4. mérés

Neurális és fuzzy rendszerek vizsgálata

Sujbert László, Gyurity Igor
BME Méréstechnika és Információs Rendszerek Tanszék

1. A mérés célja

A mérés célja a korábbi szakirányú tanulmányok során megszerzett, az intelligens rendszerek témakörébe tartozó ismeretek elmélyítése, azok néhány gyakorlati vonatkozásának megismerése. A mérési gyakorlat során egy-egy egyszerű neurális és fuzzy rendszert kell megalkotni. A neurális hálózat feladata egy egyszerű osztályozási probléma megoldása: üvegpoharak megütése után hang alapján a rendszernek el kell döntenie, hogy a pohár ép vagy sem. A fuzzy rendszer egy jól előkészített zenei hangfelvétel kottázását kell elvégezze. A fenti feladatok intenzív előfeldolgozást igényelnek, ezért a digitális jelfeldolgozás alapjainak (mintavételezés, diszkrét Fourier-transzformáció, digitális szűrés stb.) készségszintű ismeretét feltételezzük.

A megoldandó problémák a jelfeldolgozás témakörébe vágnak, hiszen hangfelvételeket, azaz akusztikus információt hordozó egydimenziós jeleket kell feldolgozni. A neurális hálózatokat vagy a fuzzy rendszereket mégsem szokás a jelfeldolgozás eszközeinek tekinteni, gyakran az intelligens rendszerek közé soroljuk őket. Ennek oka, hogy ezek a rendszerek az adatok magas szintű feldolgozását végzik, az általuk végzett műveletek gyakran nem írhatók le a klasszikus módon, pl. egyenletekkel. A klasszikus jelfeldolgozási módszereket az intelligens rendszerek is igénylik, jellegzetesen előfeldolgozásnak, vagy alacsony szintű feldolgozásnak nevezve őket. A „magas", illetve az „alacsony" szint között a határvonal nem éles, a határterületen az adaptív jelfeldolgozó eljárások helyezkednek el. A neurális és a fuzzy rendszerek közös jellemzője, hogy a hálózat megalkotásához szükséges információt csak hiányosan, illetve kvalitatív formában bocsátjuk rendelkezésre. Az alábbiakban az egyszerűség kedvéért a kétféle rendszert külön tárgyaljuk.

2. Osztályozás neurális hálózattal

2.1. Az osztályozás alapgondolata

Neurális hálózatok témakörben egy egyszerű, gyártósori ellenőrzési feladatot kell elvégezni: egy pohárról el kell dönteni, hogy jó, vagy rossz (repedéseket tartalmaz). A pohár hangját rögzíteni kell mindkét esetben (a rossz pohár szimulálása kitöltőanyaggal történik), majd egy előre meghatározott topológiájú és architektúrájú neurális hálózatot kell megtanítani a jó poharak felismerésére, illetve a rossz poharak kiszűrésére.

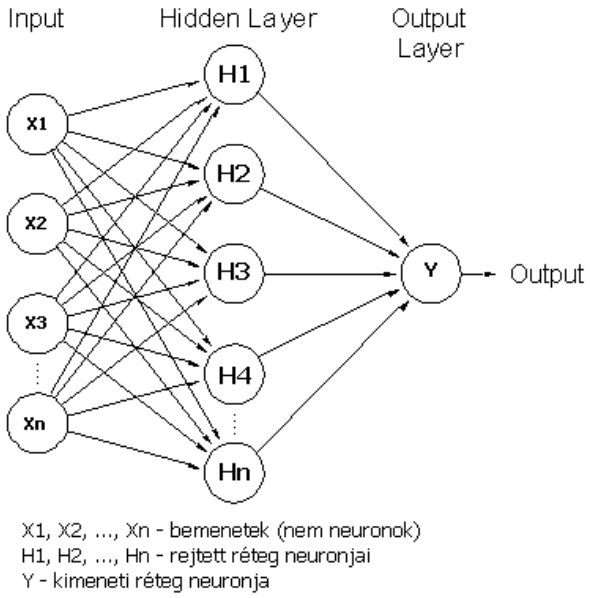
A különféle üvegből, porcelánból, kerámiából stb. készült tárgyak repedésmentességének ellenőrzésére régi módszer az, hogy a tárgyat finoman megütik, és ha szépen csengő hangot ad, repedésmentes; ellenben, ha törés vagy repedés van rajta, akkor nincs csnegő hang (v.ö. repedtfazékhang). A hang minőségének jellemzésére alkalmas a spektrum, amely nemcsak jól jellemző a zenei hangokra, hanem adatredukcióra is alkalmas, mert egy esetleg több tízezer mintás felvételt jellemez pár száz, esetleg ezer adattal.

A neurális hálózat tehát a kiválasztott hangminta spektrumát kapja meg. A spektrum összetevői alapján azonban már nehéz lenne egyértelmű szabályrendszert felállítani, hogy milyen frekvenciájú és amplitúdójú spektrum jellemző egy adott pohár esetében a jó pohárra, és milyen a rosszra. Itt használható ki a neurális hálózat tanulóképessége. A tanulás során elegendően nagy számú jó és rossz poharat „bemutatunk" a hálózatnak. Ha a tanulás és a tanító minták kiválasztása megfelelő volt, a korábban nem tanított mintákat is helyesen ismeri fel a hálózat.

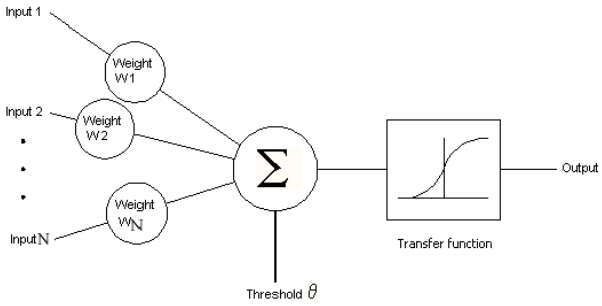
A méréshez a MATLAB Neural Network Toolboxát használjuk.

2.2. Az alkalmazott neurális hálózat

A tanítandó neurális hálózat feed-forward topológiájú backpropagation algoritmusú (előrecsatolt hiba-visszaterjesztéses), kétrétegű (rejtett réteg + kimeneti réteg) és egy kimenetű, ahogyan az 1. ábrán látható. A neuronok egy összegzőből és egy aktivációs függvényből állnak. Az összegző valamennyi bemenet súlyozott összegét állítja elő, míg az aktivációs függvény – többnyire – szigmoid átvitelű (lásd a 2. ábrát).



1. ábra. A tanítandó neurális hálózat struktúrája



2. ábra. Egy neuron struktúrája

Tanulás alatt olyan folyamatot értünk, amely során a hálózat a súlyait (esetleg a felépítését) úgy változtatjuk meg, hogy egy adott bemeneti mintához megfelelő kimeneti mintát szolgáltasson. A tanulásnak két fő típusa van: felügyelt és nem felügyelt tanulás. A mérés során csak felügyelt (ellenőrzött) tanulással fogunk foglalkozni. A felügyelt tanulásnál ismertek a kívánt kimenetek (targetek). A felügyelés egy tanítóval történik, amely a célkimenet és az aktuális kimenet különbségét (hibajelet) – valamilyen algoritmus alapján – minimalizálja. Az egyik leggyakrabban alkalmazott eljárás a Backpropagation algoritmus. Legegyszerűbb esetben a $k+1$-edik súlyvektorokat a következőképpen határozzuk meg:

$$w_{k+1} = w_k + \alpha_k g_k \tag{1}$$

ahol w_k a jelenlegi súlyvektor, α_k a tanulási ráta és g_k a jelenlegi gradiens. A tanítási algoritmusokat – a gradiens számítása alapján – két fő kategóriába lehet sorolni: legmeredekebb és változó irányú módszerek. Néhány módszert és fontosabb jelemzőiket az alábbiakban tekintjük át:

- **Gradient descent backpropagation:** a legmeredekebb lejtő (negatív gradiens) irányában változtatja meg a súlyokat, fix tanulási ráta mellett. Egyik hátránya, hogy könnyen beleragadhat egy lokális minimumba. Ez kiküszöbölhető a momentumos gradiens módszerrel (Gradient descent with momentum). Másik hátránya, hogy a fix tanulási ráta miatt az algoritmus oszcillálhat, vagy nagyon lassan konvergálhat

a minimális értekhez. Ez kiküszöbölhető adaptív tanulási ráta gradiens módszerrel (Gradient descent with adaptive learning rate). A két módszer előnyeit ötvözi a Gradient descent with momentum and adaptive learning rule backpropagation algoritmus.

