Debreceni Egyetem
Informatikai Kar

A Fuzzy ART neurális hálózatok alkalmazása

Témavezető:
Dr. Münnich Ákos
egyetemi docens

Készítette:
Nemes Sándor
programtervező matematikus

Debrecen
2008.

Tartalomjegyzék

Köszönetnyilvánítás

Szeretnék köszönetet mondani Dr. Münnich Ákosnak, aki elvállalta a diplomamunkám témavezetését, és tanácsaival segítette annak elkészítését.

1. Bevezetés

Talán mindenkiben felmerült már az egyszerű kérdés: hogyan is működik az agyunk? Hogyan vagyunk képesek tanulni? Hogyan építhetünk ilyen intelligens, alkalmazkodni, tanulni képes rendszereket? A számítógépes rendszerek sebessége, komplexitása hihetetlen ütemben növekszik. Vannak területek, ahol a számítógépek megbízhatósága és gyorsasága megkérdőjelezhetetlen. Nincs olyan ember, aki jobban tudna mátrixokat invertálni vagy differenciálegyenleteket megoldani, mint egy számítógépes rendszer. De vannak problémák, amellyel a számítógépek meglepően nehezen boldogulnak. Igencsak rosszak vagy lassúak az arcok felismerésében, amelyek egy embernek nem jelentenek problémát. Néha nem a gyors és precíz számítás a cél, hanem az emberhez hasonló viselkedés. Ha az emberi agyat próbáljuk meg utánozni, akkor néha hibázni kell, ahogyan előfordul, hogy egy ember is hibázik. Ehhez adnak eszközt a mesterséges neurális hálózatok.

Ebben a diplomamunkában olyan neurális hálózatokat ismertetünk, amelyek a Stephen Grossberg nevével fémjelzett adaptív rezonancia elméletén alapulnak. Először bemutatunk egy önszervező, nem felügyelt tanulású fuzzy neurális hálózatot, a Fuzzy ART-t [1], majd ennek egy heteroasszociatív tanulásra alkalmas kiterjesztését a Fuzzy ARAM rendszert [2]. A diplomamunka második felében ezeket az elméleti alapokat átültetjük a gyakorlatba: A Fuzzy ART és Fuzzy ARAM hálózatok felhasználásával bemutatunk egy webes keretrendszert amely a Survey ART [3] neurális hálózat implementációján alapul. Itt egy szimulációban azt próbáljuk megvizsgálni, hogy a mesterséges tesztalanyok véleményére milyen hatással vannak a különböző üzenetek. A mesterséges aggyal rendelkező tesztalanyok az üzenetek alapján különböző asszociációkat végeznek és következtetéseket vonnak le, amelyekről egy webes felületen kérdezhetjük ki őket.

2. Elméleti háttér

Ebben a részben rávilágítunk a biológiai neuronok és a mesterséges neurális hálókat alkotó neuronok közötti hasonlóságra. Áttekintjük a mesterséges neurális hálózatok működésének legfontosabb elemeit, majd bemutatjuk az adaptív rezonancia elméletét, és végül két erre az elméletre épülő neurális hálózatot a Fuzzy ART-t és a Fuzzy ARAM-ot.

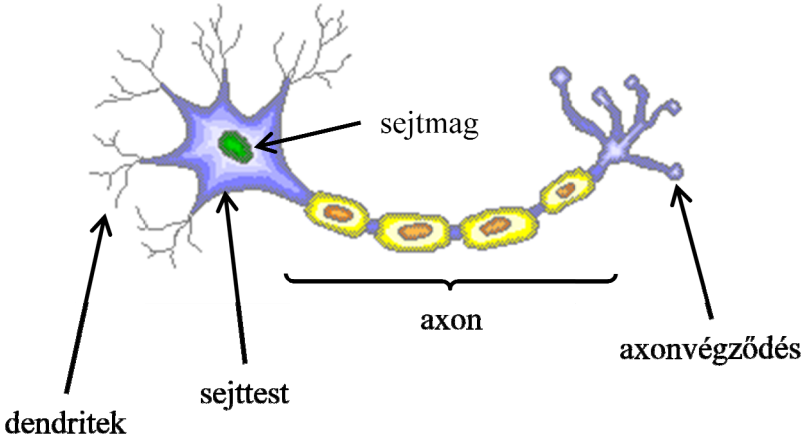
2.1. A neurális hálózatok alapjai

A neurális hálózat kifejezést már több mint száz éve használják a gerinces és gerinctelen állatok idegrendszerét alkotó biológiai neuronok hálózatának elnevezésére. Az 1940-es és főleg az 1980-as évek óta a kifejezés azonban már gyakran mást jelent. Egy olyan párhuzamos számítási technológiát, amelyben a számítást végző elemeket a biológiai neuronokról mintázták. Ezeknél a mesterséges neurális hálózatoknál, ezt a biológiai megközelítést gyakran feláldozzák és a statisztika és a jelfeldolgozás alapjaira építve, a gyakorlatban jobban alkalmazható megoldások után kutatnak. A mesterséges neurális hálózatokat azóta számos területen alkalmazzák: függvényközelítés, osztályozás, mintafelismerés, jelfeldolgozás, adatbányászat vagy esetleg a tőzsdei árfolyamok előrejelzése.

2.1.1 A biológiai neuron

Sokszor volt már rá példa hogy a kutatók a természetből lestek el valamilyen módszert, trükköt, nincs ez másképpen a mesterséges neurális hálózatoknál sem. A mesterséges neurális hálózatok olyan matematikai modellek, amelyek megpróbálják általánosan leírni az emberi érzékelést, észlelést, tanulást.

