Neurális hálózatok gyakorlati feladatok

Készítette:

Dr. Molnár András 2015. április 5.

Rev.: 1.0

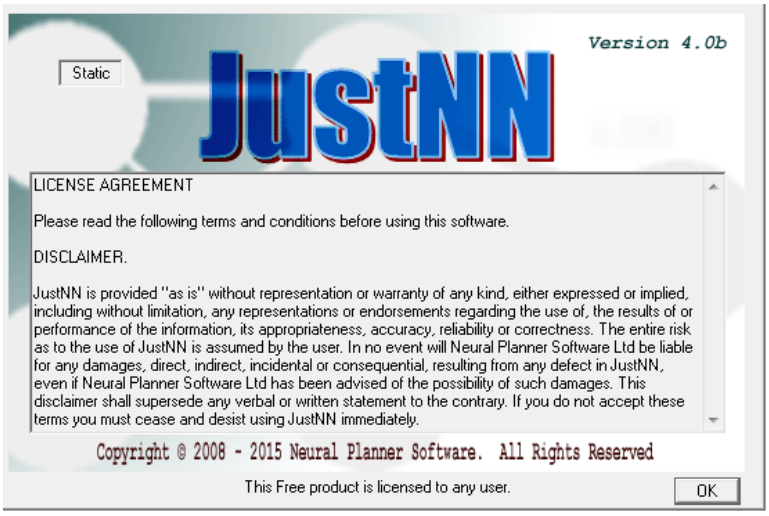
A gyakorlatokat a „JustNN” szimulátor segítségével végezzük. Az egyes feladatok teljes értékű megoldásához szükséges az „Excel” szoftver alapvető ismerete.

A „JustNN” programot az alábbi címről tölthetjük le:

http://www.justnn.com/

Telepítést követően a program azonnal használható.

A feladatok az alábbi változattal lettek tesztelve:



1. feladat: ismerkedés a „JustNN”programmal

A telepített program néhány előre elkészített és betanított példát is tartalmaz. Célszerű először ezeket betölteni és ezeken keresztül megismerni a „JustNN” programot. Vizsgáljunk meg néhány egyszerű példát (XOR) amiknek eredményét magunk is ismerjük. Inicializáljuk újra a súlyokat, vizsgáljuk meg a hálózat működését tanulás előtt és után!

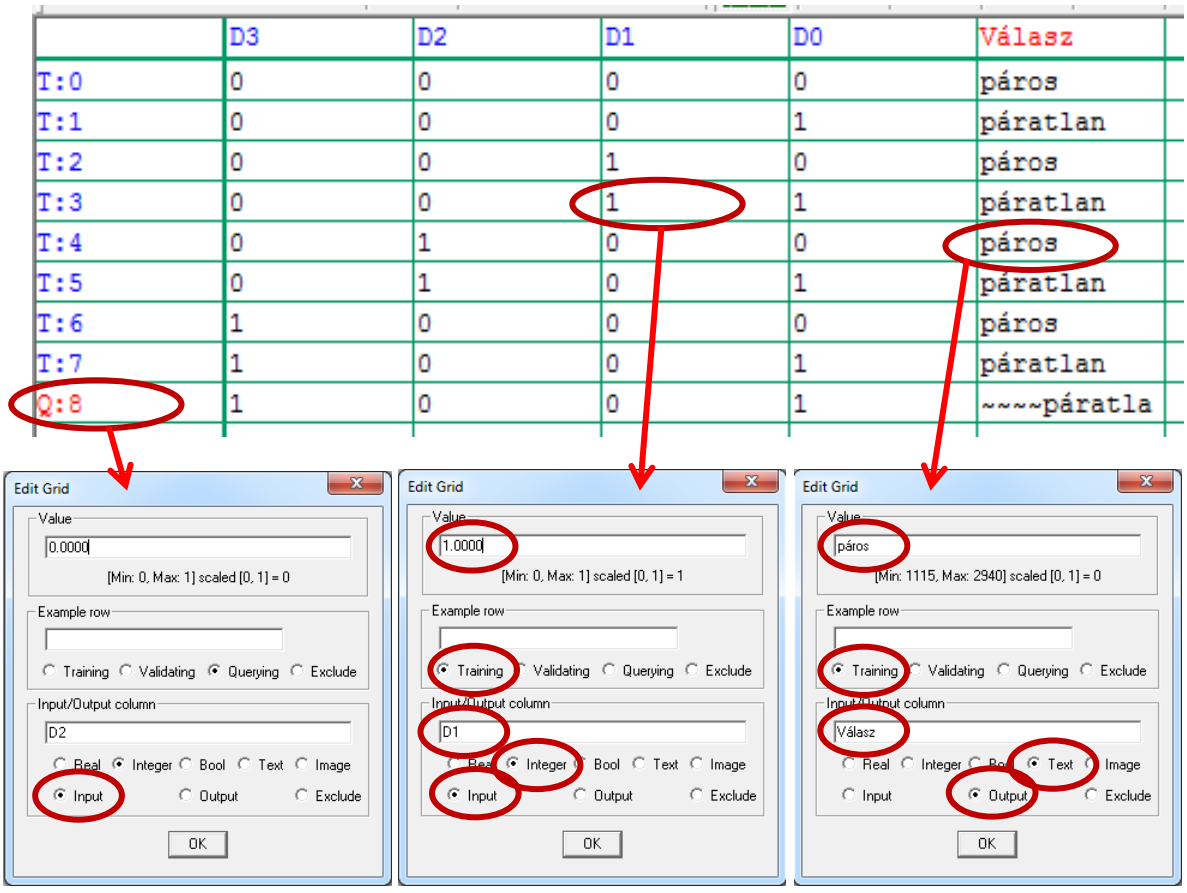
Ezt követően elkészíthetjük az első, egyszerű neurális hálózatunkat.

A hálózat feladata, hogy döntse el egy négybites számról, hogy páros vagy páratlan! A feladat nagyon egyszerű, de éppen ezért jól megfigyelhető általa a hálózat működése. Nyilvánvaló, hogy a kérdés szempontjából (páros vagy páratlan) csupán a legalacsonyabb helyértékű bit releváns. Amennyiben ez a bit 1, a szám páratlan, ellenkező esetben pedig páros.

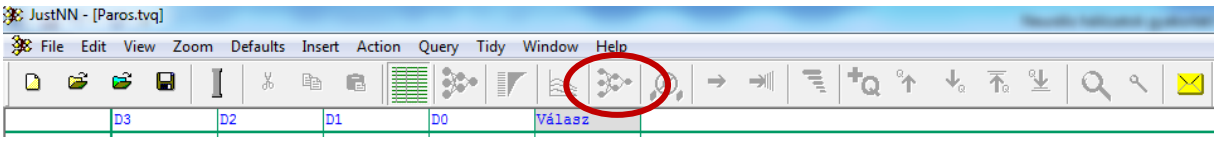
A bemenő adatok (D3, D2, D1, D0) lehetnek logikai (Bool) változók, de lehetnek egészek (Int) is. Ez utóbbi esetben áttekinthetőbb a táblázat, de ügyelnünk kell arra, hogy csak 0-t és 1-t írjunk a bemeneti mezőkbe!

A kimeneti mező lehet szöveges (Text), így a hálózat eredménye kissé látványosabb, bár minőségét tekintve egyenértékű bármely más adattípussal.

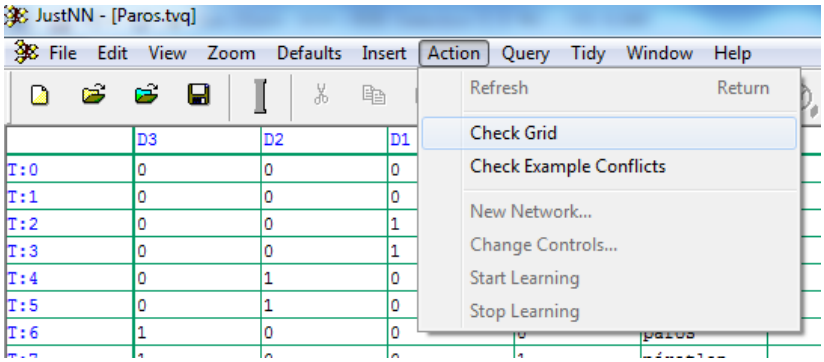
Figyelem! Az egyes mezők tulajdonságait oszloponként egyszer kell beállítani!



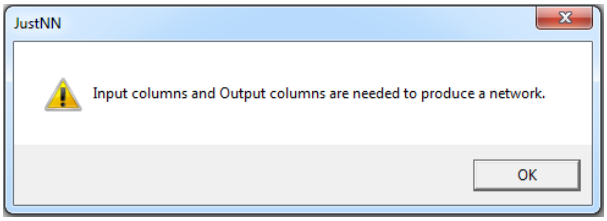
Az adatok bevitelét követően létrehozhatjuk a hálózatot. Ennek a műveletnek van néhány előfeltétele (pl. léteznie kell tanító soroknak, kimeneti mezőnek). Ha minden feltétel teljesül, a hálózat varázsló ikon aktív, ellenkező esetben szürke.



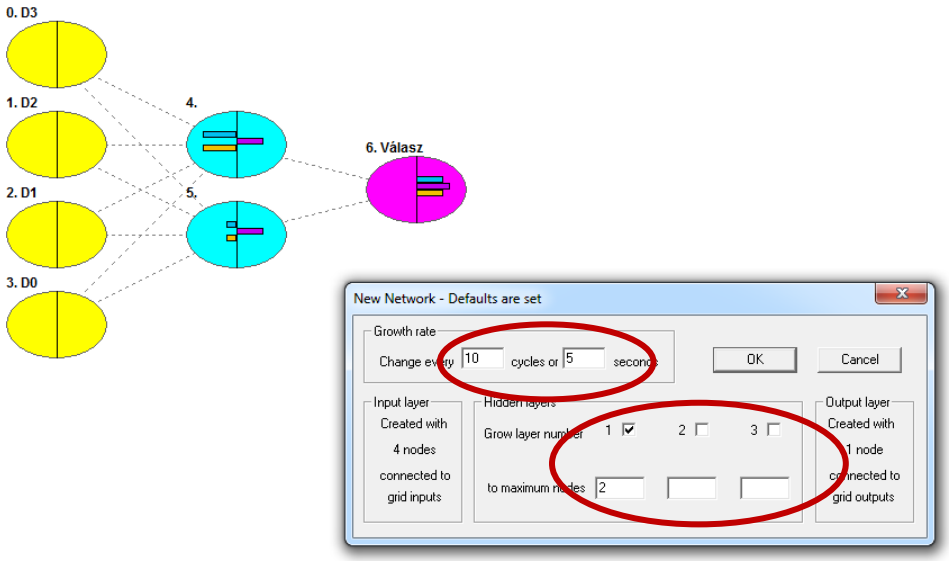
Amennyiben a fenti ábrán látható módon szürke a hálózatgeneráló ikon, miközben az adatbevitellel végeztünk, „kérdezzük meg a programot” a hiba okáról. Ehhez a legördülő menüből válasszuk az „Action” opciót és azon belül a „Check Grid” szolgáltatást.