- **Resilient backpropagation:** a legmeredekebb lejtő gondot okozhat a szigmoid átvitelnél, mivel a gradiens nagyon kicsi megváltozásának hatására a súlyoknál is kicsi lesz a változás (annak ellenére, hogy messze lehetünk az optimális értéktől). Ez az algoritmus kiküszöböli ezt a problémát: a súlyváltozás irányát a gradiens előjele adja meg, a változás mértékére a gradiens nincs hatással (ezt egy külön változó határozza meg).
- **Conjugate gradient backpropagation:** a keresés konjugált irányokban történik, hogy meghatározzuk azt a tanulási rátát, amelynél minimális a hiba. A konjugált irányok miatt a lépések száma kedvező (zajmentes) esetben megegyezik a paramétertér dimenziójával, a digitális szűrőknél megismert „FIR-szűrő" megfelelője. Ez a módszer tehát gyorsabb konvergenciát eredményez, viszont a konjugált irányok számítása idő- és számításigényesebb, mint a fenti módszerek.
- **Quasi-Newton backpropagation:** a konjugált gradiens módszer alternatívája, amely gyorsabb optimalizálást biztosít, viszont több számítást és memóriát igényel. Kis neuronszámú hálózatokra hatékony.
- **One step secant backpropagation:** a konjugált gradiens és a kvázi-Newton-módszer közti kompromisszum eredménye. Kevesebb számítást és memóriát igényel, mint a kvázi-Newton-módszer, és gyorsabb keresést eredményez, mint a konjugált gradiens módszer.
- **Levenberg-Marquardt backpropagation:** a Newton gradiens módszer alternatívája. Sok memóriát igényel, de a tanítási algoritmusok közül a leggyorsabb (még néhány száz neuron esetén is). Ezt a módszert ajánlott használni.

3. Megvalósítás MATLAB-ban

3.1. Felvételek rögzítése

A méréshez rendelkezésre áll négy, azonos gyártósorról származó borospohár, valamint háromfajta, különböző színű és méretű üvegszemcsék. A felvételek rögzítéséhez mikrofon, hangkártya és rögzítőszoftver (Audacity) áll rendelkezésre. Az Audacity beállításai a következők (Edit → Preferences):

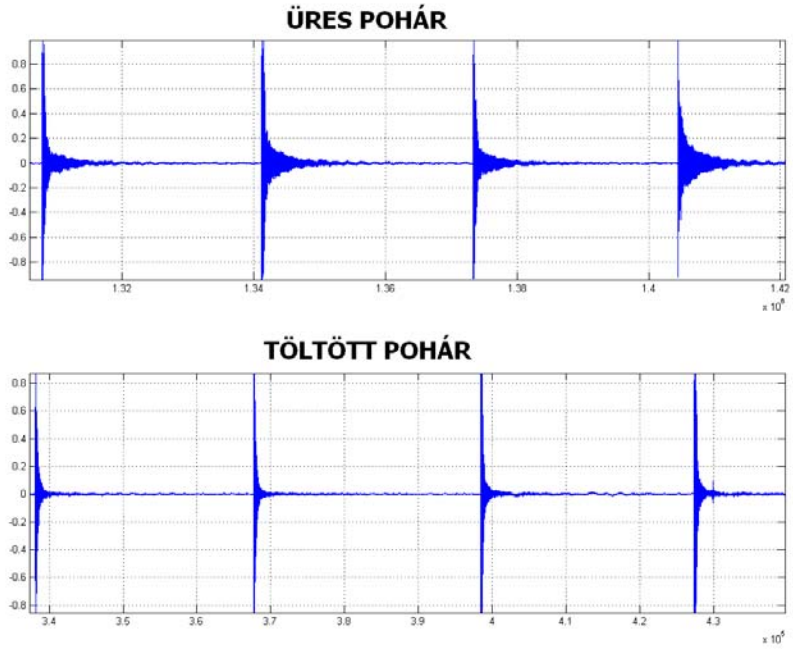
- Audio I/O channels: 1 (mono);
- Quality: 44100 Hz, 16-bit;
- File Formats: WAV (16 bit PCM).

Poharak ütögetése: az adott (üres vagy megtöltött) poharat a külső felületére merőleges síkban, a pohár szájához közel kell ütögetni. Tartózkodjunk a túlzott erő kifejtésétől, illetve gondoskodjunk a pohártalp megfelelő rögzítéséről. A repedéseket tartalmazó pohár szimulálásához kitöltőanyagot használunk. A poharat nem célszerű teljesen megtölteni, csak annyira ahol már szignifikánsan észlelhető a tompítás.

3.2. Minták kivágása

A pohár hangjáról készített felvételeket a wavread paranccsal tölthetjük be a MATLAB-ba.

```
felvetel = wavread('pohar.wav');
```



3. ábra. Megütött poharak hangjának felvételei

A 3. ábrán az egyik pohárról készített felvételsorozat látható. A felső ábrán az üres pohár, az alsó ábrán a töltött pohár megütéseiről készített felvétel látható.

Jól látható, hogy az üres pohár lecsengése hosszabb, így abból könnyebb jó mintákat kivágni. A töltött pohárnál a lecsengés nagyon gyors, így ott óvatosan kell eljárni a minták kivágásánál. Általában egy kivágott minta akkor tekinthető jónak, ha már nem tartalmaz jelentős megütési tranziens összetevőket, illetve a zaj még nem dominál a jelben (még nem került a zaj szintjére a lecsengés). A minták kivágásához a `kivag` függvény használható:

```
function     [minta,db]=kivag(felvetel, hossz, szint, eltolas);
             %bemeneti paraméterek:
             %felvetel:  ebből vágjuk ki a mintákat
             %hossz:  a kivágandó minták hossza
             %szint:  e szint felett megütési tranziensnek tekintjük az összetevőt
             %eltolas:  késleltetés az utolsó tranziens-összetevő és az első kivágandó minta között

             %kimeneti változók:
             %db:  ennyi mintát vágunk ki a felvételből
             %minta:  sorvektor, amely a kivágott mintákat tartalmazza (mérete:  db*hossz)
```

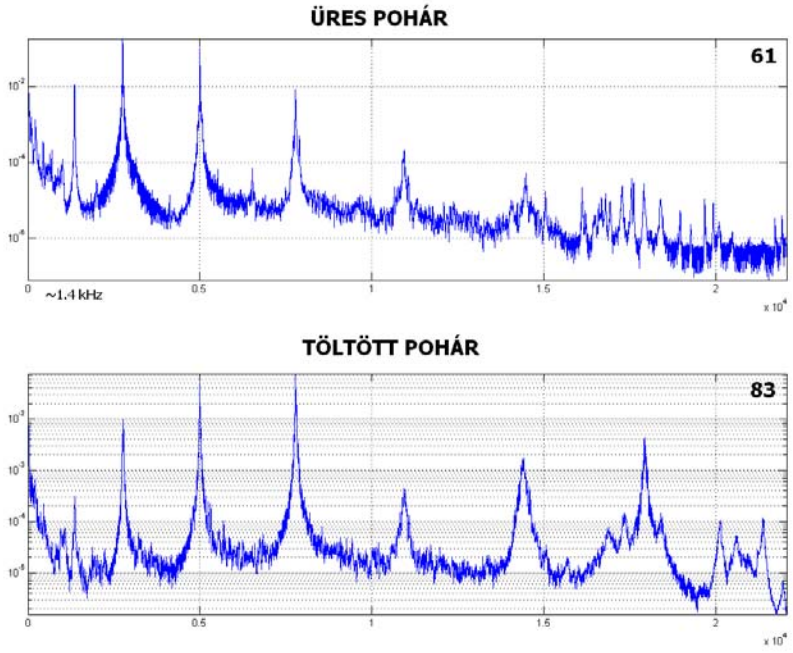
A kivágás lényege: a felvétel alapján meghatározunk egy szintet, amely felett még tranziensnek tekintjük a jelet. Az adott megütésnél, az utolsó ilyen tranziens minta utáni eltolás-adik mintát tekintjük az első kivágandó mintának. A függvény kiszűri az olyan mintákat, amelyeknél – túl nagy hossz vagy eltolás miatt – a kivágandó minta belelóg a következő megütésbe.