A biológiai neuronok egy *sejttestből*, egy hengeres *axonból* és gyökérszerű *dendritekből* állnak. A többi neurontól származó kémiai és elektromos jeleket a dendritek veszik fel, majd az axonon keresztül egy ingerületet küldenek tovább az agy felé. Tehát tekinthetünk úgy a dendritekből származó információkra, mintha az inputja lenne a neuronnak, az axonon keresztül távozó ingerület pedig az output. Innen jött az ötlet egy olyan

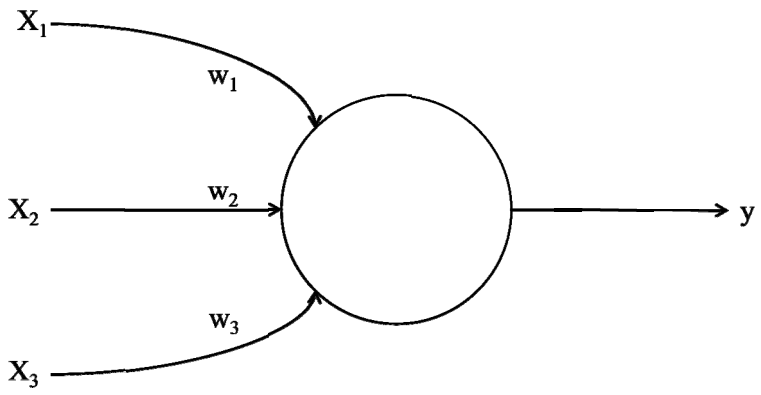


1. ábra: A biológiai neuron felépítése

párhuzamos, elosztott számítási modell kialakításához, amelyben az információ feldolgozását sok, egymáshoz hasonló csomópont, a mesterséges neuron végzi.

2.1.2 A mesterséges neuron

A mesterséges neuron tehát a biológiai neuron absztrakciója. A biológiai neuronnak több, különböző típusa van, a mesterséges neuron ezeknek a típusoknak a közös aspektusait próbálja összefogni. A mesterséges neuronokat kommunikációs csatornák kötik össze, amelyek valamilyen formában kódolt numerikus adatokat hordoznak. Az egységek csak a saját helyi memóriájukat használhatják, az adatokat pedig a kommunikációs csatornán (összeköttetéseken) kapják és továbbítják. Egy ilyen feldolgozóegységet, mesterséges neuront mutat a 2. ábra.



2. ábra: A mesterségen neuron felépítése

Az ábrán x_1, x_2, x_3 értékek adják a neuron inputját, a w_1, w_2, w_3 értékek az összeköttetés súlyai, az y pedig az output. A bemeneti értékeken a neuron végrehajt valamiféle összegzést, ez lehet egy egyszerű összeadás, esetleg minimum, maximum, átlag függvény vagy logikai műveletek. Ezt követően egy nemlineáris φ aktivációs függvény ezt az összegzett értéket átalakítja a neuron kimenetén megjelenő értékké, ami általában egy valós érték a [0, 1] vagy [-1, 1] intervallumban. Aktivációs függvénynek gyakran valamilyen szigmoid függvényt választanak, például a logisztikus függvényt vagy a hiperbolikus tangenst. [4]

Az első neurális hálózat megtervezése Warren McCulloch és Walter Pitts nevéhez fűződik, akik megalkották a McCulloch-Pitts neuront [5], azonban az első gyakorlati jelentőségű neuron modell a perceptron [6] volt, amelyről később kiderült, hogy korlátozott képességekkel rendelkezik, mivel csak lineárisan szétválasztható osztályokat képes megtanulni.

2.1.3 A hálózat felépítése

A mesterséges neurális hálózat egységei rétegekbe szerveződnek, amelynek három típusa lehet: *bemeneti*, *kimeneti* és *rejtett* réteg. Az első réteg a bemeneti réteg, közötte tetszőleges számú rejtett réteg helyezkedhet el, végül az utolsó réteg a kimeneti réteg. Az egyes rétegek neuronjai a súlyokkal ellátott irányított összeköttetéseken keresztül kapcsolódnak egymáshoz. A hálózat topológiáját tekintve két nagy típus terjedt el: a *visszacsatolt (feedback)* és az *előrecsatolt (feedforward)*. Az *előrecsatolt* neurális hálózatokban az összeköttetések között nincsenek hurkok, ezért ezek a hálózatok általában gyorsan előállítják a bemenetre adott választ. A *visszacsatolt* vagy *rekurrens* hálózatokban az összeköttetések között megengedettek a hurkok, így egyes esetekben a neurális hálózat egy viszonylag hosszú ideig iterál körbe, mielőtt választ adna. Ezeknek a hálózatoknak a tanítása általában nehezebb, mint az előrecsatolt hálózatoké.

2.1.4 Tanulási folyamat

Gyakorlati oldalról tekintve a neurális hálózatok nemlineáris statisztikai adatmodellező és döntéshozó eszközök. Bonyolult kapcsolatokat modellezhetünk a bemenő és

kimenő adatok között, vagy mintákat kereshetünk az adatokban. A legtöbb neurális hálózatnak van valamiféle *tanulási szabálya*, ami az összeköttetés súlyait a bejövő adatok alapján módosítja. Más szóval, a neurális hálózatok példák alapján tanulnak, és megfelelő tanítással, bizonyos határokon belül képesek általánosításokra is, vagyis helyes közelítő értékeket tudnak adni az új bemenő adatokra is.