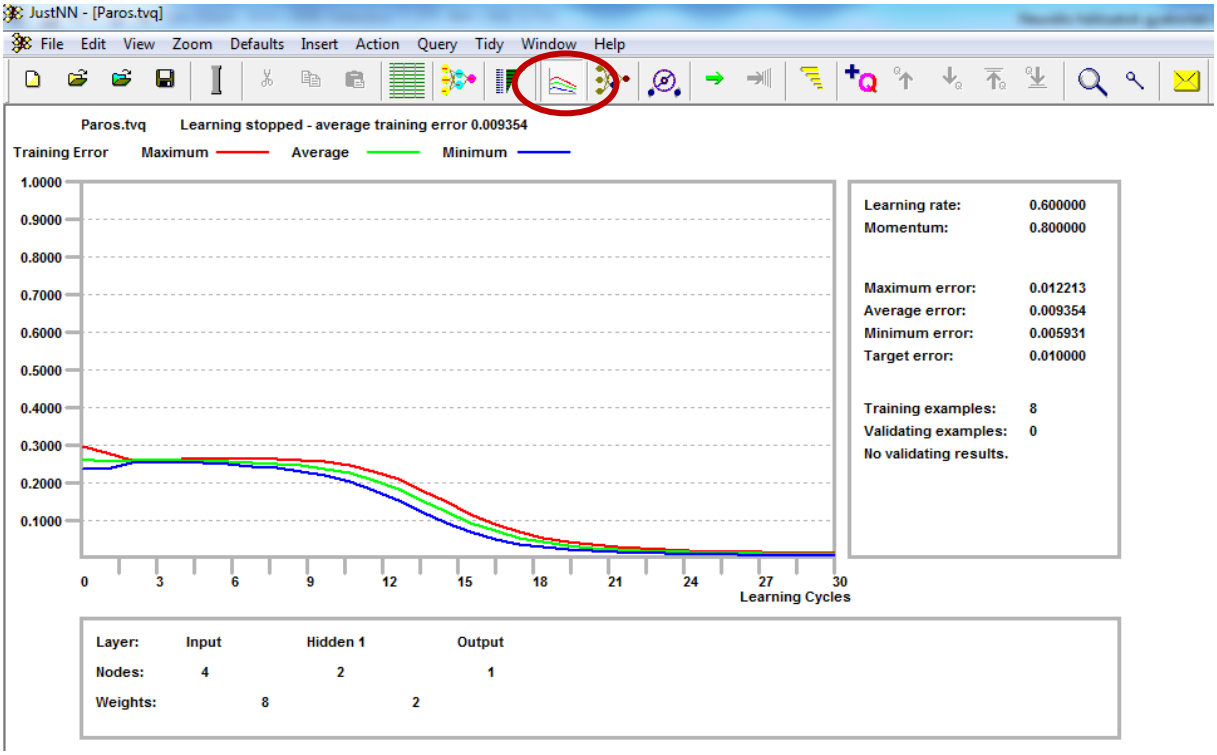
Ekkor kapunk egy válasz, amely a legvalószínűbb hiba okát tartalmazza:



Jelen esetben a kimeneti adatoknak gondolt oszlop vagy nem lett létrehozva, vagy nem lett kimenetnek beállítva. Hárítsuk el a hibát és hozzuk létre a hálózatot.

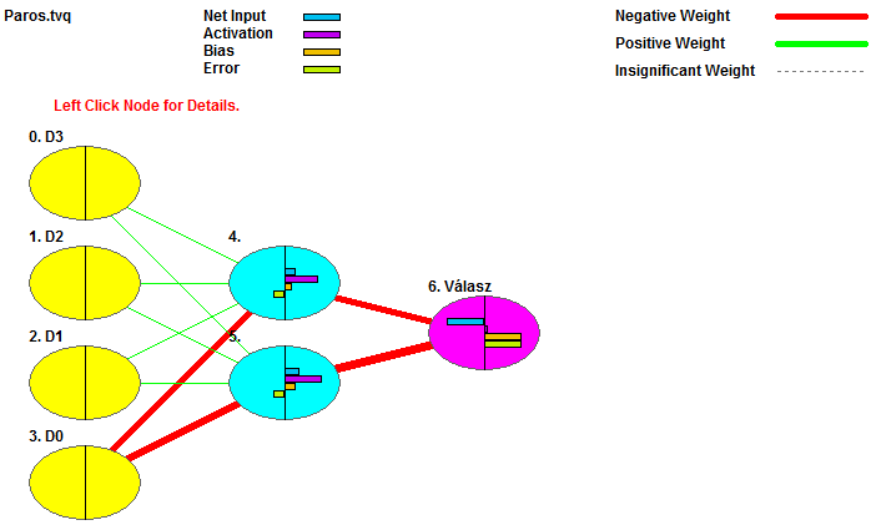


A hálózat az alap beállításokkal az adott problémára szinte azonnal „betanul”, amit a tanulás eredményét, illetve folyamatát szemléltető nézetben is jól láthatunk.



Figyeljük meg, hogy a hálózat számára nem adtuk meg az összes lehetséges bemeneti állapotot (16 állapot), hanem csupán 8-at! Mégis mind a 16 esetre jó választ fog adni.

A „JustNN” program segítségével azt is megvizsgálhatjuk, hogy a hálózat mely bemenő adatai, illetve mely súlytényezők játszanak jelentős szerepet a válasz generálásában.



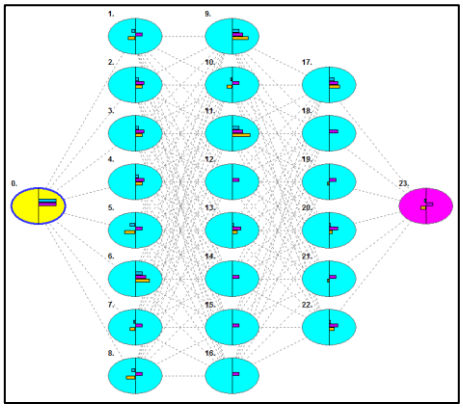
A fenti ábra alapján azt látjuk, hogy a páros/páratlan kérdés eldöntésében a D0 bitnek van kiemelkedő szerepe, ami nem meglepő, hiszen a feladat elején már ezt a tényt megállapítottuk.

2. feladat: másodfokú függvény

Készítsünk egy olyan neurális hálózatot, amely az alábbi átviteli függvényt valósítja meg -10≤x≤10 intervallumban:

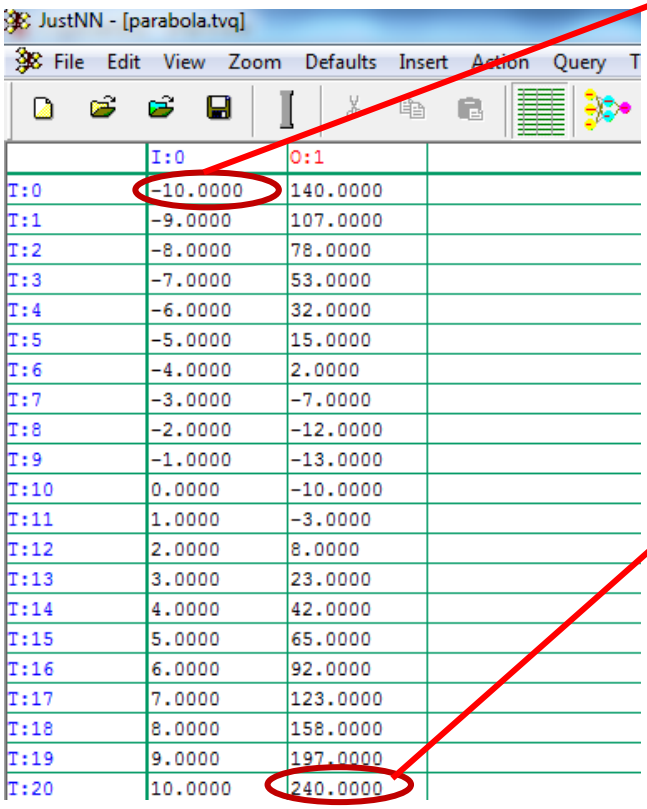
$$y = 2x^2 + 5x - 10$$

azaz

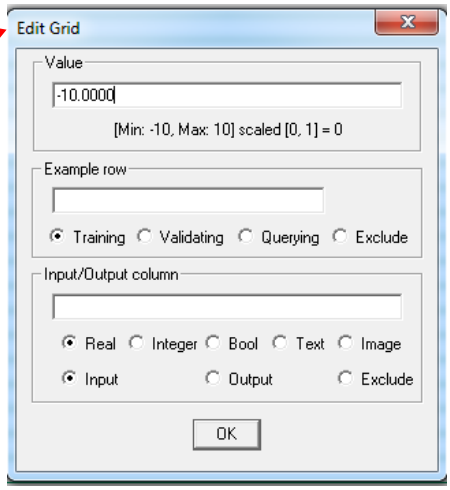
$x \rightarrow$  $\rightarrow y$

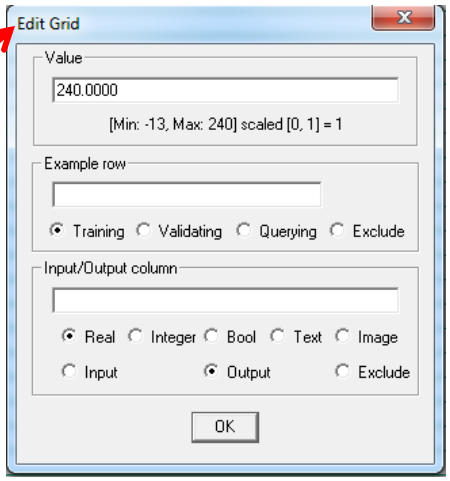
1. lépés:

Állítsuk elő a hálózat számára a reprezentatív tanító mintákat! A bemenetek (**Input**) legyenek valós (**Real**) értékek és állítsuk be a sort tanító mintának (**Training**)!



