekvés csökkenti a túltanulás esélyét
- Azonban a tanulás hatékonyságát is ronthatja
  - Adott paraméterszámú modellből nem tudjuk kihozni a maximumot, mivel az optimalizáló algoritmusaink nem garantálnak globális optimumot
- Ezért a gyakorlatban jobb stratégia paraméterszámban „túltervezni” a modellt, viszont kiegészíteni a hibafüggvényt egy plusz taggal, például:

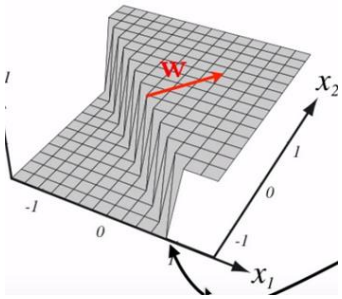
$$E = \frac{1}{2} * \sum (t_k - o_k)^2 + \frac{\lambda}{2} * \sum {w_i}^2$$

négyzetes hiba — regularizációs hiba

- Ez a regularizáció - a λ súllyal a regularizáció erőssége szabályozható
- A hibafüggvény regularizációs tagjának feladata, hogy az optimalizálás során az egyforma hibájú, de különböző alakzatot leíró modellek közül az optimalizáló az „egyszerűbbet”, kevésbé „túltanultat” preferálja
  - Ennek gépi tanulási modellenként más-más lehet az értelmezése, most megnézzük neuronhálók esetére

# Neuronhálók regularizációja

- Neuronhálók esetén a döntési felület sok-sok hipersíkból áll össze, közvetlen módon nem tudjuk befolyásolni a simaságát
- Nézzük, hogy mit tehetünk pl. egyetlen neuron esetén
  - Az aktivációk egy hipersíkra esnek: $a = \sum_{i=0}^{n} w_i x_i$
  - A döntési felület egy hipersík, ahol $0 = \sum_{i=0}^{n} w_i x_i$
  - A sigmoid függvény a hipersík két végét 0-hoz és 1-hez „görbíti”




- Ha súlyokat megszorozzuk egy nagy konstanssal, akkor a döntési felület ugyanott marad, de a hipersík meredekebb lesz, így a sigmoid meredekebben tart 0-hoz ill. 1-hez
  - Azaz nagyobb súlyok esetén a neuron „határozottabb” döntést hoz, amit a tanulás elején nem szeretnénk (túltanulás)
  - Korábban megbeszéltük, hogy a tanításra nézve is rossz (a derivált kicsi lesz), ha a $\sum_{i=0}^{n} w_i x_i$ érték nagy
  - Ezért inicializáltuk a súlyokat kis értékekkel, ezért normalizáltuk a jellemzőket 1 szórásúra

# Weight decay

- A legegyszerűbb regularizációs módszer a hálót a kis súlyok preferálására készteti – ez a weight decay
- Képlettel (itt most négyzetes hiba esetére, de az alap hibafüggvény más is lehet):

$$E = \underbrace{\frac{1}{2} * \sum (t_k - o_k)^2}_{\text{plain error}} + \underbrace{\frac{\lambda}{2} * \sum w_i^2}_{\text{L2 weight penalty}}$$

- Miért hívják weight decay-nek?
  - Emlékezzünk, hogy a backprop szabály szerint $w_i = w_i - \eta \frac{\partial E}{\partial w_i}$
  - A fenti regularizációs hibafüggvénnyel $w_i = w_i - \eta\lambda w_i = (1 - \eta\lambda) w_i$
  - Azaz a súlyokat meg kell szorozni egy egynél picit kisebb konstanssal
  - Ennek hatására a súlyok - legalábbis amelyeket a négyzetes hiba nem akar mindenáron nagy értékűnek megtartani - a nulla felé tartanak
- Hívják L2 regularizációnak is (a súlyvektor 2-es normája a regularizációs tag)

# Sparse neuronhálók

- A kettes helyett  az egyes normát is használhatjuk – ez lesz az L1 avagy „Lasso” regularizáció