A tanulási algoritmusoknak két fő csoportja van: *felügyelt* és *nem felügyelt*. A *felügyelt tanulás* során a célértékek ismertek, és tanítás közben a neurális hálózat a bemenő adatok mellett ezeket is megkapja, és ennek alapján igazítja az összeköttetések súlyait, hogy a kimenet közelítsen a célértékhez. Tanulás után a hálózat már csak a bemenő adatokat kapja meg, itt derül ki mennyire volt sikeres a tanulás, mennyire vannak közel a kívánt célértékhez. *Nem felügyelt tanulás* során a hálózat nem kapja meg az elvárt kimeneti értékeket. Itt a neurális hálózatok általában valamilyen „adattömörítést" végeznek el, pl. dimenziócsökkentést vagy klaszterezést. A felügyelt és nem felügyelt módszer közötti határvonal nem egészen éles. Egy nem felügyelt tanulású rendszer megtanulhat például egy eloszlásfüggvényt és ennek segítségével adhat előrejelzéseket. Továbbá a felügyelt tanulásnak van két további alkategóriája: az *autoasszociatív* és a *heteroasszociatív*. *Autoasszociatív* tanulás esetén a célérték megegyezik a bemenő értékkel, míg *heteroasszociatív* tanulás esetén a célérték általában különbözik. Sok felügyelet nélküli tanítási módszer ekvivalens az autoasszociatív felügyelt módszerrel. [7]

2.2. Az adaptív rezonancia elmélet

A mesterséges neurális hálózatok egy népes és egyre bővülő családja az adaptív rezonancia elméletére (ART - Adaptive Resonance Theory) épül, amelyet a bostoni egyetem professzora, Stephen Grossberg alkotott meg 1976-ban [8]. Grossberg munkásságának jelentőségét mi sem bizonyítja jobban, hogy kutatásait az Egyesült Államok védelmi minisztériumának kutatásokért felelős részlege, a DARPA támogatja, ami a 60-as évek végén az Internet elődjének tartott ARPANET kifejlesztésében is meghatározó szerepet játszott.

2.2.1 Az elmélet kialakulása

Az elmélet neurobiológiai alapokon nyugszik és célja, hogy megpróbálja modellezni azt, ahogyan az ember a kognitív információkat feldolgozza. Egy olyan próbálkozás eredményeképpen született, amelyben Grossberg azt próbálta megérteni, hogy hogyan képesek a biológiai rendszerek az élet során megőrizni formálhatóságukat anélkül, hogy ez a korábban megtanult minták stabilitásának rovására menne. A biológiai alapú tanulási rendszerek valahogyan megvédik a tárolt információkat az átmeneti változásokkal szemben, viszont megőrzik a formálhatóságukat, és ennek segítségével tudnak a környezetünkben új eseményeket megtanulni. Ezt a kompromisszumot a folyamatos tanulás és a régi emlékek védelme között Grossberg *stabilitás-plaszticitási dilemmának* nevezte. Ez különleges tervezési problémákat állít elénk, például a (felügyelt) előrecsatolt hálózatokban, amelyek napjainkban a legnépszerűbb neurális hálózatok. Mivel az új információk fokozatosan mossák el a régi információkat, ezért az előrecsatolt hálózatok nem válhatnak stabillá egy változó környezetben.

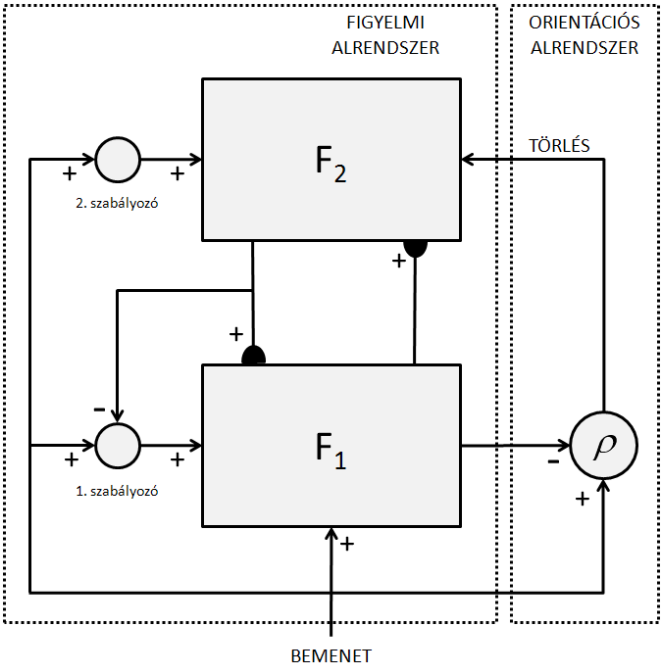
Annak érdekében, hogy a biológiai viselkedést utánozni tudjuk, az ART neurális hálózatok a nem felügyelt, önszervező tanulásra fektetik a hangsúlyt. A nem felügyelt tanulás azt jelenti, hogy a hálózat a fontos mintákat kizárólag a bemenet alapján tanulja meg, nem kap visszajelzést. Nincs egy kívülálló tanár, aki megmondja a hálózatnak, hogy egy bizonyos bemenet melyik kategóriához tartozik. A tanulás ezen kívül lehet még *megerősítő tanulás* vagy *felügyelt tanulás*. Megerősítő tanulás esetén a hálózat csak korlátozott visszajelzést kap, például „ennél a bemeneti mintánál jól teljesítettél” vagy „ennél a bemeneti mintánál hibáztál”. Felügyelt módban a hálózat minden bemenethez megkapja a helyes választ is. A biológiai rendszerekben a tanulás mindig a nem felügyelt tanulással kezdődik, hiszen egy újszülöttnél még nem létezhetnek kategóriák. Egy nem felügyelt tanulású rendszer bármikor átállítható úgy, hogy más módon tanuljon, mondjuk megerősítő vagy felügyelt tanulást végezzen, de egy kifejezetten felügyelt módú tanulásra tervezett rendszer sosem működhet nem felügyelt módban. Az önszabályozás azt jelenti, hogy a rendszer képes a felismerési kategóriákat stabilan, valós időben felépíteni. [9]

Azóta ezek a tervezési megszorítások sok, az elméletre épülő valós idejű neurális hálózat modelljét ösztönözték, amelyek felügyelten vagy felügyelet nélkül tanulnak, mintafelismerést vagy előrejelzéseket végeznek. Ebbe a családba tartozik az ART1, amely

egy tetszőleges sorrendben megadott bináris bemeneti vektort tud stabilan kategorizálni [10]; az ART2, amely már bináris és analóg bemeneti vektorokon is tud dolgozni [11] és az ART3, amely képes elosztott felismerési kódok párhuzamos keresését elvégezni egy többszintű hálózati hierarchiában [12]. A Fuzzy ART modell [1] a fuzzy logika számításain alapul, és egy átalakított ART1 modellt tartalmaz, hiszen ha a fuzzy változók bináris értékűek, akkor a fuzzy halmazelmélet műveletei a bináris műveletekhez vezetnek. Az említett nem felügyelt tanulású hálózatokon kívül, léteznek az ART-n alapuló felügyelt tanulású architektúrák, például az ARTMAP, amely egy vagy több fent említett nem felügyelt tanulású ART modulból áll [13]. [14]-[17]