	I:0	O:1
T:0	-10.0000	140.0000
T:1	-9.0000	107.0000
T:2	-8.0000	78.0000
T:3	-7.0000	53.0000
T:4	-6.0000	32.0000
T:5	-5.0000	15.0000
T:6	-4.0000	2.0000
T:7	-3.0000	-7.0000
T:8	-2.0000	-12.0000
T:9	-1.0000	-13.0000
T:10	0.0000	-10.0000
T:11	1.0000	-3.0000
T:12	2.0000	8.0000
T:13	3.0000	23.0000
T:14	4.0000	42.0000
T:15	5.0000	65.0000
T:16	6.0000	92.0000
T:17	7.0000	123.0000
T:18	8.0000	158.0000
T:19	9.0000	197.0000
T:20	10.0000	240.0000

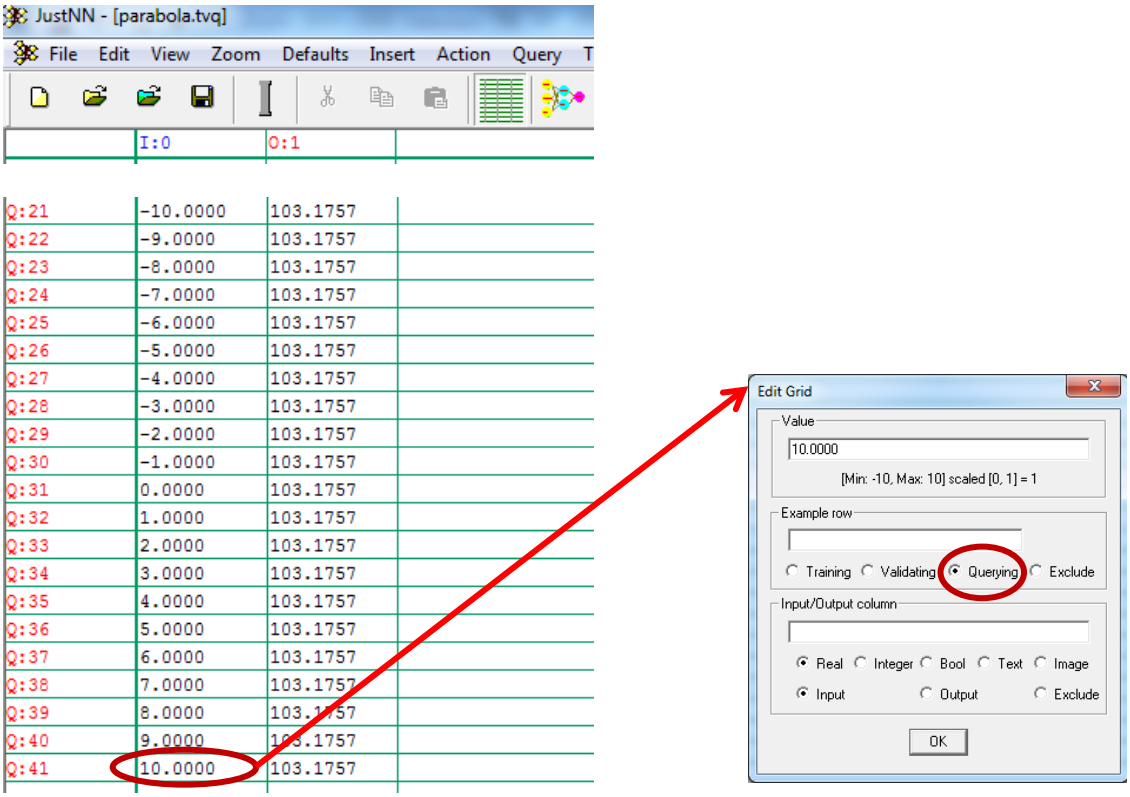




A kimenetek (**Output**) legyenek valós (**Real**) értékek és állítsuk be a sort tanító mintának (**Training**)!

2. lépés:

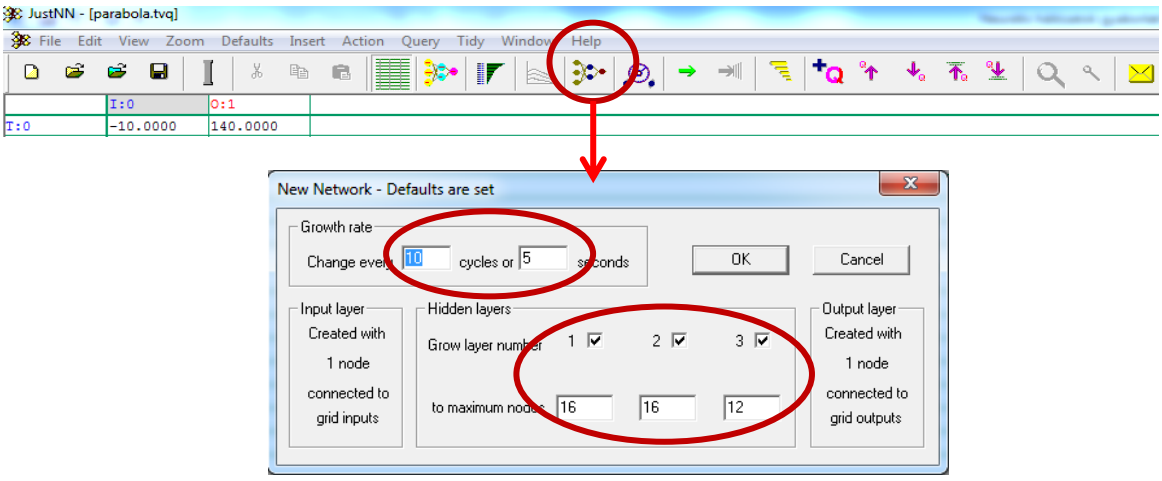
Hozzuk létre az ellenőrző sorokat! A betanulását követően innen olvashatjuk ki a hálózat által adott válaszokat az egyes bemeneti értékekre (X). Először célszerű ugyanazokra a bemeneti értékekre vizsgálni a válaszokat, mint amiket a betanításnál alkalmaztunk.



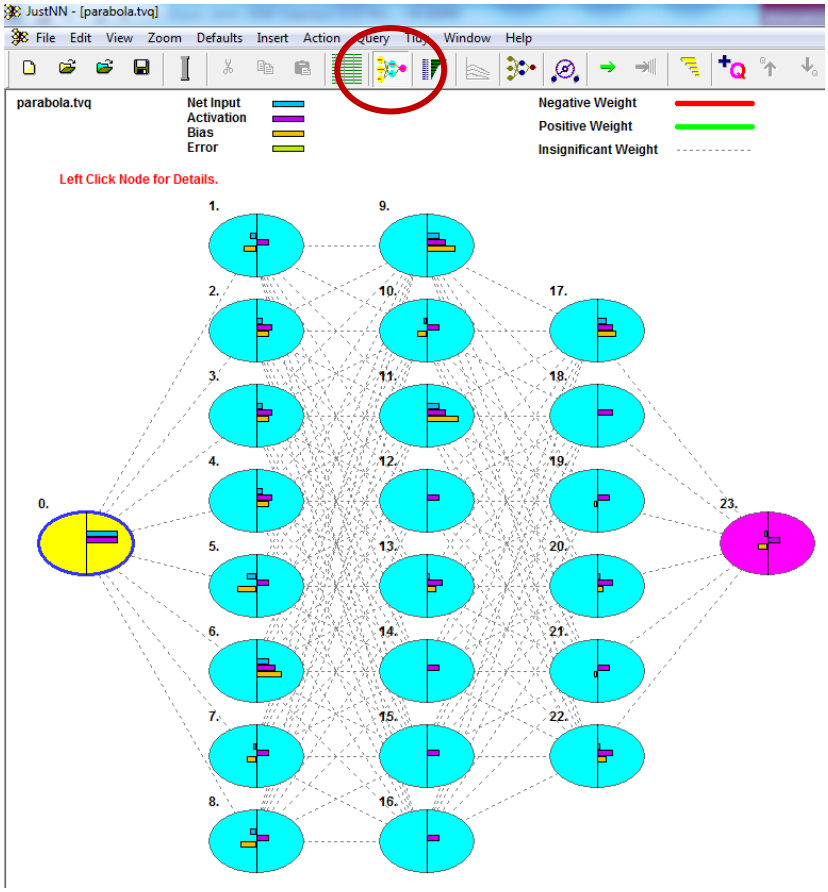
A bemenetek (**Input**) legyenek valós (**Real**) értékek és állítsuk be a sort lekérdezére (**Querying**)!

3. lépés:

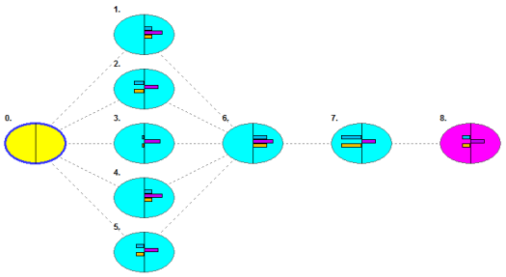
Alakítsuk ki a neurális hálózat topológiáját. Ezt a „JustNN” program erősen automatizálva végzi, csupán néhány paraméter beállítása szükséges.

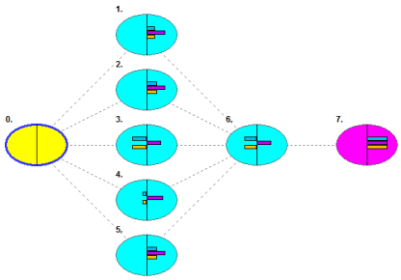


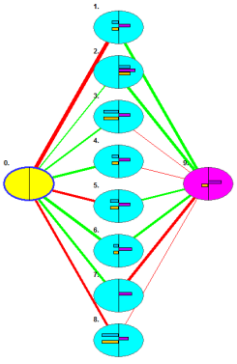
A topológi létrehozása előtt célszerű átkapcsolni hálózati nézetbe, hogy láthassuk a program által legenerált hálózati struktúrát.



Figyelem! A fenti topológiájú hálózat képes megfelelő pontossággal betanulni, de ettől egyszerűbb (például egy rétegű) hálózatok nem! Például az alábbi hálózatok nem képesek megfelelő eredményt produkálni:

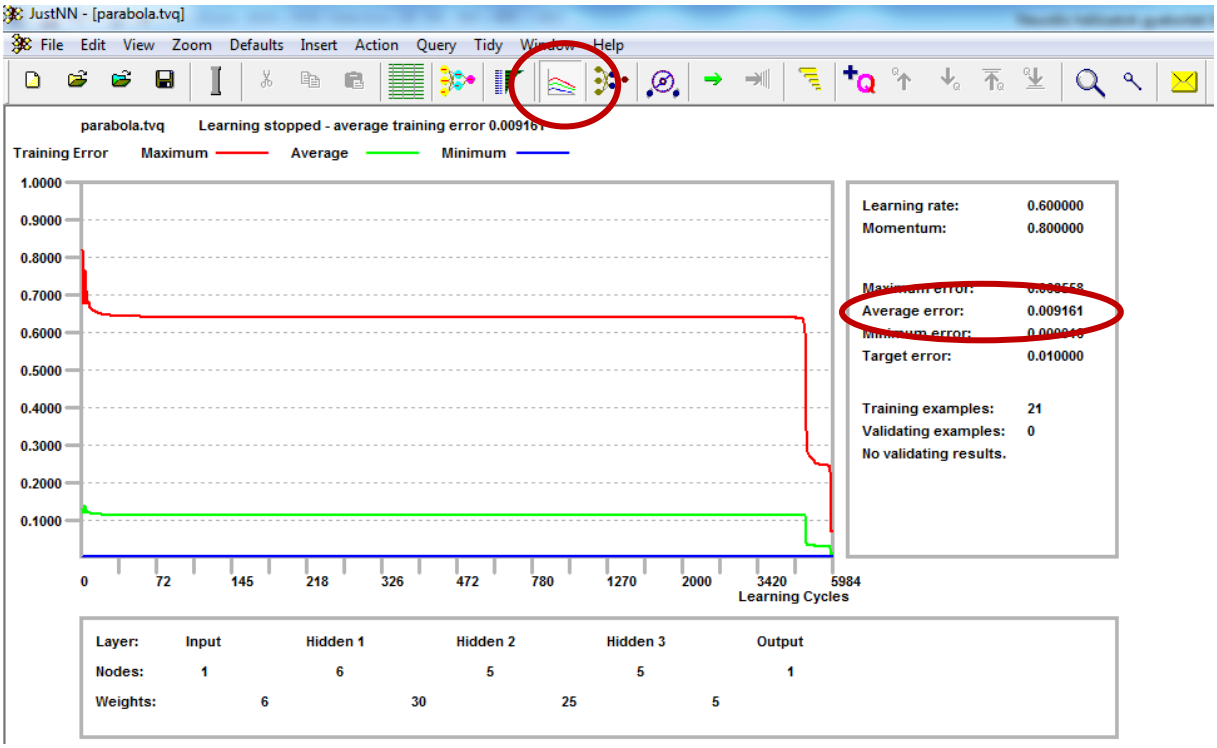






4. lépés:

A topológia generálását követően a program felajánlja a tanítás folyamatának „finomhangolását”. Itt első próbálkozás esetén hagyjunk minden beállítást a program által felkínált értéken. A tanulás hamarosan leáll. Kapcsoljunk át a hálózat átlagos hibáját és a tanulás folyamatát szemléltető nézetre.