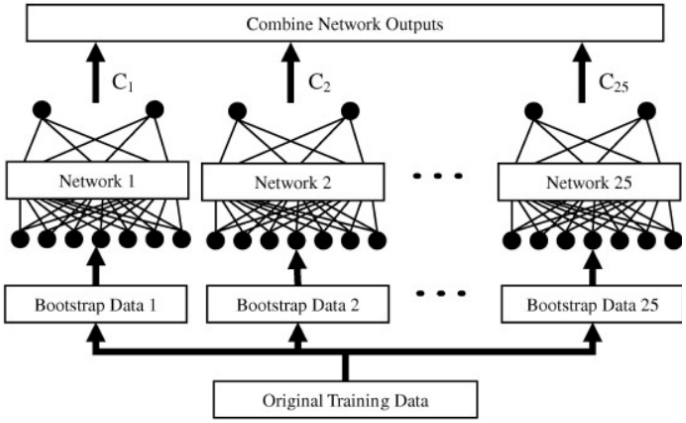
$$E = \frac{1}{2} * \sum (t_k - o_k)^2 + \lambda * \sum |w_i|$$

squared error  L1 weight penalty

- Ilyenkor a súlyokat nem egy egynél kisebb konstanssal szorozzuk, hanem egy kis konstanst hozzáadunk-kivonunk (előjeltől függően)
  - Ez jóval aggresszívebben regularizál, nem csupán nulla felé nyomja a súlyokat, hanem ki is tudja nullázni
- Nem csak a súlyokra, hanem a neuronok kimenetére is használhatjuk, így a kevésbé fontos neuronokat is kinullázás felé nyomjuk (pl. ReLU neuronokkal, mi ténylegesen is tud 0 kimenetet adni)
- Ezzel a hálót „fully connected” helyett ritka (sparse) hálóstruktúra felé tudjuk irányítani (élek vagy teljes neuronok eltüntetése)
  - Így tkp. visszajutunk az eredeti túltanulás elleni felvetéshez (kevés paraméter), csak egyfajta „soft” változatban

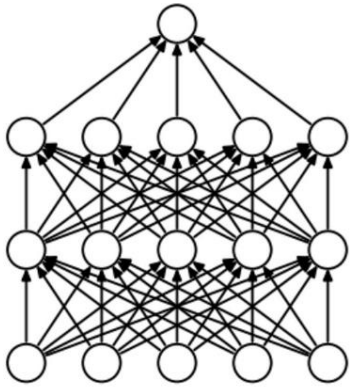
# Ensemble neural networks

- Ha nem nagyon nagy a tanító adatbázis, akkor megtehetjük, hogy a hálót sokszor betanítjuk
  - A véletlen inicializálás és az adatok random bejárása miatt kicsit eltérő súlyokat fogunk kapni
  - De azt is megtehetjük, hogy az adatok valamilyen véletlenszerű (vagy egyéb, ravaszabb módon kiválasztott…) részhalmazain tanítunk
- A kapott modellek kimeneteit végül kiátlagoljuk, így egy „ensemble” modellt kapunk
  - Ez az átlagolás is stabilizálja a modellt, hiszen az áltagolás miatt kevésbé lesz érzékeny az egyes részhalmazok megválasztásából eredő esetleges egyedi furcsaságokra

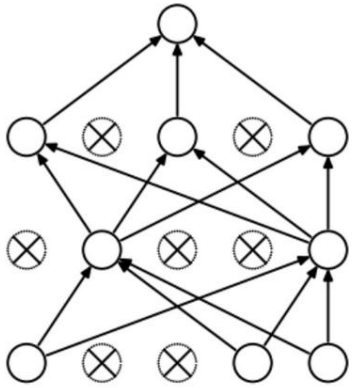


# Dropout

- A tanítás során az egyes neuronokat bizonyos valószínűséggel (pl. 0.2) „kiejtjük”, kinullázzuk (az adott input vektor esetén, a következő példánál már „élni” fog!)
- Ez a neuronokat arra kényszeríti, hogy önállóan tanuljanak, kevésbé támaszkodjanak a szomszédokra
- Miért regularizál?
  - Megmutatható, hogy olyan hatása van, mint ha a háló össze részhálóját, azaz $2^N$ hálót tanítanánk párhuzamosan, és ezeket átlagolnánk



(a) Standard Neural Net



(b) After applying dropout.