2.2.2 Az ART rendszer felépítése

Az ART rendszer két fő komponense a *figyelmi* és az *orientációs alrendszer*. A figyelmi alrendszer többek között két, neuronokat tartalmazó mezőből áll, F_1-ből és F_2-ből, amelyek számos további réteget tartalmazhatnak. Ezeket a mezőket előrecsatolt és



3. ábra: Egy tipikus ART neurális hálózat blokkdiagramja. Az előfeldolgozás után a bemeneti aktivitási szintet transzformáljuk az első mezőre, F1-re. Az F1 mezőt az F2 mezővel előre- és visszacsatolt összeköttetések kapcsolják össze, melyeket félbevágott fekete ellipszisek jelölnek. Ezek az összeköttetések alkotják a rendszer hosszú távú memóriáját. *([9] 2. ábrája alapján)*

visszacsatolt összeköttetések kapcsolják egymáshoz. Ezeknek az összeköttetéseknek a súlyai alkotják a rendszer *hosszú távú memóriáját* (LTM – Long Term Memory), amelyekkel majd az összeköttetéseken átmenő jeleket meg kell szoroznunk. A *rövid távú memória* (STM – Short Term Memory) elnevezés a bemenet feldolgozása során az egyes mezőkön kialakuló aktivitási szintet fogja jelenteni. Az orientációs alrendszer feladata, hogy stabilizálja az STM feldolgozását és az LTM-ben lezajló tanulást. Ahogyan az ábrán is látható, az F_1 mező három forrásból kaphat információkat. Ez a három forrás: az alulról-felfelé irányított bemenet, az F_2-ből érkező felülről-lefelé jövő adatok, illetve a szabályozóból jövő jel. Annak megakadályozására, hogy az F_2-ből visszajövő jel valamilyen aktivitást generáljon az F_1-ben (tehát hogy megakadályozzuk azt, hogy a rendszer hallucináljon), a rendszer dinamikája úgy van kialakítva, hogy a három bemeneti forrásból legalább kettőnek aktívnak kell lennie ahhoz, hogy az F_1 mezőn bármiféle aktivitás keletkezzen. Ezt nevezik ART terminológiával a ⅔-*os szabálynak*. Ugyanez a szabály érvényes az F_2-be menő három bemeneti forrására is.

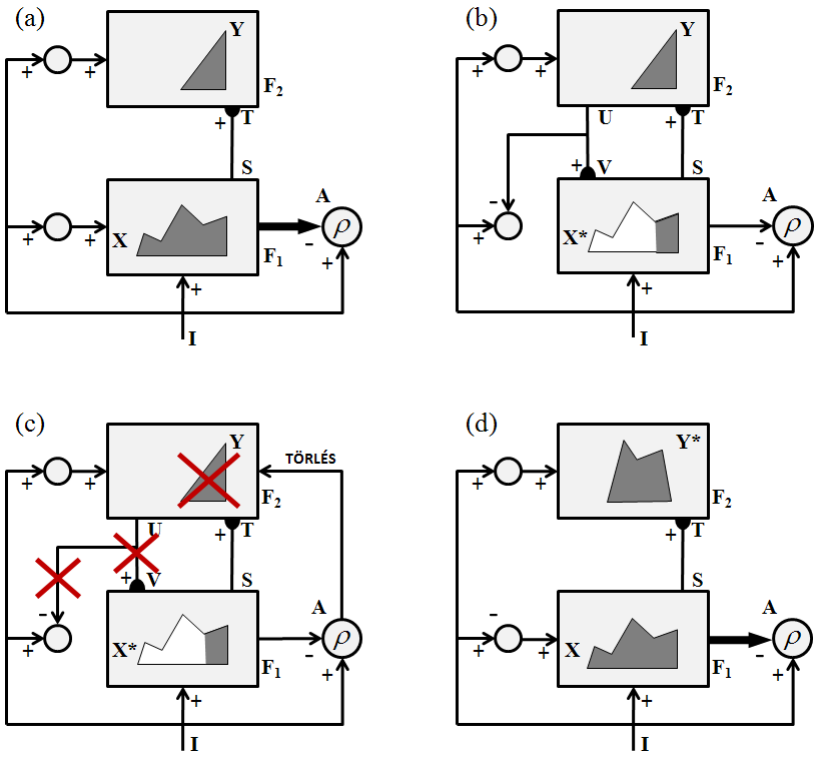
Amikor a rendszer bemenetére egy **I** vektor érkezik, akkor az F_1-rétegben lévő neuronok aktivitásában ez egy **X** aktivitási vektorként jelentkezik. Ezután az F_1 réteg kimenetén megjelenik egy **S** vektor, ami keresztülmegy az F_2-be vezető adaptív súlyokon (ezek a súlyok alkotják a rendszer „hosszú távú memóriáját"). Ennek eredményeként az F_2 rétegben megjelenik egy **T** bemeneti vektor, amiből az F_2 réteg előállít egy tömörített aktivitási vektort: **Y**-t, aminek csak egyetlen komponense lehet 1, mivel ez jelenti a kiválasztott kategóriát. A *győztes-mindent-visz* elv alapján azt a csomópontot választjuk ki, ahol az aktivációs függvény értéke a legnagyobb. Az **Y** vektornak ez a komponense lesz 1, a többi helyen 0. Az F_2-nek ezt az aktív egységét, tekinthetjük úgy, mint egy hipotézist, feltevést. Az **Y** vektor aktiválása generál egy **U** jelvektort, amely egy adaptív szűrőn keresztül visszakerül az F_1 rétegbe. Miután megszoroztuk az **U** vektort az adaptív súlyok mátrixával, ennek eredménye, egy **V** vektor kerül az F_1 réteg bemenetére. Ez a **V** vektor egyfajta elvárást jelent, és a bemeneti vektornak meg kell felelnie ennek az elvárásnak. Erre gondolhatunk úgy, hogy ez az előzőleg feltett hipotézis ellenőrzése, vagy másképpen az kiválasztott kategória prototípusának lekérdezése. Ez az illesztési eljárás módosíthatja az F_1 réteg aktivitását, az **X** vektort, olyan módon, hogy gátolja az **I** vektornak azon tulajdonságait, amelyet az F_2 rétegből érkező **V** vektor nem hagyott jóvá, nem igazolt. Ennek eredményeképpen létrejön egy **X*** vektor, ami a mintának azokat a tulajdonságait kódolja, amelyre a hálózat figyelmet fordít. Ha a **V** elvárás elég közel van az **I** bemenethez, akkor bekövetkezik a *rezonancia* állapota. Az