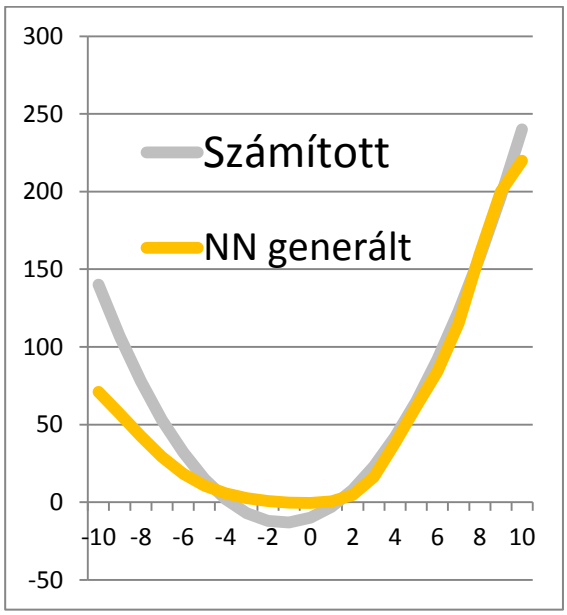


Látni fogjuk, hogy a hálózat átlagos hibája a beállított (ehhez most nem nyúltunk) 0,01 érték alatt van.

5. lépés:

Ellenőrizzük le a hálózat megfelelő működését! Ehhez a művelethez érdemes az „Excel” programot igénybe venni. Az „Excel” programban hozzunk létre egy táblázatot ahol az „X”, bemeneti értékeket konstansként vigyük be, az „Y” kimeneti értékeket pedig számíttassuk ki a feladatban megadott összefüggés szerint. Az „NN” oszlop celláit töltsük fel a „JustNN” szimulátor által a lekérdező mezőkbe megjelenő adatokkal. Ezt követően ábrázoljuk vonaldiagram segítségével a két adatsort (Y oszlop és NN oszlop) az X függvényében.

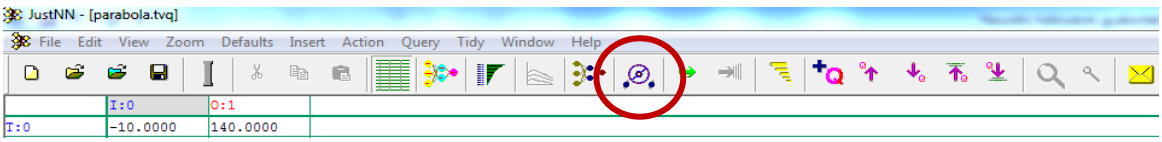
X	Y	NN (0,01)
-10	140	71,02178
-9	107	57,08103
-8	78	42,53561
-7	53	29,12988
-6	32	18,34735
-5	15	10,66586
-4	2	5,64472
-3	-7	2,523316
-2	-12	0,66028
-1	-13	-0,32438
0	-10	-0,50603
1	-3	0,544987
2	8	4,580849
3	23	16,17043
4	42	38,0741
5	65	61,79667
6	92	84,32262
7	123	115,2194
8	158	159,8856
9	197	199,9829
10	240	219,7413

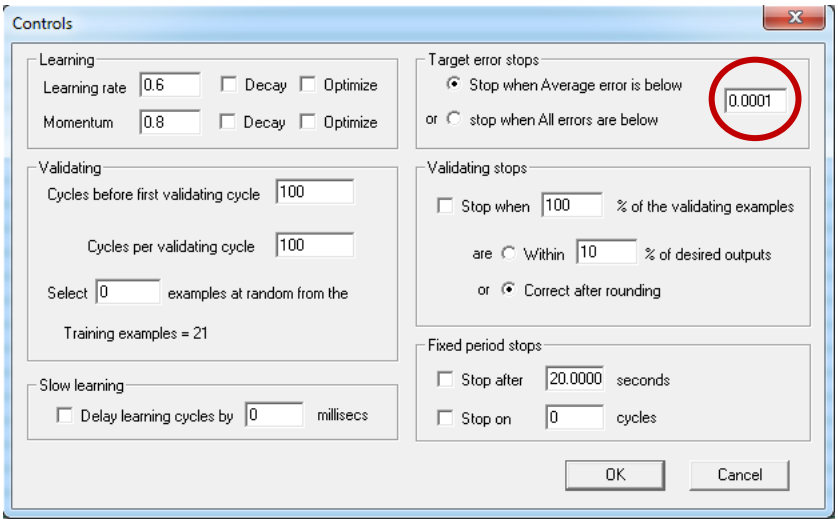


A diagramon jól látható, hogy a hálózat által generált értékek, bár jellegüket tekintve hasonló görbét alkotnak a számított függvénnyel, mégis erősen eltérnek azoktól.

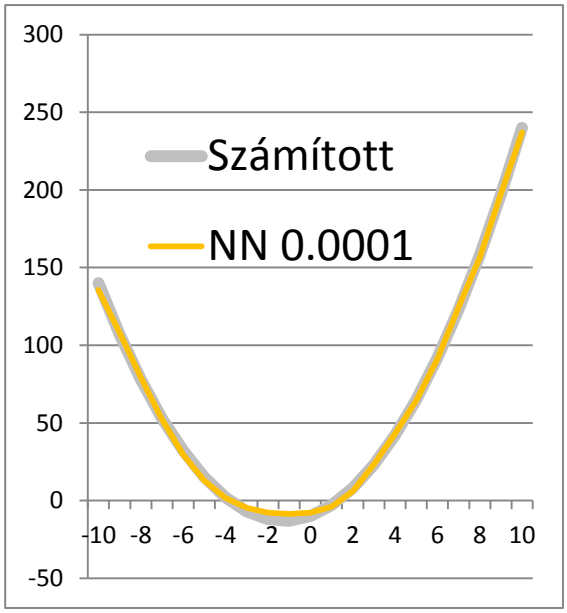
6. lépés:

A hálózat pontosságának növelése. A következő tanítás előtt állítsuk át a tanulás leállásának feltételét 0,01 átlagos hibahatárról 0,0001-re! Ezt követően újra indítsuk el a tanulást. Valószínű, hogy most a tanulási folyamat jól érzékelhetően tovább tart. A tanulás alatt érdemes megfigyelni a tanulási diagramot. Itt jól látható a hálózat pillanatnyi hibája és megfigyelhető a folyamatosan csökkenő hibaérték is.





X	Y	NN (0,0001)
-10	140	135,6581
-9	107	108,226
-8	78	79,2239
-7	53	52,15847
-6	32	29,60054
-5	15	12,72466
-4	2	1,609246
-3	-7	-4,6863
-2	-12	-7,69274
-1	-13	-8,64469
0	-10	-7,83575
1	-3	-3,98679
2	8	6,101277
3	23	23,33616
4	42	43,1454
5	65	64,39134
6	92	91,21231
7	123	124,3996
8	158	156,5914
9	197	197,69
10	240	237,0643



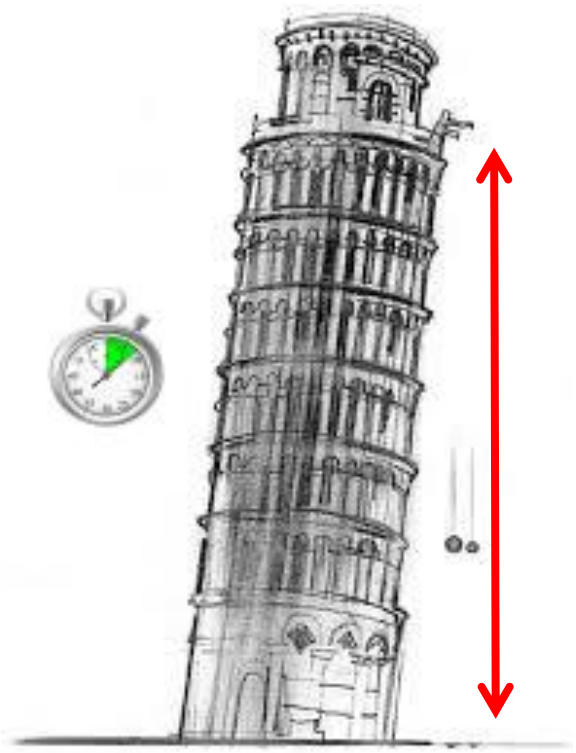
A diagramon most jól látható, hogy a hálózat igen jó közelítő értékeket szolgáltat. Érdemes kipróbálni a szimulátort olyan lekérdező értékekre is, melyek nem szerepeltek a tanító mintában. A hálózat ekkor is jó közelítést ad! Fontos megjegyezni, hogy a hálózat kizárólag a minták alapján „tanult”, azaz sehol sem lett felhasználva a másodfokú összefüggés!

3. feladat: szabadesés

Készítsünk egy olyan neurális hálózatot, amely a szabadesés törvényszerűségét képes megtanulni, természetesen a szabadesés konkrét összefüggéseinek meghatározása nélkül.