3.3. Jelfeldolgozás

A kivágott mintákból spektrumot kell számolni, amely a neurális hálózat bemenetei lesznek. A spektrumokat célszerű a MATLAB beépített `psd` függvényével számítani:

```
spektrum = psd(minta,N,Fs,window(N));
```

A 4. ábrán látható az egyik pohárra számolt spektrum, az üres és a töltött esetben. A teljes regisztrátumról készült 1–1 db, a domináns komponensre normalizált, spektrum. A jobb felső sarokban látható az egyes esetekben átlagolt minták száma. A fenti spektrumok segítségével eldönthetjük, mely frekvenciatartományt, illetve



4. ábra. Megütött poharak hangjának spektrumai

milyen amplitúdószintet hagyjunk meg a jelből, és az optimális DFT-pontszámot is meg tudjuk határozni. A neurális hálózat bemenete egy adott megütésről készített felvétel spektruma. Célszerű a méréshez előre megírni egy for-ciklust, amely a kivágott minták tömbjéből (sorvektorból) külön-külön kiszámolja minden egyes megütés spektrumát. Az optimális pontszám fontos, mert a neurális hálózat csak akkor hatékony (időben és memóriával), ha kevés hasznos (számításigényes) bemeneti mintával terheljük. A pontszám csökkentésnek a spektrumból való információvesztés szab határt. Ha túl kicsi pontszámot választunk, akkor a számunkra fontos harmonikus komponensek eltűnhetnek a spektrumból. Magasabb pontszámnál célszerű az irreleváns mintákat kiszűrni a spektrumból. Irreleváns mintának számítanak a DC-szint közeli minták (felbontásfüggő, de minimum az első komponens), illetve egy bizonyos relatív amplitúdószint alatti, vagy egy frekvenciaérték feletti minták.

3.4. Neurális hálózat megalkotása

3.4.1. `Newff` parancs

Egy N rétegű, feed-forward backpropagation neurális hálózatot hoz létre.

```
net = newff(PR, [S1 S2 ...  SN], TF1 TF2 ...  TFN, BTF, BLF, PF);
%bemeneti paraméterek:
```

```
%PR: egy adott bemeneti mintához tartozó min. és max. értékeket kell megadni
%(lásd Példa 1.)
%[S1 S2 ... SN]: a hálózat 1., 2., ..., n. rétegének mérete (neuronszáma)
%{TF1 TF2 ... TFN}: egy adott réteghez tartozó neuronok átviteli függvényei,
ĵdefault: 'tansig'
%BTF: backpropagation hálózat tanító függvénye, default = 'trainlm'
%BLF: backpropagation hálózat súlytanuló függvénye, default = 'learngdm'
%PF: performance function (hibaminimalizálás módszere), default: 'mse'
```

A neuronok kimeneti karakterisztikái az 1. táblázatban láthatók.

Átviteli függvény	Megadása	Algoritmus	Értékkészlet
Logaritmikus szigmoid függvény	`'logsig'`	$\frac{1}{1+e^{-n}}$	$(0;\ 1)$
Tangens szigmoid függvény	`'tansig'`	$\frac{2}{1+e^{-2n}} - 1$	$(-1;\ 1)$
Lineáris függvény (egyenes)	`'purelin'`	n	$(-\infty;\ \infty)$
Szaturált lineáris függvény	`'satlins'`	$y = \begin{cases} -1, \text{ ha } n \leq -1 \\ n, \text{ ha } -1 < n < 1 \\ 1, \text{ ha } n \geq 1 \end{cases}$	$[-1;\ 1]$

1. táblázat. Neuronok kimeneti karakterisztikái

A hálózat tanító módszerei a 2. táblázatban láthatók.

Tanító módszer	Megadása
gradient descent backpropagation	`'traingd'`
gradient descent with momentum backpropagation	`'traingdm'`
gradient descent with adaptive learning rule backpropagation	`'traingda'`
gradient descent with momentum and adaptive learning rule backpropagation	`'traingdx'`
resilent backpropagation	`'trainrp'`
scaled conjugate gradient backpropagation	`'trainscg'`
BFGS quasi-Newton backpropagation	`'trainbfg'`
one step secant backpropagation	`'trainoss'`
Levengerg-Marquardt backpropagation	`'trainlm'`

2. táblázat. Hálózat tanító módszerei

3.4.2. 1. példa

```
net = newff(minmax(input),[5 1],{'tansig' 'purelin'},'trainlm');
```

Az input egy oszlopvektorok összefűzésével kapott mátrix. Minden oszlopában egy minta adatai (pl. spektruma) találhatók. A `minmax` függvény oszloponként megadja a minimális és maximális értékeket. A mérésnél ezt célszerű használni. A második argumentum a neurális hálózat méretéről ad információt: kétrétegű a hálózat, az első (rejtett) réteg neuronszáma 5, a második (kimeneti) réteg neuronszáma 1. A harmadik argumentum az egyes rétegek neuronjaihoz tartozó karakterisztikákat adja meg: a rejtett réteg átvitele tangens-szigmoid, a kimeneti réteg átvitele lineáris. A negyedik argumentum a tanítási módszert adja meg, amely itt a Levenberg–Marquardt-algoritmus. Az utolsó két argumentumot nem adtuk meg, így azok default értékek: a súlytanulási módszer 'learngdm' (gradient descent with momentum), a hibaminimalizálás módszere 'mse' (átlagos négyzetes hiba). A backpropagation típusú hálózatoknál e két módszert ajánlott használni.

A mérés során célszerű több tanítási algoritmust kipróbálni: egyet a legmeredekebb lejtő módszerek közül, egyet pedig a változó irányú módszerek közül. A neuronok karakterisztikái legyenek:

- a rejtett rétegben: szigmoid;
- a kimeneti rétegben: lineáris.

3.5. Neurális hálózat tanítása

3.5.1. Train parancs

Egy adott neurális hálózat tanítását végzi.

```
[newnet, tr, Y, E, Pf, Af] = train(net, P, T, Pi, Ai, VV, TV);
%bemeneti paraméterek:
%net:  a tanítandó neurális háló
%P: a bemeneti minták tömbje (a minták oszlopvektorok, ezeket kell összefűzni)
%T: a kívánt kimeneti értékek (targetek), sorvektorként megadva, default:  0
%Pi:  a kezdeti bemeneti késleltetés, default:  0
%Ai:  a kezdeti rétegkésleltetés, default:  0
%VV: validációs minták struktúrája (lásd később), default:  []
%TV: tesztminták struktúrája, default:  []

%kimeneti változók:
%newnet:  a tanított neurális háló (lehet azonos a bemenettel)
%tr:  training record (tanítással kapcsolatos paraméterek)
%Y: a háló kimenetei
%E: a háló hibái
%Pf:  a végső bemeneti késleltetés
%Af:  a végső rétegkésleltetés
```

3.5.2. 2. példa

```
[NET,TR]= train(net,input,output,[],[],validation,test);
```

A `net` és az `input` ugyanaz, mint az 1. példában, az `output` a targeteket tartalmazza minden egyes mintára. Ahány oszlopa (mintája) van az inputnak, annyi targetet kell megadni az outputban (sorvektorként). A hálózat késleltetéseivel a mérés során nem foglalkozunk, így a negyedik és ötödik argumentum: `[]`. `validation` és `test` struktúrák. A `validation.P`, illetve a `test.P` alatt meg kell adni a validációs és tesztelési mintákat (oszlopvektorokban, összefűzve). A `validation.T`, illetve a `test.T` alatt a targeteket adjuk meg (sorvektorban).