ART rendszer nem példákat tanul meg, hanem prototípusokat, mivel az **I** bemeneti vektor helyett az **X*** vektort tanulja meg, ami a mintának azon tulajdonságait tartalmazza, amelyre a rendszer felfigyelt.

Az összes ART rendszer sajátossága az a mintaillesztés, ami a lentről-felfelé irányított bemenet és a fentről-lefelé jövő megtanult prototípusvektor között történik. Az illesztés eredményeképpen vagy létrejön egy *rezonáns* állapot, ami a rendszer figyelmét összpontosítja és megkezdi a stabil prototípus megtanulását, vagy elindít egy önszabályozó párhuzamos keresést a memóriában. Ez a keresés kétféleképpen érhet véget. Ha egy kialakított kategóriát választottunk ki, akkor úgy finomítjuk annak prototípusát, hogy tükrözze a bemeneti mintából származó új információkat. Abban az esetben beszélhetünk *rezonanciáról*, amikor a bemenet illeszkedik valamelyik kialakított kategória prototípusára. Ez a rezonáns állapot elég sokáig tart ahhoz, hogy végbemenjen a tanulás, innen származik az adaptív rezonancia elnevezés. Másrészt, ha a keresés egy olyan csomópontnál ér véget, amelynél még nem játszódott le tanulás, akkor megkezdődik egy új kategória kialakítása. [1]

A hálózatot úgy tervezték, hogy a felhasználó szabályozni tudja az egy kategóriába kerülő minták közötti hasonlóságot. Ezt egy úgynevezett *vigilancia paraméterrel* érik el: ha a paraméter értéke alacsony, akkor a hálózat kevés, általános kategóriát hoz létre, míg magas vigilancia esetén sok, speciális kategória jön létre. A keresés során a találathoz szükséges hasonlóságot előbb említett *vigilancia* határozza meg. Ettől függ, hogy a bemenetnek mennyire kell hasonlítania a fentről-lefelé irányított prototípusra, ahhoz hogy bekövetkezzen a *rezonancia*. Mivel a *vigilancia paraméter* változhat a tanulási szakaszok során, ezért egy ART rendszer a tanulás során akár teljesen eltérő általánosítási fokozatokban is kódolhatja a mintákat. Az alacsony vigilancia széleskörű általánosításokat és absztrakt prototípusokat eredményez, ellentétben a magas vigilanciával, ahol viszont a prototípusok tanulása helyett a rendszer egyszerűen a bemeneten megjelenő példákat tanulja meg. A 4. ábra segítségével bemutatunk egy ART keresési ciklust. Ezen az ábrán nem látható az F_0 előfeldolgozó mező, ami mindössze a bemeneti minta normalizálását végzi.

(a) Az I bemeneti minta, miután átjutott az előfeldolgozást végző F_0 mezőn, az F_1 mező neuronjaiban valamilyen aktivitási szint képében jelentkezik. Itt teljesül a ⅔-os szabály, mivel az I bemenet az F_1 szintjén lévő szabályozót is aktiválja. A szabályozó aktivitása nem függ a minta értékétől, kizárólag attól, hogy van-e bemenő minta vagy nincs. Az X



4. ábra: (a) Megfelelő normalizálás után az I bemeneti minta az F1 mezőben kialakít egy X aktivitási szintet, és aktiválja az A orientációs alrendszert. A kialakult X minta gátolja A-t és generál egy S kimeneti mintát. Az S-et a súlyok egy T bemeneti mintává transzformálják, ami az F2 mezőben egy Y aktivitási szintként jelentkezik. (b) Az Y minta generál egy fentről-lefelé irányított jelet, az U-t, ami a súlyokon keresztüljutva egy prototípust leíró V mintává alakul. Ha a V minta F1-ben nem illeszkedik I-re, akkor F1-ben egy új X* aktivitási szint jön létre, ami lecsökkenti az A-t gátló hatást. (c) Ha az illeszkedés sikertelen, akkor az A orientációs alrendszer egy törlő jelet küld az F2 mezőnek, ami törli az Y mintát az F2 mezőből. (d) Az Y minta törlése miatt a fentről-lefelé irányított jel megszűnik, ezért az F1 mezőben visszaáll az X aktivitási szint, ami újra a T bemeneti mintát küldi F2-nek, de ezúttal F2-ben egy új Y* minta jön létre. Ha az illeszkedés most sem sikerül, akkor folytatódik a keresés az F2 egy megfelelő kódja után. ***([1], 761. oldal 2. ábrája alapján)***

aktivitási szint gátolja A-t és létrehozza az F_1 mező kimenő jelét, S-et. Az A csomópont gátlása azért szükséges, mert máskülönben aktiválná az F_2 mező felé a törlő jelet. Az S jelet megszorozzuk az alulról-felfelé irányított összeköttetések súlyaival, aminek eredménye az F_2 mező bemenő jele, a T. Ez a T jel az F_2 mezőben kialakít egy Y aktivitási szintet. Itt szintén teljesül a ⅔-os szabály, mivel az I bemenő jel az F_2 szintjén lévő szabályozót is aktiválta.