A hálózat számára készítsünk reprezentatív mintahalmazt, melyet egy kísérlet segítségével állítunk elő. A kísérlet során egy testet (mondjuk egy fém golyót) ejtsünk le rendre 1, 2,3,… méter magasról és mindegyik ejtés során mérjük meg a zuhanás idejét. Tegyük fel, hogy méréseink meglehetősen pontosak és az alábbi táblázatban foglaltuk össze:

magasság [m]	zuhanás [sec]
0	0
1	0,451524
2	0,638551
3	0,782062
4	0,903047
5	1,009638
6	1,106003
7	1,194619
8	1,277102
9	1,354571
10	1,427843
11	1,497535
12	1,564124
13	1,627992
14	1,689447
15	1,748744
16	1,806095
17	1,86168
18	1,915653
19	1,968146
20	2,019275



1. lépés:

Állítsuk elő a hálózat számára a reprezentatív tanító mintákat! A bemenetek (**Input**), azaz az egyes ejtési magasságok legyenek valós (**Real**) értékek és állítsuk be a sort tanító mintának (**Training**)!

A kimenetek (**Output**), azaz a mért zuhanási idők legyenek valós (**Real**) értékek és állítsuk be a sort tanító mintának (**Training**)!

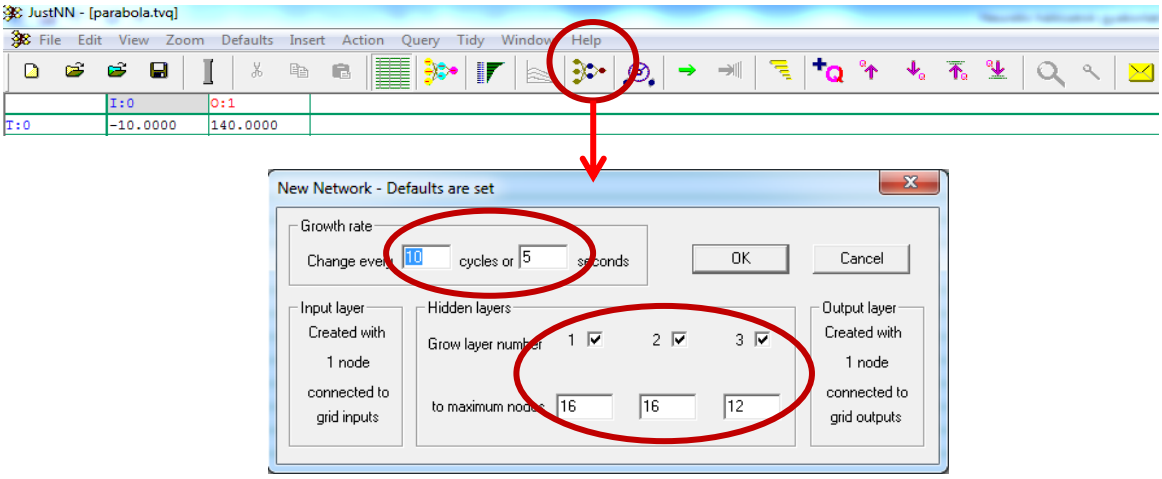
2. lépés:

Hozzuk létre az ellenőrző sorokat! A betanulását követően innen olvashatjuk ki a hálózat által adott válaszokat az egyes bemeneti értékekre (magasság). Először célszerű ugyanazokra a bemeneti értékekre vizsgálni a válaszokat, mint amiket a betanításnál alkalmaztunk.

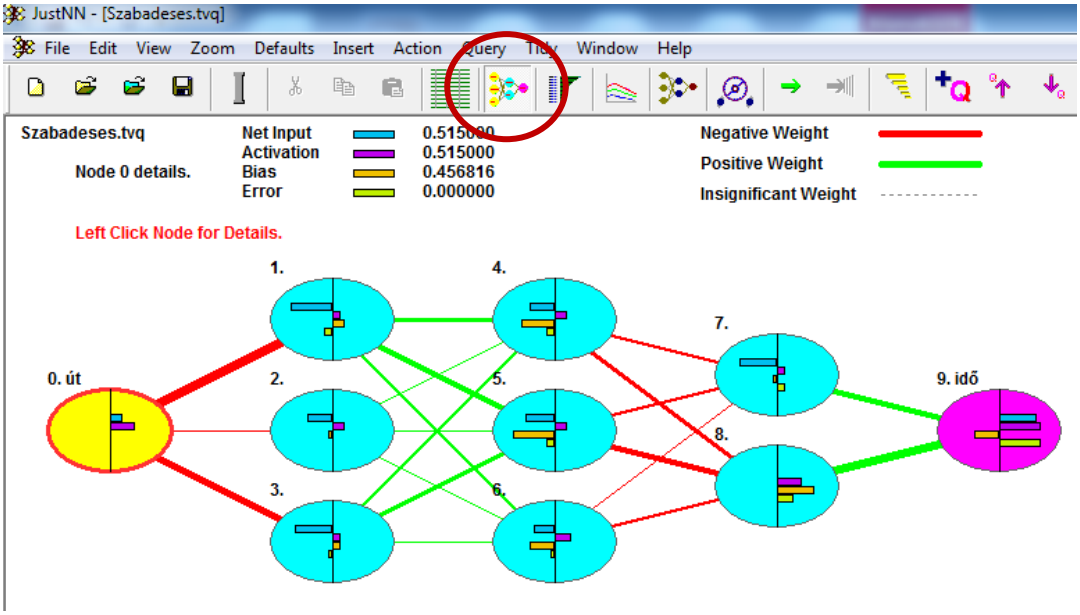
A bemenetek (**Input**) legyenek valós (**Real**) értékek és állítsuk be a sort lekérdezére (**Querying**)!

3. lépés:

Alakítsuk ki a neurális hálózat topológiáját.



A topológia létrehozása előtt célszerű átkapcsolni hálózati nézetbe, hogy láthassuk a program által legenerált hálózati struktúrát.

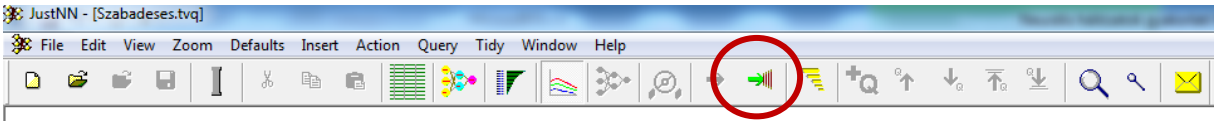


Figyelem! A fenti topológiájú hálózat képes megfelelő pontossággal betanulni, de ettől egyszerűbb (például egy rétegű) hálózatok nem! Érdemes kipróbálni több, eltérő topológiát.

4. lépés:

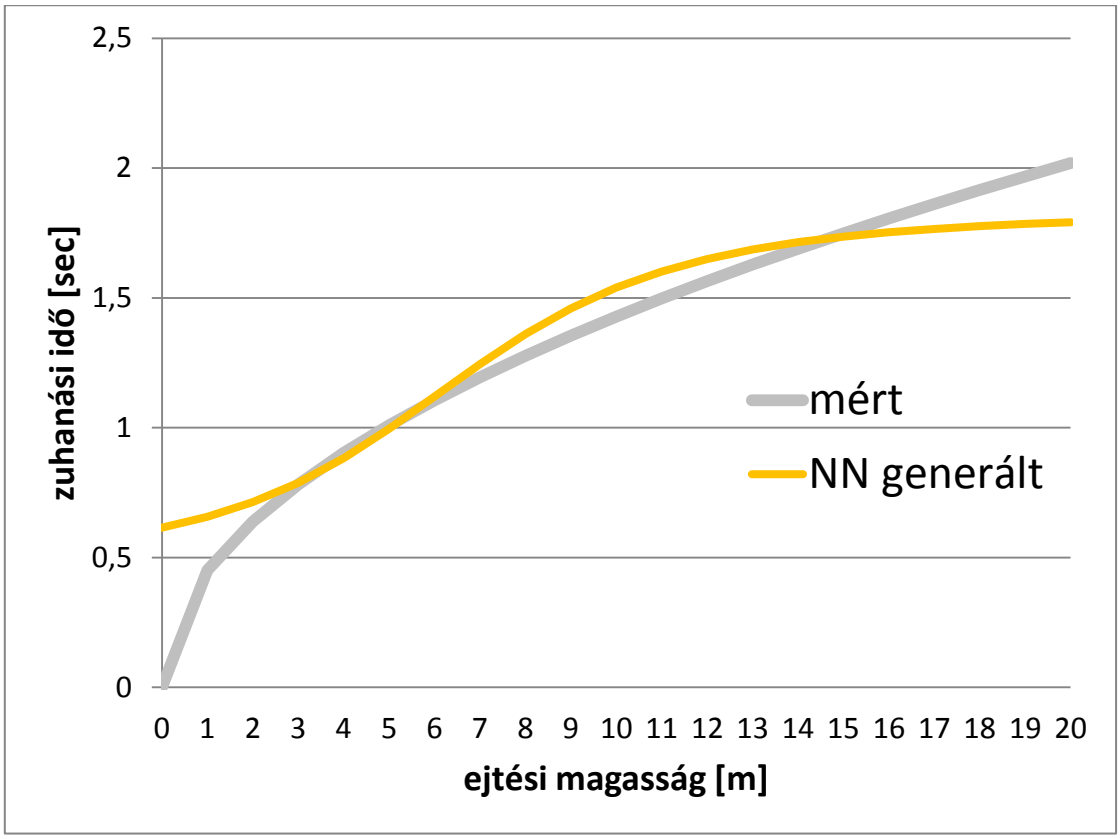
A topológia generálását követően a program felajánlja a tanítás folyamatának „finomhangolását”. Itt első próbálkozás esetén hagyjunk minden beállítást a program által felkínált értéken. A tanulás hamarosan leáll.

Figyelem! Nem megfelelő topológia (jelen esetben például egyrétegű hálózat) esetén előfordulhat, hogy a hálózat nem képes a beállított hibahatáron belüli eredményt elérni, így a tanulás nem áll le. Ebben az esetben felhasználói beavatkozással megállítható a tanulás.



5. lépés:

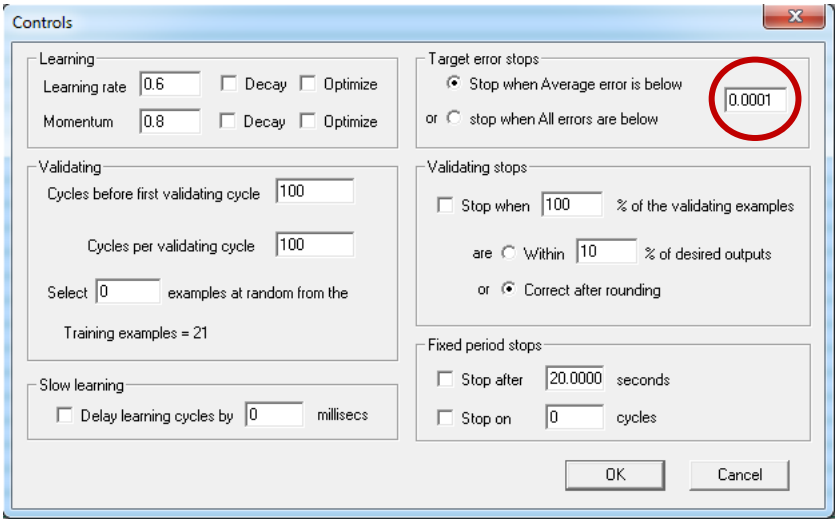
Ellenőrizzük le a hálózat megfelelő működését! Az „Excel” program segítségével ábrázoljuk a mért és a hálózat által generált zuhanási időket az ejtési magasság függvényében.