3.5.3. A tanítás menete

A tanításhoz két adathalmazt használunk: tanítómintákat (training set) és tesztmintákat (test set). A tanítómintákat a hálózat paramétereinek módosításához használjuk, míg a tesztminták a hálózat hatékonyságát mutatják meg. A validációs minták a tanítóminták részhalmazát képezik.

Egy tanítási iteráció (epoch) három részből áll: a tanítóalgoritmusnak megfelelően a tanító mintákra módosítjuk a súlyokat (tanítjuk a hálózatot), majd kritériumot ellenőrzünk a validáció során (hogy a tanítás hatására megengedhető módon változott-e a hiba) és ennek megfelelően módosítjuk a paramétereket. Végül, a tanítás hatékonyságát a tesztmintákon vizsgáljuk meg (a tesztelésnek nincs hatása a tanításra).

A tanítás leállásának négy feltétele van:

1. elértük a maximális epoch számot (default: 100);
2. elértük a kitűzött performance értéket (default error goal: 0);
3. elértük a minimális gradienst (default error goal: 10^{-10});

4. validáció miatt maradt abba a tanítás (default max. fail: 5).

A train–validation–test pontok arányát körültekintően kell megválasztani. Ha túl sok a tanítási pont, akkor a validáció szerepe minimális lesz (hamar elérjük a minimális gradienst). Ha túl kevés a tanítási pont, akkor a validáció miatt hamar leáll a tanítás, és így nagyon rossz hatásfokú hálózatot kapunk. Javasolt az 50-70% train pontok esetén. A validation–test pontok aránya kb. 1:1-hez (a validációs mintákból lehet kissé több).

3.5.4. Tanítási paraméterek megváltoztatása

A mérésre nem jellemző, de előfordulhat, hogy egyes tanítási paramétereket meg kell változtatni. A fontosabb paraméterek a következők:

```
net.trainParam.epochs = 50              max.  epoch megadása
net.trainParam.goal = 0.001             kitűzött hiba célérték beállítása
net.trainParam.min_grad = 10^-5         min.  gradiens értékének beállítása
net.trainParam.max_fail = 8             max.  ennyi epoch-on keresztül nem javul a hiba
                                        (a validációs mintákra)
net.initFcn = 'initlay'                 súlyinicializációs függvény beállítása
net.initParam = []                      kezdeti súlyértékek megadása
net.performFcn = 'mse'                  performance function beállítása
net.trainFcn = 'trainlm'                tanítási algoritmus beállítása
```

Ha inicializációs paramétereket változtatunk meg, akkor – a hálózat tanítása előtt – kötelező inicializálni a hálózatot, a következőképpen:

```
net = init(net);
```

Az inicializáció során véletlen értekeket kapnak a súlyok. Ez nagyban megnehezíti az optimális hálózat megalkotását.

3.6. Neurális hálózat szimulációja

3.6.1. Sim parancs

Egy neurális hálózat működését szimulálja adott bemenetekre.

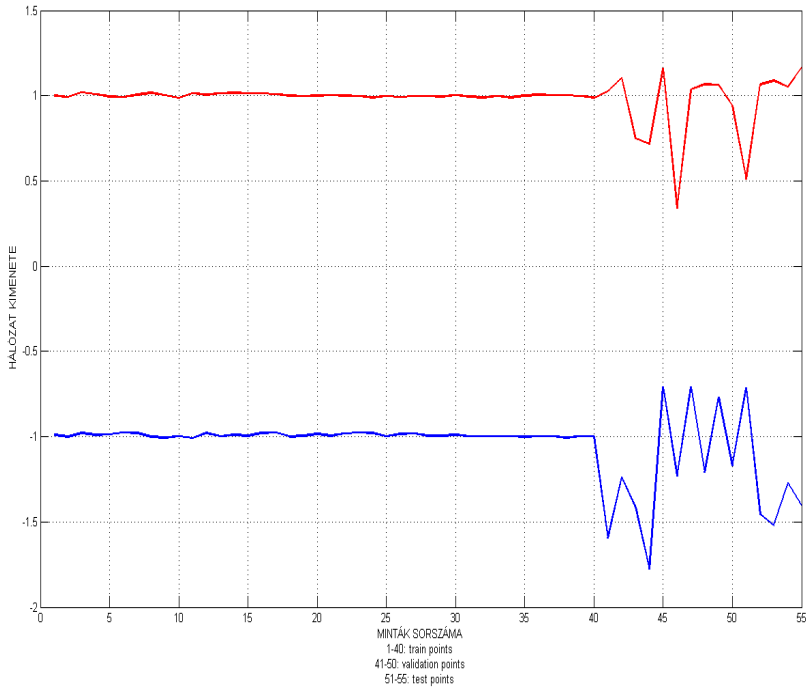
```
[Y, Pf, Af, E, perf] = sim(net, P, Pi, Ai, T);
%bemeneti paraméterek:
%net:  a szimulálandó hálózat
%P: a hálózat bemenetei (ezekre szimuláljuk a hálózat működését)
%T: targetek
%Pi:  a kezdeti bemeneti késleltetés
%Ai:  a kezdeti rétegkésleltetés

%kimeneti változók:
%Y: a háló kimenetei (a szimuláció eredménye)
%E: a háló hibái
%perf:  a hálózat teljesítőképessége
```

3.6.2. 3. példa

Tanított hálózat esetén elég csak az első két argumentummal meghívni a függvényt.

```
kimenet = sim(tanított_háló, bemenet);
```

A bemenet továbbra is oszlopvektorok összefűzéséből áll. Az 5. ábrán az egyik pohár hangjának felismerésére tanított hálózat szimulálási eredménye látható. A hálózat mérete 50×1 neuron, a spektrum 64 pontos DFT



5. ábra. Tanított pohár hangjának felismerését végző hálózat kimenete

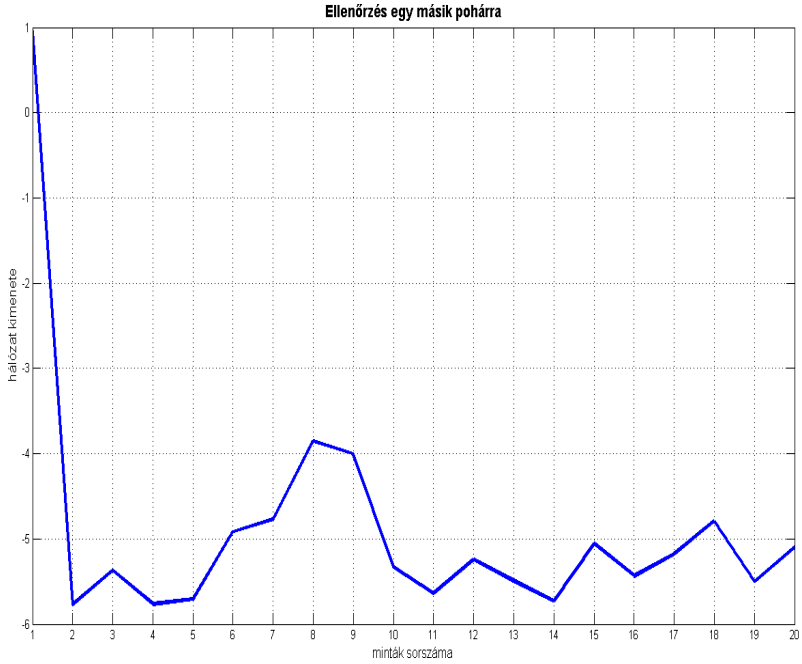
eredménye, amelyből nem hagytunk el mintát (tehát a nem domináns frekvenciakomponensekre is tanítottuk a hálózatot). A mérés során célszerű frekvenciaalapú szűrést végezni, így kisebb neuronszámmal is elfogadható eredményt kapunk. A 6. ábrán pedig egy másik (nem tanított) pohár mintáira adott eredmény látható. Hálózatunk inkább rossznak osztályozta az egy másik pohárról készített felvételt. Ennek az a magyarázata, hogy – bár azonos gyártósorról származnak a poharak – a hangjuk mégsem azonos, így a nagy neuronszámú (nagy érzékenységű) hálózatunk nem tolerálja ezt a frekvencia-eltolódást.