(b) Az Y minta generál egy U jelet, ami a fentről-lefelé irányított súlyokkal megszorozva, egy prototípust jelölő V mintát eredményez. Az F_1 rétegben megtörténik a V prototípus és az I bemeneti minta összehasonlítása. Ennek eredménye az F_1 mező új X* aktivitási szintje. Ha az illeszkedés rossz, akkor ez egy jelentősen lecsökkent X* aktivitási szintet eredményez. Ennek hatására lecsökken az A-t gátló jel erőssége is.

(c) Ha az A orientációs alrendszerben nem teljesül a vigilanciával szemben támasztott követelmény, akkor A az F_2 mezőnek egy olyan törlő jelet küld, ami a mező legaktívabb neuronjait gátolja. Ennek hatására az Y jel törlődik, és ezzel együtt a U válasz és a V prototípus is.

(d) Az F_1 mezőben visszaáll az X minta, F_2 mezőben pedig egy új Y* aktivitási szint alakul ki. Ha fentről-lefelé jövő prototípus ismét nem illeszkedik az I-re F_1-ben, akkor addig folytatódik a keresés, amíg egy olyan prototípust nem talál a rendszer, ami megfelel az A által támasztott követelményeknek, vagy ha nincs ilyen, akkor egy új kategóriát kell létrehozni a még nem tanított csomópontok segítségével. [9]

Minden ART rendszer középpontjában ez a mintaillesztési eljárás áll, amely a kívülről érkező bemenetet összehasonlítja a belső memóriájában tárolt információval. Ez az összehasonlítás vagy egy *rezonáns* állapothoz vezet, ami elég ideig áll fent ahhoz, hogy a tanulás végbemenjen, vagy elindít a memóriában egy keresést. Ha a keresés egy létező kategóriánál áll meg, akkor a memóriába beépülnek az aktuális bemenetből származó információk. Ha a keresés egy új kategóriánál áll meg, akkor a rendszer megtanulja az aktuális bemeneti mintát. Ez a *találat-alapú tanulási folyamat* az ART stabilitásának alapja. Így a rendszer memóriája kizárólag akkor változik meg, ha a kívülről származó bemeneti minta elég közel van a belső elvárásokhoz, vagy ha valami teljesen új minta érkezik. Ez a tulajdonság alkalmassá teszi az ART rendszereket olyan feladatokra, amelyekhez nagy és fejlődő adatbázisok megtanulása szükséges.

2.2.3 Az ART alapú hálózatok tanulása

Az ART alapú neurális hálózatok esetén a tanulásra két alapvető módszer áll rendelkezésünkre: a *lassú tanulás* és a *gyors tanulás*. *Lassú tanulás* esetén a tanulás mértékét differenciálegyenletekkel kiszámított folytonos értékek adják meg, melyek függenek az bemeneti vektor, mint inger időtartamától. Ez a tanulási módszer biológiailag jobban megfelel a valóságnak, és emellett használható folytonos idejű hálózatoknál is, ahol a bemeneti vektor időben folytonosan változhat. *Gyors tanulás* esetén a súlyok változása leírható algebrai egyenletekkel, mivel feltételezzük, hogy a tanulás az inger időtartamához képest gyorsan, elhanyagolható időn belül lezajlik. Itt a bemeneti értékekre adott válaszok eredményeképpen az adaptív súlyok egy egyensúlyi állapothoz tartanak. A *gyors tanulás* lehetővé teszi, hogy a

rendszer gyorsan alkalmazkodjon az olyan mintákhoz, amelyek ritkán fordulnak elő, de esetleg azonnal, pontosan kell őket előhívni a rendszer memóriájából.

2.3. A Fuzzy ART

Ennek a résznek a középpontjában az adaptív rezonancia elméletére épülő neurális hálózatok egyik típusa, a Fuzzy ART áll. Egy rövid áttekintést követően bemutatjuk a hálózat működésének algoritmusát.

2.3.1 A Fuzzy ART áttekintése

A Fuzzy ART az ART2 hálózatot ötvözi a fuzzy halmazelmélet eredményeivel, egy olyan fuzzy neurális hálózatot eredményezve, amely képes folytonos értékű vektorok gyors és stabil osztályozására [1]. A gyors jelző arra utal, hogy a rendszer a *gyors tanulás* paradigmáját használja, míg a *stabilitás* arra vonatkozik, hogy az adaptív súlyok idővel csak csökkenhetnek, ezért a rendszer előbb-utóbb elér egy egyensúlyi, stabil állapotba.

Az Fuzzy ART hálózatok hatékonyságát két módszerrel szokás növelni, ez a *gyors tanulás, lassú felejtés* technikája és a *komplemens kódolás*. Amikor egy eddig teljesen újszerű, sajátos minta érkezik a rendszerbe, amelyik egyik kategóriába sem sorolható, akkor a rendszer egy új kategóriát hoz létre. A gyors tanulás, lassú felejtés módszer azt jelenti, hogy ilyenkor a tanulási paramétert 1-re állítjuk, és a mintát így kódoljuk a kategória súlyaiba. Ezzel a hálózat képessé válik az új események pontos megtanulására, viszont a már kialakult kategóriákat nem teszik tönkre a zajos bemeneti mintákból származó hibák, mivel ezeknek a tanulása egy kisebb tanulási paraméterrel történik. Azt ART rendszereknél problémát jelenthet a kategóriák mértéktelen elszaporodása. Ennek kiküszöbölésére bevezették a bemeneti adatok normalizálását. Ezt valósítja meg a komplemens kódolás, mégpedig úgy, hogy megőrzi a bemeneti vektor amplitúdóját, vagyis a komponensek értékei közötti különbséget. Ez úgy valósítható meg, hogy **I** bemeneti vektort a következő alakban kódoljuk:

$$I = (a, a^c) = (a_1, \ldots, a_n, 1 - a_1, \ldots, 1 - a_n)$$

Vagyis az eredeti **I** bemeneti vektorhoz hozzáfűzzük a komplemensét, és így nem csak az egyes tulajdonságok jelenlétét tudjuk a vektorban jelölni, hanem a komplemens részben az egyes tulajdonságok hiányát is.