A diagramon jól látható, hogy a hálózat által generált értékek erősen eltérnek a mért értékektől. Ez azt jelenti, hogy a beállított átlagos 0,01-es hibahatár túl magas.

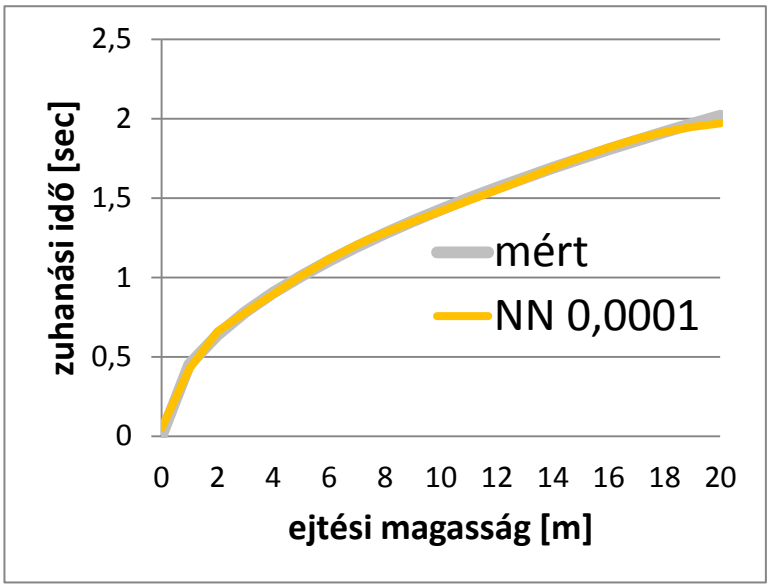
6. lépés:

Állítsuk a hibahatár értékét 0,0001-re!



Ezt követően tanítsuk újra a hálózatot. Most a tanulás érezhetően hosszabb ideig tart. A tanulás leállása után ismét ellenőrizzük a hálózat jóságát a már ismert módon az „Excel” program segítségével.

Jól látható, hogy az alacsonyabb hibahatár elérésével a hálózat igen jó közelítéssel adja meg a kísérletek során mért időértékeket. Természetesen a hálózat olyan esetekben is jó eredményeket szolgáltat, amely esetek a tanítómintában nem szerepeltek. Valójában a hálózat az alábbi, jól ismert összefüggést közelíti:

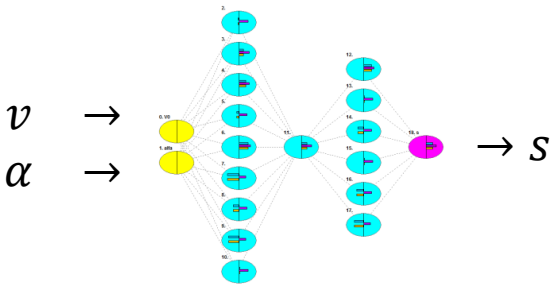


$$t = \sqrt{\frac{2 * s}{g}}$$

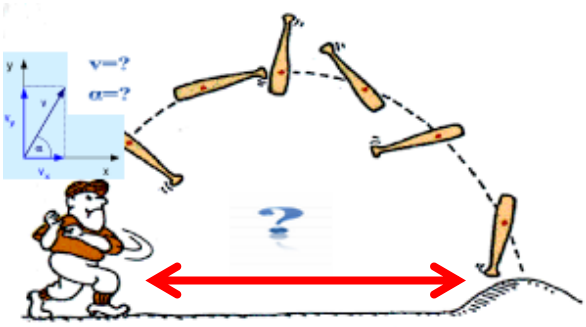
Természetesen most is igaz az, hogy a hálózat a fenti összefüggést nem „ismeri”. A válaszokat csupán a tanítási minták alapján generálta.

4. feladat: ferdehajítás

Készítsünk egy olyan neurális hálózatot, amely a ferde hajítás törvényszerűségét képes megtanulni. Ebben a példában kétváltozós problémára keressük a választ.



Egy olyan hálózatot szeretnénk kialakítani, amely megmondja, hogy tetszőleges kezdősebességgel (v) és meredekséggel (α) eldobott test tőlünk milyen messze esik le.



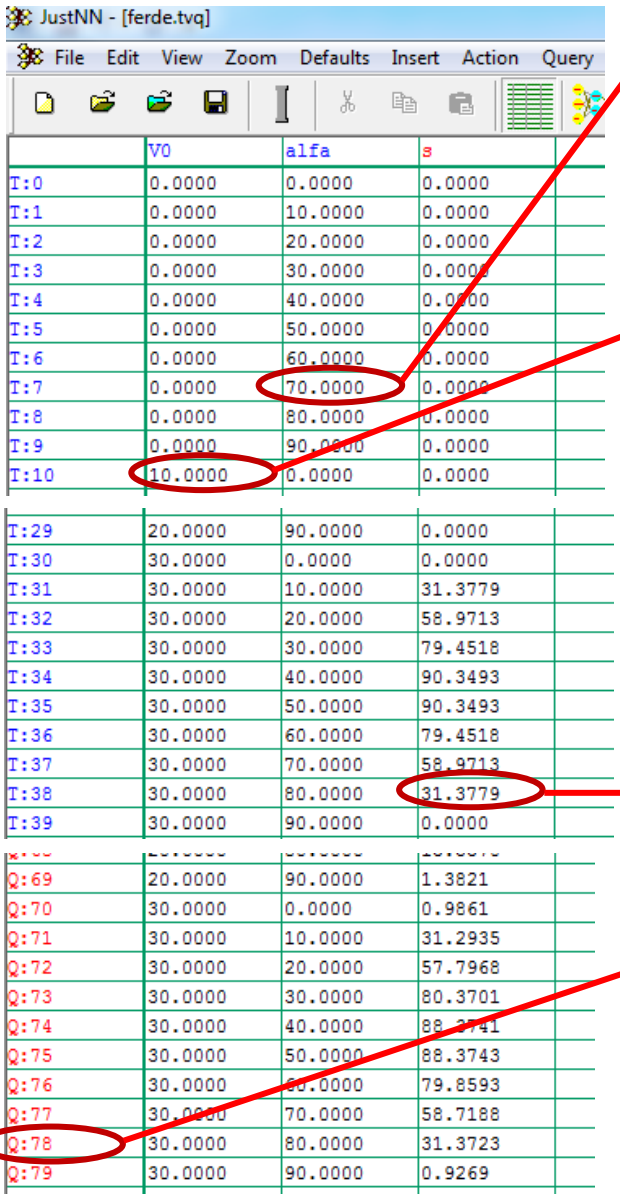
Természetesen az amúgy ismert összefüggés használata nélkül ez úgy lehetséges, hogy kísérleteket kell végezni és az így kapott eredményeket kell a hálózat számára tanítómintaként felhasználni. A kísérleteket szisztematikusan 0, 10, 20 és 30 m/s kezdősebességgel, különböző szögben (0^0-90^0-ig) eldobott tárggyal végezzük. Tegyük fel, hogy méréseink meglehetősen pontosak és a jobb oldali táblázatban foglaltuk össze:

α	v	s
0	0	0
10	0	0
20	0	0
30	0	0
40	0	0
50	0	0
60	0	0
70	0	0
80	0	0
90	0	0
0	10	0
10	10	3,486443867
20	10	6,552371149
30	10	8,827985767
40	10	10,03881502
50	10	10,03881502
60	10	8,827985767
70	10	6,552371149
80	10	3,486443867
90	10	0
0	20	0
10	20	13,94577547
20	20	26,20948459
30	20	35,31194307
40	20	40,15526006
50	20	40,15526006
60	20	35,31194307
70	20	26,20948459
80	20	13,94577547
90	20	0
0	30	0
10	30	31,3779948
20	30	58,97134034
30	30	79,45187191
40	30	90,34933514
50	30	90,34933514
60	30	79,45187191
70	30	58,97134034
80	30	31,3779948
90	30	0

1. lépés:

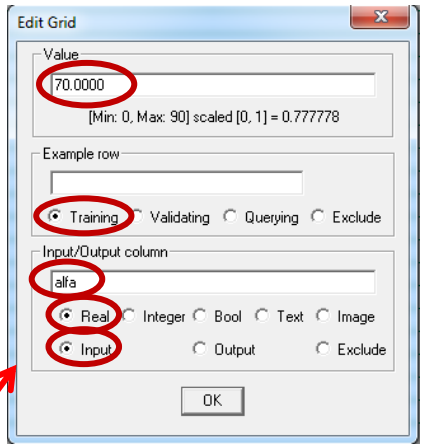
Állítsuk elő a hálózat számára a reprezentatív tanító mintákat! Ne felejtsük el, hogy ebben az esetben két bemeneti mezőnk (v és α) és egy kimeneti mezőnk (s) lesz. A bemenetek (**Input**), azaz a hajítás kezdő sebessége (v) és a hajítás szöge (α) legyenek valós (**Real**) értékek és állítsuk be a sort tanító mintának (**Training**)!

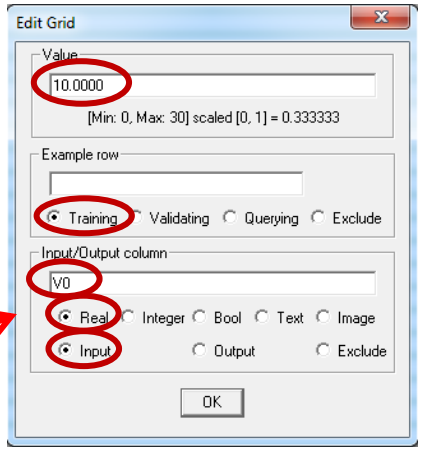
A kimenetek (**Output**), azaz a mért távolságok legyenek valós (**Real**) értékek és állítsuk be a sort tanító mintának (**Training**)!