- Technikai problémák
  - Lassítja a tanulást, jóval (5-10x) többször kell végigmenni az adatokon
  - Egy adott neuronnak kevesebb inputja lesz → a megmaradt inputokat felszorozzuk egy konstanssal, hogy az összeg hasonló skálára essen

# N-szeres keresztvalidáció

- Ezt a módszert akkor szoktuk használni, ha tényleg nagyon kevés tanítópéldánk van
  - Luxus lenne validációs célra „elherdálni” akár a példák 10% is.
- A példáinkat felosztjuk N egyenlő részre (mondjuk N=10)
- 9 részen tanítunk, a tizediken validálunk
  - És ezt megismételjük 10-szer, mindig másik halmazt kihagyva validálásra
  - Ily módon az összes adaton tudunk tanítani, de azért a független adaton való validálás is megvolt
  - Validálási pontosság: a 10 validálási pontosság átlaga
- De hogyan lesz egy végleges modellünk a 10-ből?
  - Átlagolhatjuk a modellek kimeneteit (ld. ensemble model)
  - Esetleg csinálhatunk egy végleges tanítást (validálás nélkül!) az összes adaton a legjobbnak bizonyult meta-paraméterekkel
- Extrém eset: „leave-one-out” – egyetlen példát hagyunk ki validálásra

# Data augmentation

- Ez nem regularizációs módszer, de a túltanulás esélyét csökkenti
- Tudjuk, hogy a túltanulás fő oka a kevés tanítópélda
- Hogyan tudunk a meglevőkből mesterségesen további tanítópéldákat kreálni?
- A példákon kisebb transzformációkat alkalmazunk, olyanokat, amelyek az osztálycímkét nem befolyásolják
  - Pl. karakterfelismerés: eltolás, kicsinyítés-nagyítás, kisebb fokú elforgatás
  - Pl. beszédfelismerés: pár százaléknyi lassítás-gyorsítás, háttérzaj hozzáadása
- Ezekkel csökkenthetjük a háló érzékenységét ezekre a transzformációkra, így nőni fog az általánosító képessége

# Noise injection

- Pici (általában normál eloszlású) zajt adunk az inputhoz, vagy akár az egyes neuronok kimenetéhez is
  - Tulajdonképpen a data augmentation egy speciális, „buta” esete (amikor mondjuk nem tudjuk, hogy hogyan érdemes az adatokat transzformálni)
  - A tapasztalatok szerint ez is segít a túltanulás ellen, főleg amikor nagyon kevés a tanító adat
  - De olyan változatok is vannak, ahol a hibafüggvény deriváltjához, sőt akár a tanítócímkékhez adnak zajt