2.3.2 A Fuzzy ART algoritmusa

A rendszer bemeneti mintáját egy n-dimenziós vektor adja, ami a következőképpen írható fel:

$$(I_1, \ldots, I_N), \text{ ahol } I_i \in [0, 1], i = 1, \ldots, N$$

Ezt a vektort **I**-vel fogjuk jelölni, elemei 0 és 1 közötti valós számok. A rendszer minden (kezdetben üres) kategóriájához tartozik egy

$$w_j = (w_{j1}, \ldots, w_{jN})$$

súlyvektor, ahol j jelöli a kategória sorszámát. Tetszőleges számú kategória létrehozható, jelölje most a kategóriák maximális számát M, ennek megfelelően j = 1, …, M. Kezdetben

$$w_{j1} = \ldots = w_{jN} = 1$$

és minden kategóriára azt mondjuk, hogy *szabad*. Miután egy kategóriát kódolásra kiválasztunk *foglalttá* válik.

A Fuzzy ART működését az alábbi paraméterek határozzák meg:

$\alpha > 0$, a *választási paraméter*

$\beta \in [0, 1]$ *tanulási paraméter,* ami a tanulási sebességet szabályozza

$\rho \in [0, 1]$ *vigilancia paraméter*, ami a létrejövő kategóriák számát és finomságát szabályozza

1. lépés: Tegyük fel, hogy a rendszerben még nincsenek kategóriák és a bemenetre egy minta érkezik. Ez a minta úgy jelenik meg az F_1 mezőben, mint a neuronok aktivitási szintje.

Első lépésben aktiválnunk kell az F_2 mező neuronjait. Ekkor a T_j választási függvényt minden I bemenetre és j kategóriára a következőképpen definiáljuk:

$$T_j(I) = \frac{|I \wedge w_j|}{\alpha + |w_j|}$$

ahol a $\wedge$ a fuzzy ÉS operátort jelenti, amelyet a következőképpen számolunk:

$$(\mathrm{x} \wedge \mathrm{y})_\mathrm{i} = \min(\mathrm{x_i}, \mathrm{y_i})$$

A képletben szereplő fuzzy $|\cdot|$ normát pedig az alábbi egyenletből kapjuk:

$$|x| = \sum_{i=1}^{N} |x_i|$$

Tehát ez annyit jelent, hogy egy ciklussal végig kell menni az összes kategórián és kiszámolni a választási függvény értékét, amelyet egy T_j vektorban fogunk tárolni, ahol j = 1, …, M.

2. lépés: A második lépésben kiválasztjuk a F_2 mező neuronjai közül azt amelyiknek a legnagyobb az aktivitása, tehát itt érvényesül a győztes-mindent-visz szabály. A leginkább aktív neuron fogja a kiválasztott kategóriát jelképezni. Ezt a kiválasztott kategóriát J-vel jelöljük. Azt a kategóriát fogjuk kiválasztani, amire teljesül az alábbi egyenlet:

$$T_J = \max \{ T_j : j = 1, \dots M \}$$

Abban az esetben, ha több maximális T_j is létezik, akkor azt választjuk, amelyiknek a legkisebb az indexe.

3. lépés: A harmadik lépésben megvizsgáljuk, hogy a rendszer rezonál-e. Ha igen, akkor ez azt jelenti, hogy a minta eléggé hasonló a kiválasztott kategóriához, ekkor létrejöhet a tanulás. Ha a minta nem hasonlít eléggé, akkor a rendszer megpróbál egy jobb

kategóriát keresni, vagy ha egyik sem elég jó, akkor létrehoz egy újat. Azt mondjuk, hogy a rendszer *rezonál*, ha a kiválasztott kategória értékelő függvénye megfelel a vigilancia kritériumának, vagyis:

$$\frac{|I \wedge w_J|}{|I|} \geq \rho$$

Ekkor megkezdődik a tanulás, amit lentebb tárgyalunk. Ellenkező esetben, vagyis ha

$$\frac{|I \wedge w_J|}{|I|} < \rho$$

akkor a T_J értékét a bemenet feldolgozásának idejére –1-re állítjuk (egyszóval töröljük), és ezzel a keresés során megakadályozzuk az ismételt kiválasztását. A T_J törlése után a 2. lépéssel folytatjuk az algoritmust.

4. lépés: Amennyiben bekövetkezett a rezonancia vagy kiválasztottunk tanulásra egy új kategóriát, akkor végbemegy a súlyok módosulása. A tanulás során a w_J súlyvektort az alábbi összefüggés alapján frissítjük:

$$w_j^{(\text{új})} = \beta\left(I \wedge w_J^{(\text{régi})}\right) + (1 - \beta)w_J^{(\text{régi})}$$