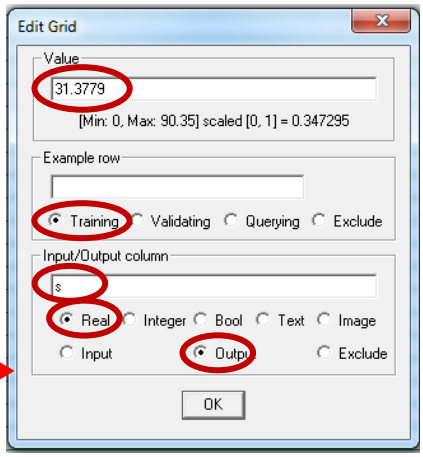


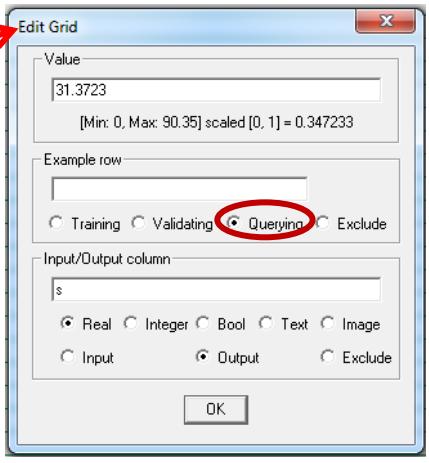
	V0	alfa	s
T:0	0.0000	0.0000	0.0000
T:1	0.0000	10.0000	0.0000
T:2	0.0000	20.0000	0.0000
T:3	0.0000	30.0000	0.0000
T:4	0.0000	40.0000	0.0000
T:5	0.0000	50.0000	0.0000
T:6	0.0000	60.0000	0.0000
T:7	0.0000	70.0000	0.0000
T:8	0.0000	80.0000	0.0000
T:9	0.0000	90.0000	0.0000
T:10	10.0000	0.0000	0.0000
T:29	20.0000	90.0000	0.0000
T:30	30.0000	0.0000	0.0000
T:31	30.0000	10.0000	31.3779
T:32	30.0000	20.0000	58.9713
T:33	30.0000	30.0000	79.4518
T:34	30.0000	40.0000	90.3493
T:35	30.0000	50.0000	90.3493
T:36	30.0000	60.0000	79.4518
T:37	30.0000	70.0000	58.9713
T:38	30.0000	80.0000	31.3779
T:39	30.0000	90.0000	0.0000
Q:69	20.0000	90.0000	1.3821
Q:70	30.0000	0.0000	0.9861
Q:71	30.0000	10.0000	31.2935
Q:72	30.0000	20.0000	57.7968
Q:73	30.0000	30.0000	80.3701
Q:74	30.0000	40.0000	88.3741
Q:75	30.0000	50.0000	88.3743
Q:76	30.0000	60.0000	79.8593
Q:77	30.0000	70.0000	58.7188
Q:78	30.0000	80.0000	31.3723
Q:79	30.0000	90.0000	0.9269

0









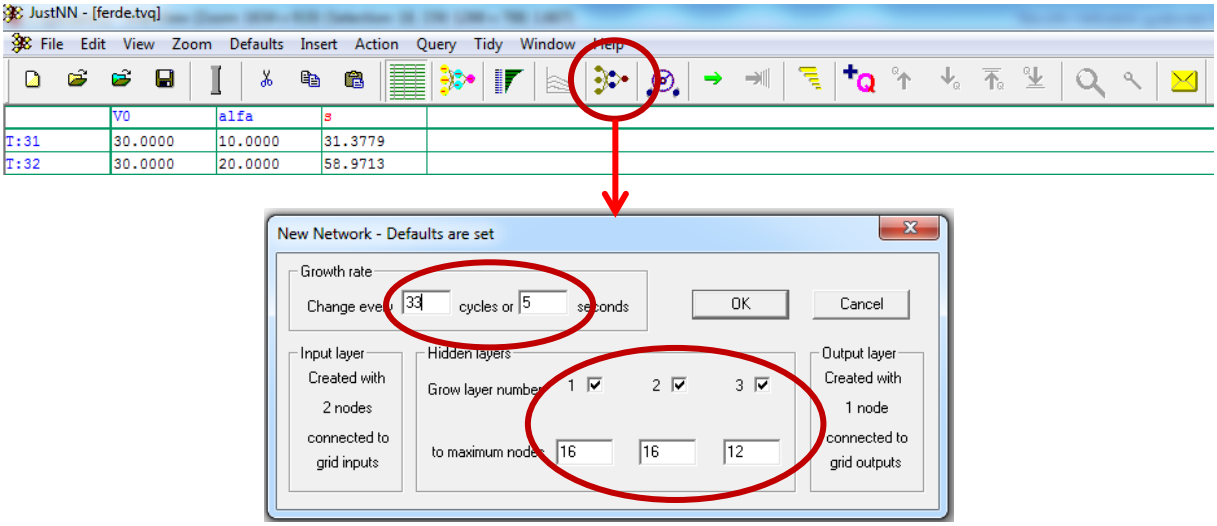
2. lépés:

Az eddigiekhez hasonló módon hozzuk létre az ellenőrző sorokat! A bemenetek (**Input**) legyenek valós (**Real**) értékek és állítsuk be a sort lekérdezésre (**Querying**) (fenti ábra)!

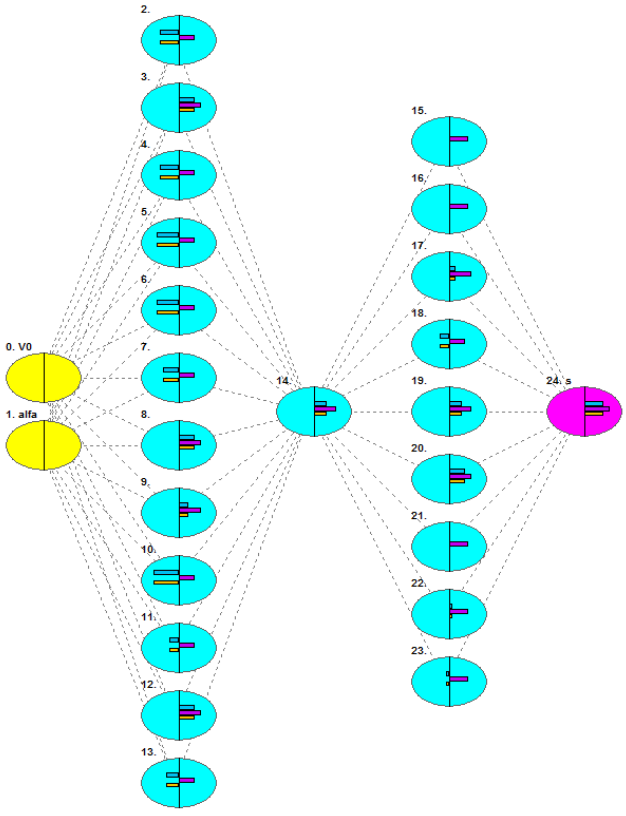
A betanulását követően innen olvashatjuk ki a hálózat által adott válaszokat az egyes bemeneti értékekre (v és α). Először célszerű ugyanazokra a bemeneti értékekre vizsgálni a válaszokat, mint amiket a betanításnál alkalmaztunk.

3. lépés:

Alakítsuk ki a neurális hálózat topológiáját.



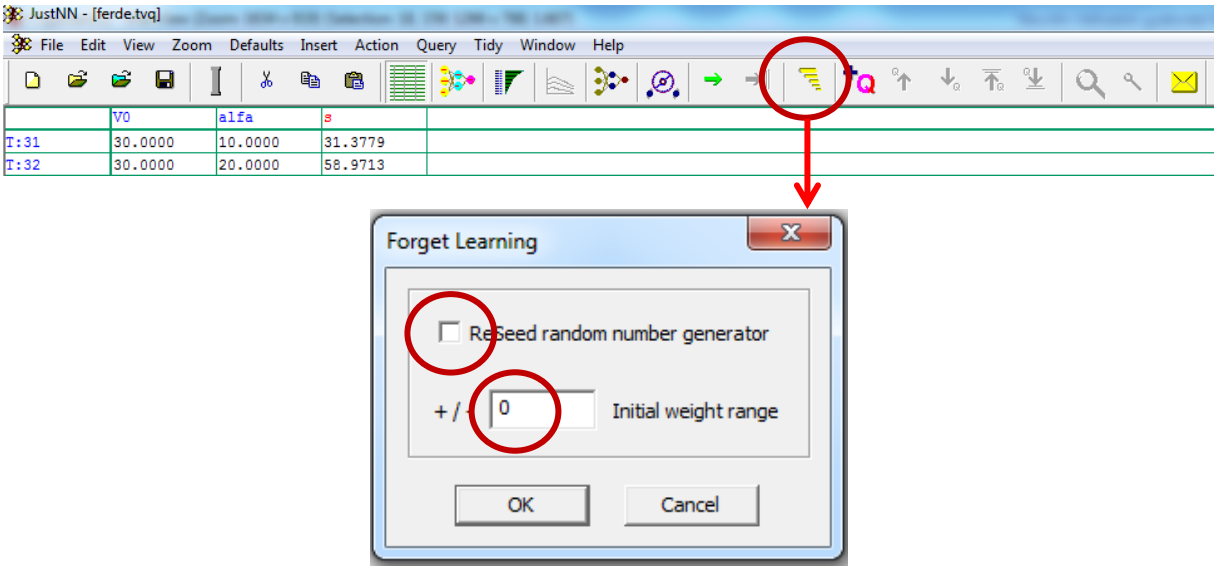
A topológia létrehozása előtt célszerű átkapcsolni hálózati nézetbe, hogy láthassuk a program által legenerált hálózati struktúrát.



Figyelem! A feladat viszonylagos komplexitása miatt célszerű a példában szereplő beállításokkal kezdeni a szimulációt. Ezekkel a beállításokkal a hálózat képes megfelelő hibahatáron belül működni.

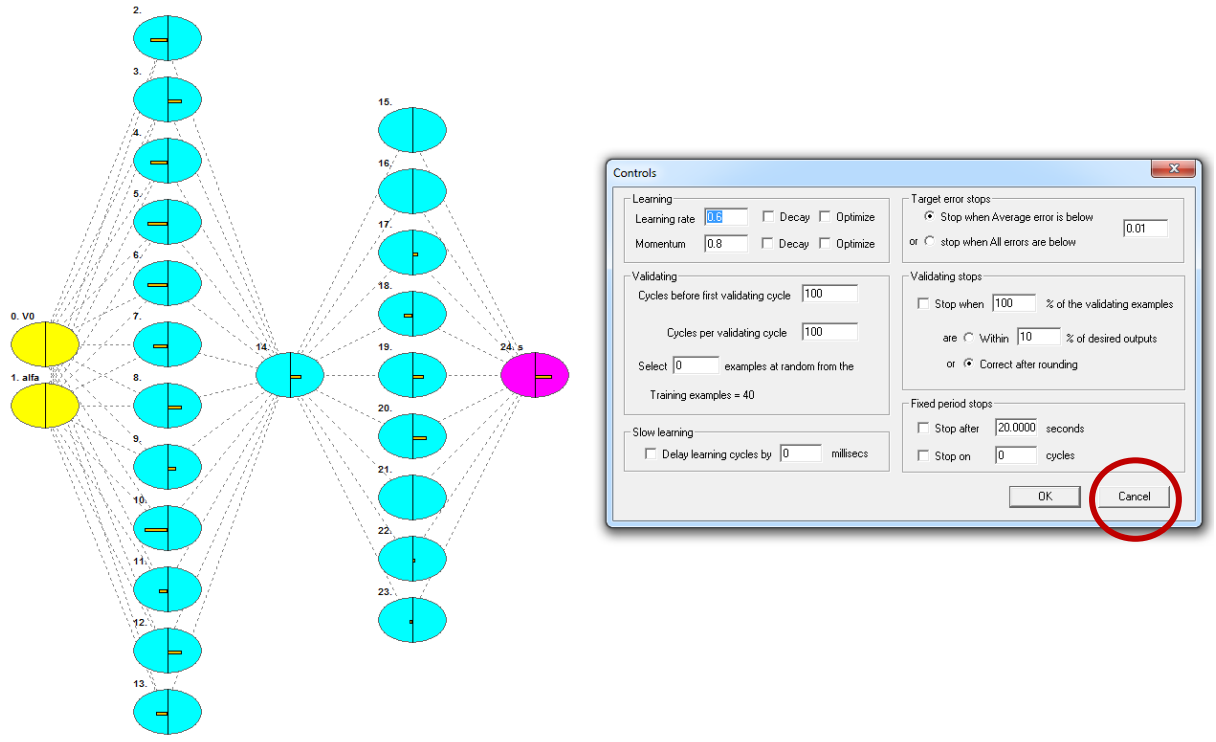
Előfordulhat, hogy a fenti beállítások ellenére sem a jobb oldalon látható topológiát generálja a szimulátor. Ekkor a kezdeti súlytényezőket ki kell nullázni és azt követően újra legeneráltatni a fenti beállításokkal a hálózatot.