4. Osztályozás fuzzy rendszerrel

4.1. Az osztályozás alapgondolata

A feladat egy egyszerű, szóló zongoradarab dallamának kottázása (esetleg más hangszer is elképzelhető). Itt kotta alatt egy adott zeneműben azonosítható hangok időbeli lefutását értjük. A zenei művet időben kisebb szegmensekre kell feldarabolni, majd minden egyes szegmensnek meg kell határozni a zenei hangját (pitch notation). A zenei hang meghatározása osztályozási feladat, itt azonban, a pohár azonosításával szemben, nem csupán kettő, hanem több (pl. két oktáv esetén 23) osztályba kell sorolni a hangokat. A hangfelvételt nem a mérés során kell elkészíteni, az rendelkezésre áll, de saját felvétel is hozható (lásd még a mérési feladatok részt).

A zenei hangok magassága, és ennek megfelelően a kottában elfoglalt helye alapvetően az alapharmonikus frekvenciája alapján dönthető el. Vannak azonban olyan hangszerek és hangok (pl. hegedű mély hangjai), hogy nem az alapharmonikus a domináns; vagy olyan hangszerek (pl. ütőhangszerek), amelyek nem tiszta peri-



6. ábra. Nem tanított pohár hangjának felismerését végző hálózat kimenete

odikus (tehát alapharmonikus – felharmonikusok) hangokat produkálnak. Ezeknél a hangmagasság felismerése bonyolultabb, ezért választottuk mérési feladatként a zongorahangot (orgona, furulya is lehet).

A hangok azonosításához felhasználható az alábbi táblázat:

`http://en.wikipedia.org/wiki/Scientific_pitch_notation#Table_of_note_frequencies`

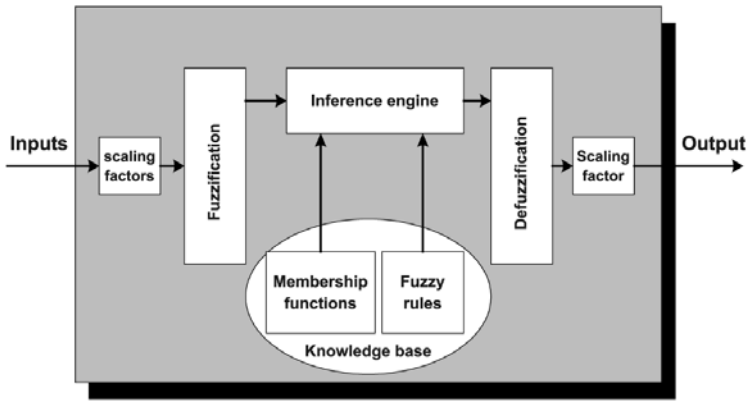
Zongorahangok azonosításához célszerűbb a következő táblázat használata:

`http://en.wikipedia.org/wiki/Piano_key_frequencies`

A megvalósítandó fuzzy rendszer feladata, hogy egy bemeneti dallamrészlethez (szegmenshez) egy zenei hangot rendeljen. Itt ügyelni kell arra, hogy a fuzzy szabálybázis eltérő lehet más-más hangszerre. Zongoradaraboknál egy egyszerű maximumkeresés a feladat, azaz a spektrumban domináns komponens frekvenciája lesz a zenei hang frekvenciája. A zenei hangokhoz rendelt alapharmonikus-frekvenciák nem teljesen egyértelműek, továbbá kisebb-nagyobb mértékben minden hang hamis (a csak nagyon kicsit hamis hangokat ítéljük jónak), ezért az osztályozás nem egyszerű leképezés a frekvenciák és a hangok között. Az osztályozás végrehajtására nem lenne feltétlenül szükség fuzzy rendszerre, hiszen ez megoldható lenne a detektált frekvenciához legközelebb eső zenei hang hozzárendelésével is. A mérés azonban rámutat arra, hogy a fuzzy rendszer felépítéséből és működéséből mintegy kiadódik egy ilyen egyszerű eljárás megvalósítása is, továbbá a rendszer alkalmas olyan irányú bővítésre, hogy bonyolultabb szabályok alapján is képes legyen zenei hangfelismerésre.

4.2. Az alkalmazott fuzzy következtető rendszer

Egy fuzzy rendszer (Fuzzy Inference System – FIS) a 7. ábrán látható egységekből épül fel. A fuzzifikálás egy crisp (valós) bemeneti értékhez fuzzy tagsági érték hozzárendelése. A mérés során szingleton fuzzifikálást használunk. A következtető gép (Inference Engine) adott fuzzy bemeneti értékékhez a tudásbázis (Knowledge Base) segítségével fuzzy kimeneti értékeket rendel. A tudásbázis tagsági függvényekből és a szabálybázisból (fuzzy szabályokból) áll. A defuzzifikálás fuzzy értékekből crisp kimenetet állít elő.



7. ábra. Fuzzy következtető rendszer

A hangok azonosítása itt is spektrumszámítással kezdődik. A spektrumból a maximális amplitúdójú komponenst kell továbbvinni a következtető rendszerbe. Az egyes skálabeli hangokhoz tagsági függvényeket rendelhetünk, így a tagsági függvény értékének fizikai jelentése az, hogy a spektrum maximumának megfelelő frekvencia melyik skálabeli hanghoz milyen mértékben tartozik hozzá. A tagsági függvényeket tehát úgy kell definiálni, hogy maximumuk az adott skálabeli hang névleges frekvenciáján legyen.

A szabályok az adott tagsági függvényérték mellett figyelembe veszik, hogy milyen amplitúdója volt az adott komponensnek. Ha túl alacsony, nem minősítik hangnak az adott szegmenst. Alacsony amplitúdó adódhat abból, hogy az adott szegmens idején egyetlen hang sem szól, illetve azért is, mert két hang közötti váltás esik a szegmensre, így egy adott frekvenciához kisebb amplitúdó tartozik, még megfelelő hangerő esetén is. A defuzzifikáció már visszaadja az adott skálahang frekvenciáját. Jelen esetben ezt pl. Mamdani-rendszert és a legnagyobb tagsági érték közepét választva érhetjük el.

A fuzzy rendszer tehát mintegy „behúzza" a nem tökéletes frekvenciát a megfelelő értékre, feltéve, ha elegendően nagy az amplitúdója. A rendszer akkor is működőképes lenne, ha minden spektrumkomponenst továbbvinnénk, ekkor azonban a szabálybázist kellene sokkal összetettebben megalkotni. Ezen az úton a fuzzy rendszer továbbfejleszthető.

A méréshez a MATLAB Fuzzy Logic Toolboxát használjuk.