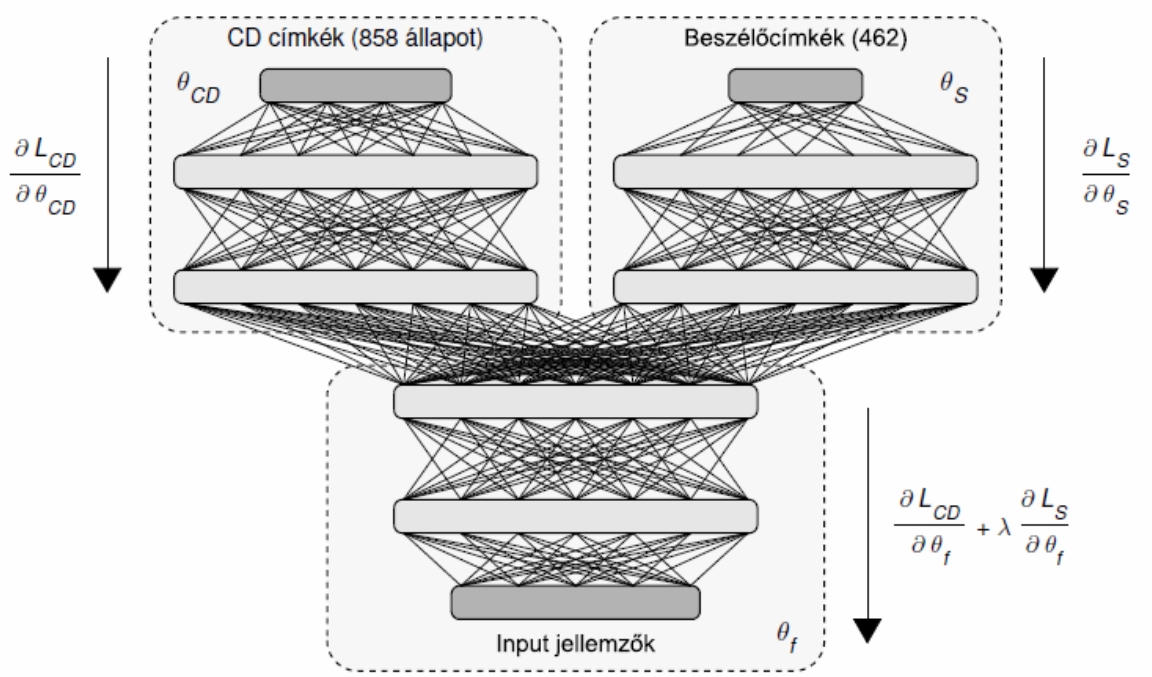
  - https://machinelearningmastery.com/train-neural-networks-with-noise-to-reduce-overfitting/

# Multi-taszk tanítás

- A neuronhálónak párhuzamosan több (általában 2) tanulandó feladatot adunk – ehhez speciális hálózati struktúra kell
- Alapvetően arra találták ki, hogy két feladatot egyszerre oldjanak meg, de:
  - Megfigyelték, hogy gyakran mindkét feladaton jobb eredményt kapnak, mint ha külön-külön hálót tanítanának rájuk
  - Azért, mert az „egyben” tanulásnak regularizáló hatása van
- Példa beszédfelismerésből (ld. következő ábra): a főfeladat a beszédhangok felismerése, a másodlagos feladat a beszélő személyének felismerése

# Multi-taszk hálóstruktúra

- A két feladat párhuzamos megoldásához speciális hálózati struktúra kell:
  - Közös input réteg és alsó rejtett rétegek
  - Feljebb feladat-specifikus rétegek
  - Két kimenő réteg
  - Két hibafüggvényt minimalizálunk
  - Hibavisszaterjesztés során a két ágból érkező hibát súlyozottan összegezzük (λ paraméter)



# Multi-taszk tanítás

- Nyilván akkor van értelme, ha a két feladat hasonlít, de nem ugyanaz
- $\lambda$ mellett egy további paraméter az is, hogy milyen mélységben ágaztassuk el a hálót
  - Ésszerűnek tűnik, hogy minél eltérőbbek a feladatok, annál korábbi rétegbe kell helyezni az elágazást (kevesebb közös és több feladatspecifikus réteg kell)
  - De ezt jelenleg csak kísérleti úton lehet belőni
- Mire jó a multi-taszk tanítás (amellett, hogy 2 feladatot oldunk meg egyszerre)?
  - A hálónak a közös rétegekben olyan reprezentációt kell találnia, ami mindkét feladat megoldását segíti
  - Ennek regularizáló hatása van (kisebb túltanulás, jobb általánosítás)
  - Ezért gyakran az eredmények is jobbak mindkét feladaton, mint ha külön-külön oldanánk meg őket