A súlytényezők nullázása az alábbi módon történik:

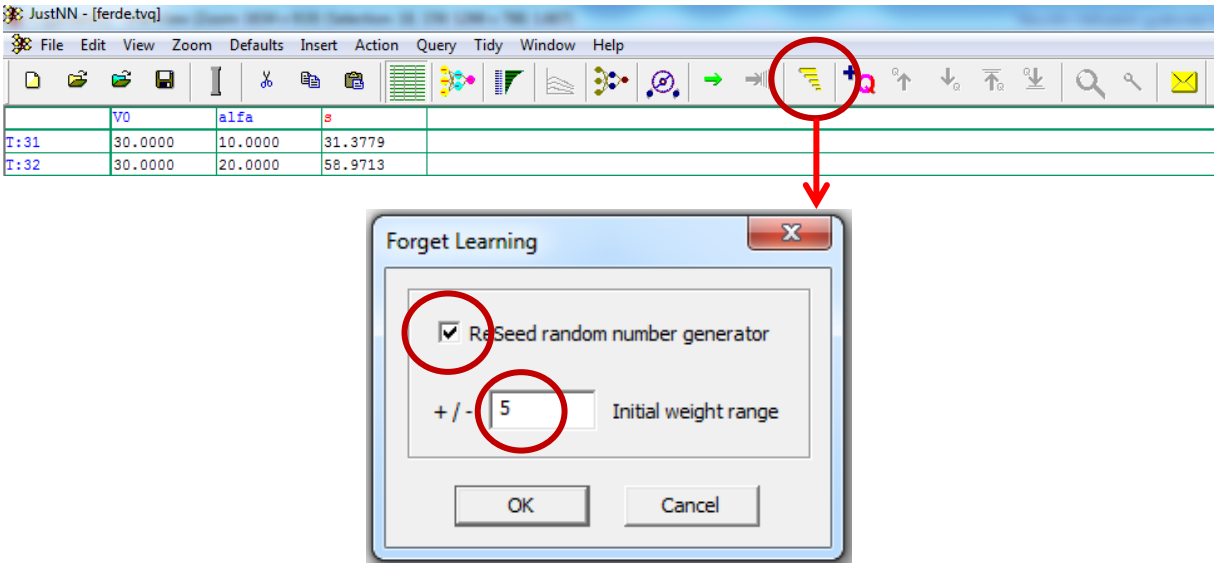


Ekkor azt fogjuk tapasztalni, hogy a hálózat egyes csomópontjainak (neuronjainak) összeköttetése szaggatott vonalakká változik.

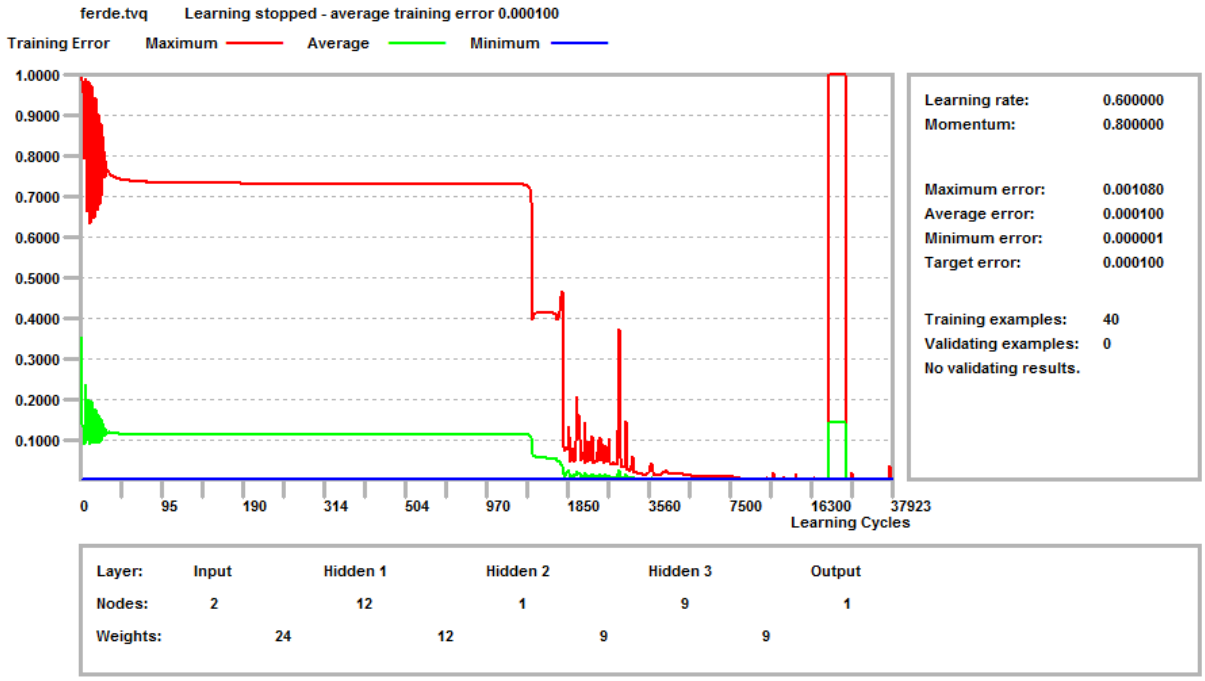
Amikor a „JustNN” program létrehozta a hálózatot, ne lépjünk tovább a tanulás felé, hanem lépjünk ki a folyamatból.



A megfelelő topológiájú hálózat esetén a kezdeti súlytényezőket állítsuk be az alábbi módon:



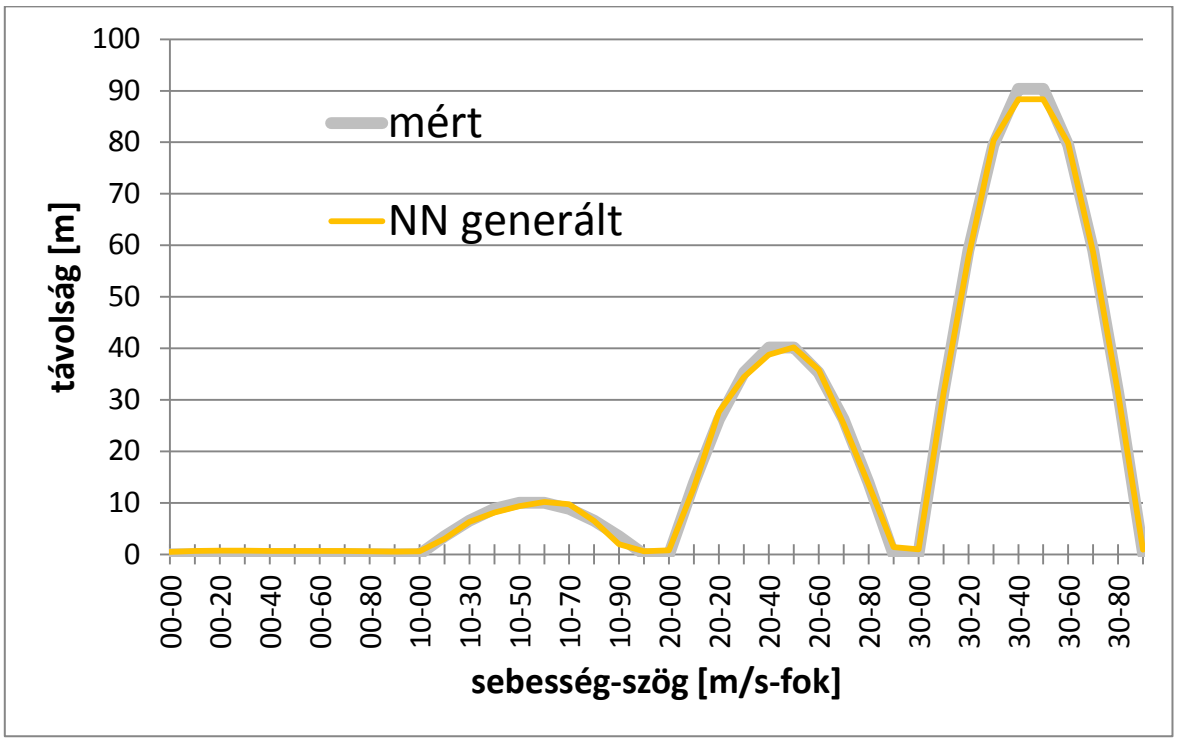
Ezt követően állítsuk be a tanulás leállási feltételénél a hibahatárt 0,01-ről 0,0001-re. Kapcsoljunk át a tanulás folyamatát mutató nézetre és indítsuk el a tanulást. A diagramon jól nyomon követhetjük a hálózat hibájának alakulását. A tanulás most az előző szimulációkhoz képest érezhetően hosszabb ideig fog tartani.



A betanulás folyamata a fenti ábrától eltérő lehet, mivel a hálózat kezdeti súlyi véletlenszerűen lettek feltöltve. Ennek megfelelően előfordulhat, hogy a hálózat rövidebb idő alatt tanul, de előfordulhat az is, hogy nem tanul. Ez esetben célszerű a kezdeti súlyokat újra generáltatni az előzőekben ismertetett módon, majd újra megpróbálkozni a tanítással.

4. lépés:

A betanult hálózat eredményét a már jól ismert módon „Excel” program segítségével ellenőrizhetjük.



A hálózat meglepő pontossággal képes az amúgy jól ismert ferdehajítás összefüggésének megfelelő választ adni. Természetesen most is igaz az, hogy az alábbi összefüggést a hálózat nem „ismeri”, az csupán a mérési eredmények alapján generálja a válaszokat.

$$s = \frac{v^2 sin2\alpha}{g}$$

Érdemes kipróbálni a hálózatot olyan esetekre is, melyek a tanítómintában nem szerepeltek. Természetesen a hálózat akkor ad elfogadható eredményt, ha a bemenő adatok a tanítómintához közeli tartományúak.