5. Megvalósítás MATLAB-ban

5.1. Szegmentálás

A választott dallamot fel kell osztani egyenlő hosszú szegmensekre, majd minden szegmensre külön-külön spektrumot kell számítani. Itt célszerű nagy DFT-pontszám használata (legalább 1024), mivel két szomszédos hang közötti frekvenciatávolság nagyon kicsi is lehet (10 . . . 20 Hz). A kapott spektrumok lesznek a fuzzy rendszer bemenetei. A spektrumból célszerű csak azokat a pontokat használni, amelyek az adott dallam hangtartományába beleesnek (pl. C-dúr skála esetén teljesen felesleges 200 Hz alatti és 600 Hz feletti spektrumkomponensekkel foglalkozni).

A szegmentálásra célszerű külön Matlab-függvényt írni, amely átlapolódó szegmenseket is képes generálni. A függvény egy lehetséges deklarációja:

```
function     szegmensek = szegmental(dallam, szegmens_hossz, átlapolódás_mértéke);
             %bemeneti paraméterek:
             %dallam:  a szegmentálandó felvétel
             %szegmens_hossz:  egy szegmens mérete
```

```
%átlapolódás_mértéke:  átlapolódó minták száma

%kimeneti változó:
%szegmensek:  a szegmentált felvétel
```

Az átlapolódó szegmensekkel minimalizálható a szegmensen belüli hangváltás által okozott probléma (hangváltás esetén milyen hangot tulajdonítsunk a szegmensnek).

5.2. Fuzzy rendszer megalkotása

5.2.1. Newfis parancs

Létrehoz egy FIS-t, az első paramétert kötelező megadni.

```
a = newfis(fisName, fisType, andMethod, orMethod, impMethod, aggMethod, defuzzMethod);
%bemeneti paraméterek:
%fisName:  a létrehozandó fuzzy rendszer neve
%fisType:  a FIS típusa:  'mamdani' vagy 'sugeno'
%andMethod:  ÉS módszer megadása:  'min' vagy 'prod' (prod:  szorzás)
%orMethod:  VAGY módszer megadása:  'max' vagy 'probor' (probor:  valószínűségi)
%impMethod:  implikációs módszer:  'mamdani' esetén:  'min' vagy 'prod',
%'sugeno' esetén:  'prod'
%aggMethod:  aggregációs módszer:  'mamdani' esetén:  'max', 'sum' vagy 'probor',
%'sugeno' esetén:  'probor'
%defuzzMethod:  deffuzifikációs módszer:  'mamdani' esetén:  'centroid', 'bisector',
% 'mom', 'som' vagy 'lom', 'sugeno' esetén:  'wtaver' vagy 'wtsum'

%kimeneti változó:
%a:  a létrehozott hálózat
```

Valamennyi argumentumot sztringként kell megadni. Legegyszerűbb esetben elegendő csak a rendszer nevét megadni. Ilyenkor egy default Mamdani-féle fuzzy rendszer jön létre.

5.2.2. Getfis parancs

Visszaadja egy létező FIS tulajdonságait.

5.2.3. 4. példa

```
a = newfis('Mamdani-féle','mamdani');
getfis(a)

Name         =           Mamdani-féle
Type         =           mamdani
NumInputs    =           0
InLabels     =
NumOutputs   =           0
OutLabels    =
NumRules     =           0
AndMethod    =           min
OrMethod     =           max
ImpMethod    =           min
```

```
            AggMethod    =            max
            DefuzzMethod = centroid

b = newfis('Takagi-Sugeno-Kang-féle','sugeno');
            getfis(b)
            Name         =            Takagi-Sugeno-Kang-féle
            Type         =            sugeno
            NumInputs    =            0
            InLabels     =
            NumOutputs   =            0
            OutLabels    =
            NumRules     =            0
            AndMethod    =            prod
            OrMethod     =            probor
            ImpMethod    =            prod
            AggMethod    =            sum
            DefuzzMethod =            wtaver
```

5.2.4. Setfis parancs

Egy létező FIS tulajdonságait változtatja meg (kívánt értékre állítja be). Három különböző megadási módja van.

1. Ez a megadás egy adott tulajdonságot változtat meg.

```
myFIS = setfis(myFIS, 'propname', 'newprop');
%bemeneti paraméterek:
%myFIS: a módosítandó fuzzy rendszer
%'propname':  a megváltoztatandó tulajdonság neve
%('name', 'type', 'andmethod', 'ormethod', 'impmethod', 'aggmethod'
%vagy 'deffuzmethod')
%'newprop':  az új tulajdonság értéke (lásd fent)
```

2. Ez a megadás egy adott bemeneti/kimeneti változó nevét, illetve értelmezési tartományát változtatja meg.

```
myFIS = setfis(myFIS, 'vartype', varindex, 'varpropname', 'newvarprop');
%bemeneti paraméterek:
%myFIS: a módosítandó fuzzy rendszer
%'vartype':  a változó típusa:  'input' vagy 'output'
%'varindex':  a bemeneti/kimeneti változó sorszáma
%'varpropname':  a beállítandó változómező:  'name' vagy 'range'
%'newvarprop':  'name' esetén:  a változó új neve,
%'range' esetén:  a változó új értelmezési tartománya
```

3. Ez a megadás egy adott bemeneti/kimeneti változóhoz tartozó tagsági függvény görbesereg közül kiválaszt egyet, és annak a nevét, típusát, illetve paramétereit változtatja meg.

```
myFIS = setfis(myFIS, 'vartype', varindex, 'mf', mfindex, 'mfpropname', 'newmfprop');
%bemeneti paraméterek:
%myFIS: a módosítandó fuzzy rendszer
```

```
%'vartype':  a változó típusa:  'input' vagy 'output'
%'varindex':  a bemeneti/kimeneti változó sorszáma
%'mf':  ebben az alakban kell megadni
%'mfindex':  az adott bemeneti/kimeneti változóhoz tartozó tagsági függvény sorszáma
%'mfpropname':  a beállítandó tulajdonság:  'name', 'type', 'params'
%'newmfprop':  'name', 'type' esetén:  új név/típus,
%'params' esetén:  sorvektorként megadva az adott tagsági függvény
%típusához tartozó paraméter
```

5.3. Tagsági függvények megadása

5.3.1. Addvar parancs

Egy bemeneti/kimeneti változó értelmezési tartományát adja meg.

```
myFIS = addvar(myFIS, 'varType', 'varName', varBounds);
%bemeneti paraméterek:
%myFIS: a fuzzy rendszer
%'vartype':  a változó típusa:  'input' vagy 'output'
%'varName':  a változó neve
%'varBounds':  a változó értékkészletének szélső határai (min.  és max.  érték)
```

5.3.2. 5. példa

```
kotta        =           newfis('Kottázzás!!!');
kotta        =           addvar(kotta,'input','frekvencia',[200 600]);
kotta        =           addvar(kotta,'input','amplitúdó',[0.0001 10]);
kotta        =           addvar(kotta,'output','hang',[0 5]);
```

Létrehoztunk két bemeneti és egy kimeneti változót. Az azonos típusú változók indexe megegyezik a létrehozási sorrenddel, vagyis a 'frekvencia' indexe 1, az 'amplitúdó' indexe 2, a 'hang' indexe 1. A frekvencia nevű változó 200 és 600 között van definiálva, a jelamplitúdó 10^{-4} és 10 között mozog. A kimeneten pedig négy hangot különbözhetünk majd meg (lásd 6. példa).

5.3.3. Addmf parancs

Egy létező bemeneti/kimeneti változóhoz hozzáad egy új tagsági függvényt (MF: membership function).

```
myFIS = addmf(myFIS, 'varType', varIndex, 'mfName', 'mfType', mfParams);
%bemeneti paraméterek:
%myFIS: a fuzzy rendszer
%'vartype':  a változó típusa:  'input' vagy 'output'
%'varIndex':  a változó indexe (lásd 5.  példa)
%'mfName':  a létrehozandó tagsági függvény neve
%'mfType':  a tagsági függvény típusa (lásd alább)
%'mfParams':  az adott típusú függvény paraméterei (lásd alább)
```

A tagsági függvények típusai és paraméterei a 3. táblázatban láthatók.