# Ellenséges (adversarial) multi-taszk tanítás

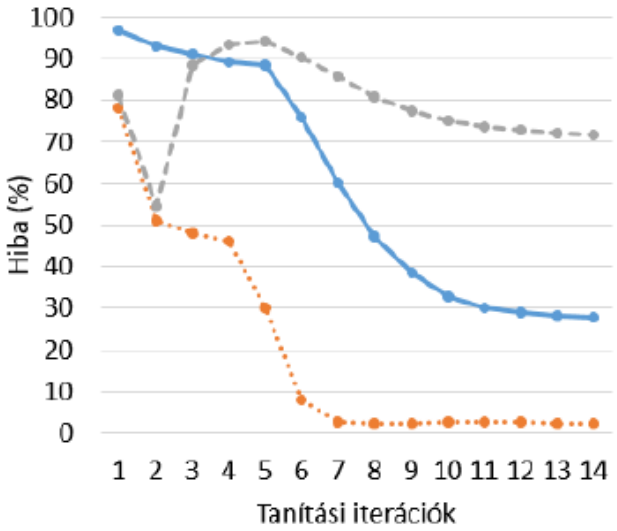
- Egy kis csavart visz a multi-taszk modellbe
- A másodlagos feladat hibáját minimalizálás helyett maximalizáljuk
- Technikai kivitelezés: λ értékét negatívnak választjuk, a másodlagos ág hibáját továbbra is minimalizáljuk
  - A feladatspecifikus rétegek továbbra is minimalizálásra törekednek
  - Az előjelváltás a legfelső közös rétegben fejti ki a hatását
  - Csak egy előjelet kell módosítani a kódban
  - λ abszolút értékét fokozatosan, c lépésben érdemes növelni
  - Azaz a k-adik iterációban

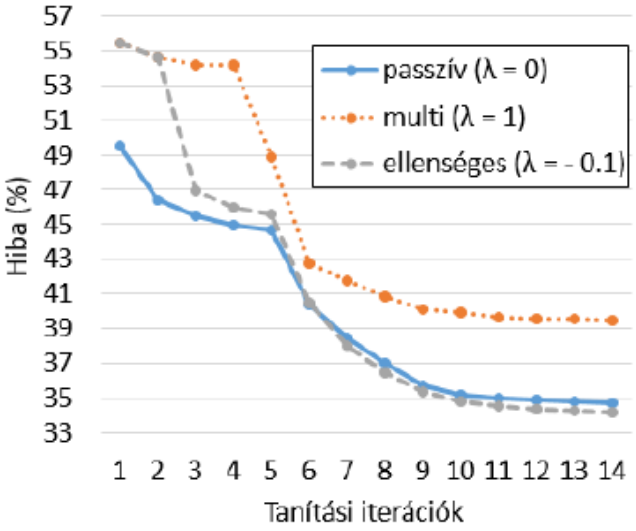
$$\lambda_k = min(\frac{k}{c}, 1) \cdot \lambda$$

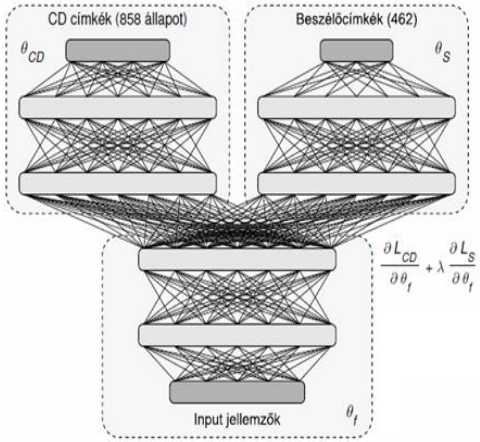
- Hatása: a háló olyan közös reprezentációt keres, ami alapján a másodlagos feladatot minél kevésbé lehet megoldani
  - Az előző példában beszélőfüggetlen reprezentációt fog keresni

# A tanítás szemléltetése a példán

- Bal oldal: A másodlagos (beszélőfelismerési) ág a tanítóhalmazon
- Jobb oldal: a fő feladat hibája a fejlesztési halmazon




- „passzív” tanulás: a másodlagos ág tanul, de nem szólhat bele a közös rétegekben kialakuló reprezentációba (a főág „egyfeladatosan” tanul)
- Eredmény: a multi-taszk ront, az ellenséges multi-taszk kicsit javít az elsődleges (beszédfelismerési) feladat pontosságán

# KÖSZÖNÖM A FIGYELMET!

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