Tagsági függvény	'mfType'	'mfParams'	Paraméterek jelentése
Gauss-görbe	'gaussmf'	`[sig c]`	`sig`: szórás, `c`: várható érték
háromszög	'trimf'	`[a b c]`	a háromszög csúcsainak x-koordinátája: $(a < b < c)$ $\mu(a) = \mu(c) = 0$; $\mu(b) = 1$
trapéz	'trapmf'	`[a b c d]`	a trapéz csúcsainak x-koordinátája: $(a < b < c < d)$ $\mu(a) = \mu(d) = 0$; $\mu(b) = \mu(c) = 1$
S-alakú	'smf'	`[a b]`	$\mu(x) = \begin{cases} 0, \text{ ha } x < a \\ 1, \text{ ha } x > b \end{cases}$
Z-alakú	'zmf'	`[a b]`	$\mu(x) = \begin{cases} 1, \text{ ha } x < a \\ 0, \text{ ha } x > b \end{cases}$

3. táblázat. Tagsági függvények

5.3.4. 6. példa

Az 5. példában létrehozott rendszert kiegészítve:

```
kotta        =            addmf(kotta,'input',1,'mély','trapmf',[200 250 350 450]);
kotta        =            addmf(kotta,'input',1,'magas','trapmf',[350 450 550 600]);
kotta        =            addmf(kotta,'input',2,'kicsi','zmf',[0.1 1]);
kotta        =            addmf(kotta,'input',2,'nagy','smf',[0.1 1]);
for n = 1:4,
             kotta = addmf(kotta,'output',1,num2str(n),'trimf',[(n-1),n,(n+1)]);
end
```

Az `input 1` a frekvencia, melyet két trapéz alakú tartományra osztottunk (magas és mély). Az `input 2` az amplitúdó, melyet szintén két tartományra osztottunk (kicsi és nagy). A kimenetet négy, háromszög alakú tartományra osztottuk. A `num2str` utasítás egy integert alakít sztringgé (mivel a tagsági függvény neve csak sztring lehet).

5.3.5. Plotmf parancs

```
plotmf(myFIS, 'varType', varIndex);
```

Az adott bemeneti/kimeneti változóhoz tartozó tagsági függvény görbesereget ábrázolja.

5.3.6. 7. példa

A fenti három (frekvencia, amplitúdó, hang) változó görbeseregeit ábrázoljuk.

`plotmf(kotta,'input',1);` A görbék a 8. ábrán láthatók.

`plotmf(kotta,'input',2);` A görbék a 9. ábrán láthatók.

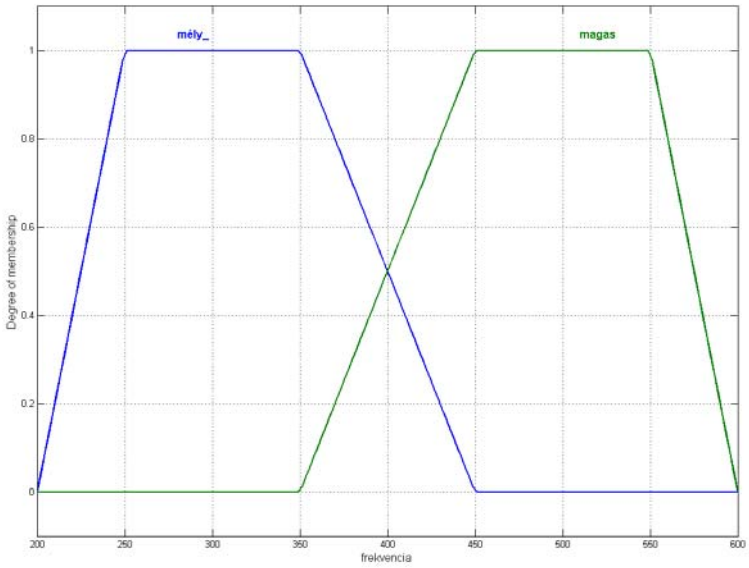
`plotmf(kotta,'output',1);` A görbék a 10. ábrán láthatók.

5.4. Szabálybázis megalkotása

5.4.1. Addrule parancs

Egy meglévő szabálybázist ad hozzá a fuzzy rendszerhez.

```
myFIS = addrule(myFIS, ruleList);
%bemeneti paraméterek:
%myFIS: a fuzzy rendszer
```



8. ábra. A frekvenciára vonatkozó tagsági függvények

```
%'ruleList':  a FIS szabálybázisa, mérete:  r × (i + o + 2),
%ahol r a szabályok száma; i, o a bemenetek/kimenetek száma.
%Tehát, a mátrix egy sora egy darab szabályt tartalmaz!
```

5.4.2. Szabályok felépítése:

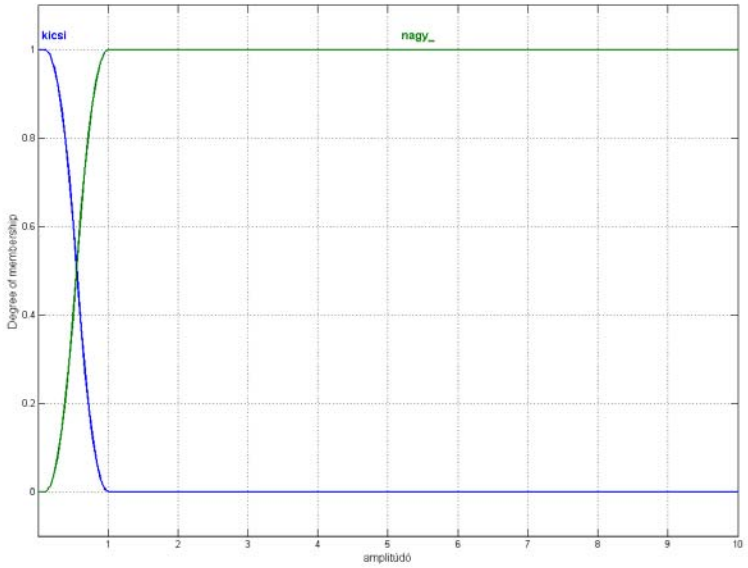
Az első i db oszlop az adott bemeneti változóhoz tartozó tagsági függvény sorszáma. A következő o db oszlop az adott kimeneti változóhoz tartozó tagsági függvény sorszáma. Az $(i + o + 1)$-edik oszlop: a szabály súlyát kell megadni (0 és 1 között), többnyire 1 értékű. Az $(i + o + 2)$-edik oszlop: értéke 1, ha az antecedens operátora ÉS, értéke 2, ha az operátor: VAGY. A tagsági függvények sorszáma (a változók sorszámához hasonlóan) a létrehozási sorrenddel egyezik meg.

5.4.3. 8. példa

Szabálybázis megalkotása a kotta nevű FIS-re. Itt most egy teljesen határozott szabálybázist adunk meg (minden lehetséges bemeneti kombinációhoz tartozik egyértelmű kimenet), de a fuzzy rendszer működéséhez ez nem feltétlenül szükséges.

R1. HA (frekvencia = mély) ÉS (amplitúdó = kicsi), AKKOR (hang = 1).
R2. HA (frekvencia = mély) ÉS (amplitúdó = nagy), AKKOR (hang = 2).
R3. HA (frekvencia = magas) ÉS (amplitúdó = kicsi), AKKOR (hang = 3).
R4. HA (frekvencia = magas) ÉS (amplitúdó = nagy), AKKOR (hang = 4).

```
szabaly      =          [ 1 1 1 1 1;
                        1 2 2 1 1;
                        2 1 3 1 1;
                        2 2 4 1 1];
kotta        =          addrule(kotta,szabaly);
```



9. ábra. Az amplitúdóra vonatkozó tagsági függvények

5.5. Deffuzifikáció

A setfis paranccsal be lehet állítani a kívánt deffuzifikációs módszert. A deffuzifikáció minden esetben a kimeneti változó x-koordinátáját adja vissza.

1. Mamdani-féle rendszer esetén:
   - centroid: center of area (kimeneti alakzat súlypontját adja vissza);
   - bisector: az az egyenes, amely két egyenlő területű részre osztja az alakzatot;
   - mom: middle of maximum (legnagyobb tagsági értékek közül a középső),
   - som: smallest of maximum (legnagyobb tagsági értékek közül a legkisebb),
   - lom: largest of maximum (legnagyobb tagsági értékek közül a legnagyobb).
2. Takagi-Sugeno-Kang-féle rendszer esetén:
   - wtaver: weighted average (súlyozott átlag);
   - wtsum: weighted sum (súlyozott összeg).

5.5.1. 9. példa

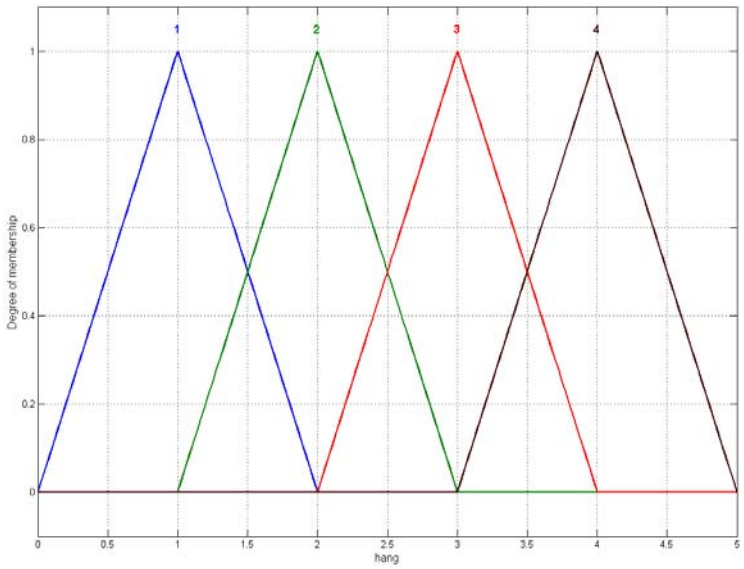
Változtassuk meg a deffuzifikációs módszert a default értékről 'mom'-ra!

```
kotta = setfis(kotta,'defuzzmethod','mom');
```

5.6. Kottázás

5.6.1. Evalfis parancs

Egy létező FIS bemeneteire adott kimeneti értékeket értékeli ki.

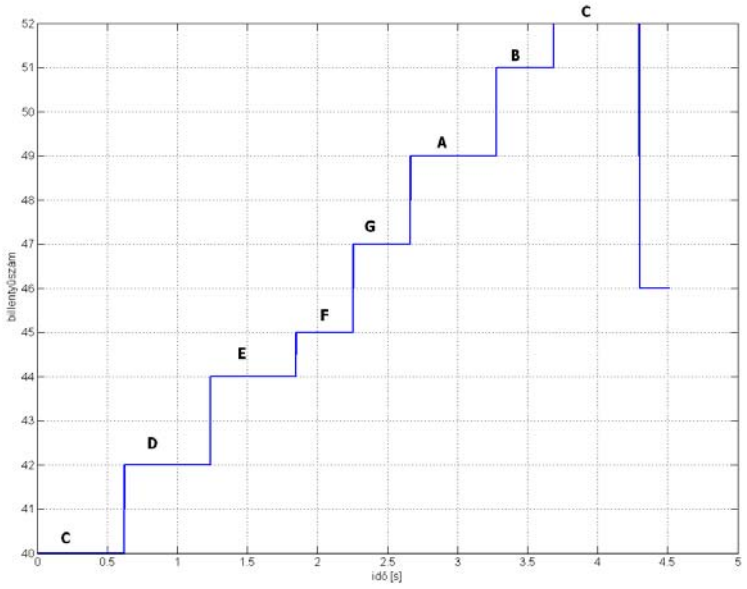


10. ábra. A hangjegyekre vonatkozó tagsági függvények

```
out = evalfis(input, myFIS);
%bemeneti paraméterek:
%myFIS: a fuzzy rendszer
%input:  a rendszer bemeneti értékei [input_1 input_2 ...  input_N] alakban,
%ahol input_i oszlopvektor (azaz a minták sorvektorok)

%kimeneti változó:
%out:  a rendszer kimenete
```

5.6.2. Kotta ábrázolása

Minden egyes szegmenshez a fuzzy rendszer hozzárendel egy zenei hangot. Zongoradallamok esetén célszerű egy zongorabillentyű sorszámot hozzárendelni az adott szegmenshez, mivel a zongorabillentyű száma egyértelműen azonosítja a zenei hangot. A C-dúr skáláról készített felvétel kottájának ábrázolása látható a 11. ábrán. A szegmensekhez tartozó billentyűszámot ábrázoljuk az idő függvényében (egy adott billentyűhöz tartozó hang hány szegmensnyi ideig szól).



11. ábra. A fuzzy rendszer kimenete

6. A mérési feladatok

6.1. Osztályozás neurális hálózattal

1. Ütögetéssel rögzítse az üres pohár, majd a kapott anyaggal kitöltött pohár hangját. Végezzen legalább 50–50 megütést!

2. Töltse be a hangmintákat a Matlabba! A mintákból vágjon ki egyenlő hosszú intervallumokat (minden egyes megütésre)!

3. Az üres, illetve a megtöltött pohár kivágott mintáira számoljon spektrumot és hasonlítsa őket össze! A neurális háló tanításához válassza ki az optimális DFT-pontszámot, majd a kapott spektrumokból próbáljon – a tanításhoz irreleváns – pontokat elhagyni!

4. Tervezzen egy kétrétegű, egy kimenetű neurális hálót a `newff` parancs segítségével, majd végezze el e hálózat tanítását a `train` parancs segítségével! A target legyen két, szignifikánsan különböző érték. A rejtett réteg neuronszámának változtatásával keresse meg az optimális hálóméretet!

5. Egy másik pohár hangjáról készítsen felvételeket, vágja ki a megfelelő mintákat, a korábban kiválasztott DFT-pontszámra számoljon spektrumokat, majd e mintákkal szimulálja le a fenti neurális hálózatot. Milyen kimeneti értékeket kapunk, és ezekből milyen következtetést lehet levonni?

6.2. Osztályozás fuzzy rendszerrel

A méréshez egy zenei adatbázis áll rendelkezésre, de nem kötelező ezt használni. Lehet saját dallamot is hozni, de ebben az esetben az alábbi kritériumoknak teljesülniük kell:

- csak egy hangszer játszik;
- egyszerre csak egy hang szól;

- a dallamban szereplő hangtartomány nem szélesebb két oktávnál.

Ilyenkor a kottázás ellenőrzéséhez célszerű kottát is hozni.

1. Válasszon ki egy dúr- vagy moll-skálát, majd végezze el a szegmentálást (átlapolódás nélküli esetben), illetve a spektrumszámítást! A spektrumból hogyan azonosítható egy adott szegmens zenei hangja?
2. Tervezzen meg egy fuzzy rendszert, amely a bemenetére kapott spektrumhoz hozzárendel egy zenei hangot!
3. Végezze el a skála kottázását, majd a kottát grafikusan ábrázolja!
4. Átlapolódó szegmensek esetén végezze el az előző feladatokat! Hasonlítsa össze a két esetet!
5. Az adatbázisból válasszon ki egy dallamot, majd oldja meg a fenti feladatokat! Értékelje a kottázás eredményét!