Gyors tanulás, lassú felejtés: A zajos minták hatékony tanulása érdekében érdemes a *tanulási paramétert* 1-re állítani, ha a J kategória nem foglalt, és kisebb, mint 1-re állítani, ha már foglalt. Ekkor a nem foglalt kategóriák esetében a tanulás megegyezik a következő összefüggéssel:

$$w_j^{(\text{új})} = I \wedge w_J^{(\text{régi})}$$

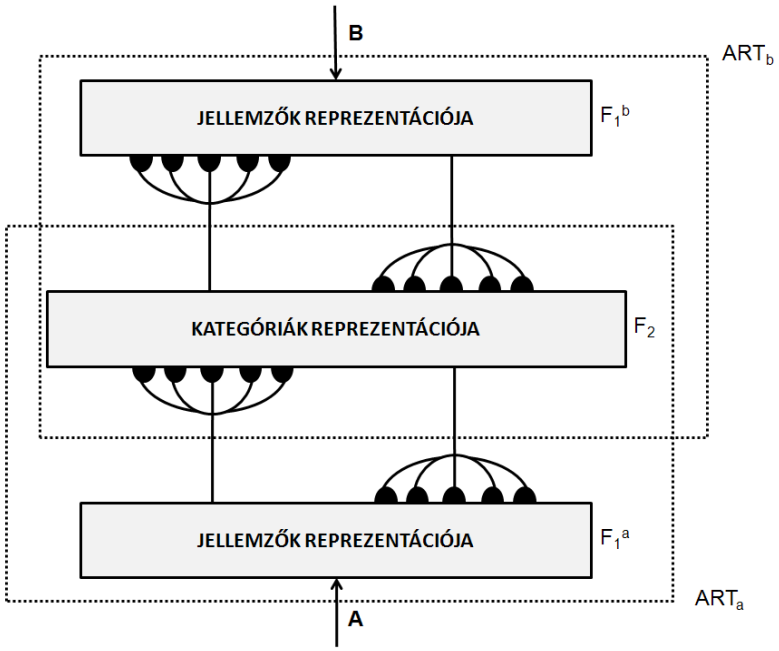
[1]

2.4. A Fuzzy ARAM

Az előző részben ismertetett Fuzzy ART neurális hálózatot felhasználva egy olyan asszociatív neurális hálózatot ismertetünk, amely két ART modulból áll. Rövid áttekintés után rátérünk a hálózat működését leíró algoritmus bemutatására.

2.4.1 A Fuzzy ARAM áttekintése

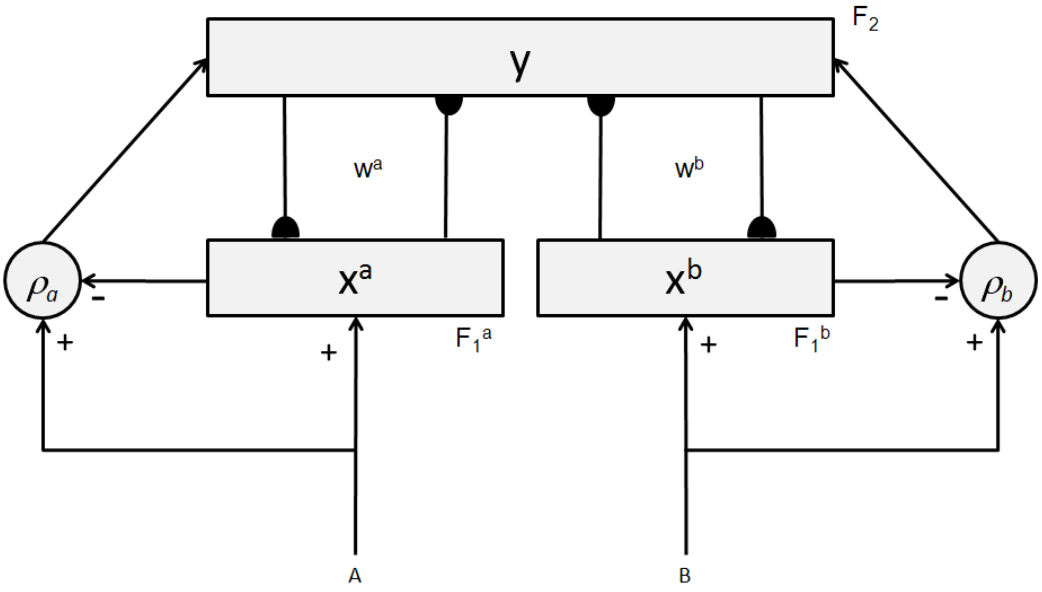
Az ARAM (Adaptive Resonance Associative Map) az adaptív rezonancia elméletére épülő gyors, stabil, heteroasszociatív tanulású rendszer [2]. Látványosan úgy képzelhető el, mint két egymással átfedésben lévő ART rendszer, amelyeknek a kategóriákat tartalmazó mezője közös. Bár az ARAM hálózat architektúrája egyszerűbb, mint a felügyelt tanulású ARTMAP modellé, ennek ellenére a két rendszer ugyanolyan osztályozási képességekkel és teljesítménnyel rendelkezik. Mivel az ARAM hálózati struktúrája és műveletei szimmetrikusak, ezért az asszociatív adatok visszakeresése mindkét irányban megoldható. Maximális vigilancia beállítás mellett, az ARAM hálózat a megtanult mintapárokat veszteség nélkül tudja visszaadni.



5. ábra: Egy Fuzzy ARAM rendszer sematikus felépítése, látszik a két ART modul, az ART_a és az ART_b, amelyek külön-külön végzik a jellemvonások reprezentációját, valamint a kategóriákat reprezentáló közös F2 mező. ([2] 2. ábrája alapján)

Az ARAM hálózat két, egymástól némileg eltérő feladatot végez, a minták osztályozását és a heteroasszociatív emlékek előhívását. A minták osztályozásához a rendszer 1:N hozzárendelést végez a minták halmazából a minták osztályaiba. Bár az ARAM szerkezetileg egyszerűbb mint az ARTMAP hálózatok, ennek ellenére a meghatározott paraméter-értékek esetén ugyanolyan teljesítményt nyújt.

A rendszer két ART modulból áll, amelyek átfedik egymást. A két ART modult A-val és B-vel fogjuk jelölni. A jellemvonásokat tartalmazó F_1 mezőkre pedig az F_1^a és F_1^b jelölést használjuk. A kategóriákat tartalmazó F_2 mezőn a két ART modul közösen osztozik.



6. ábra: Az ARAM neurális hálózat architektúrája ([2] 5. ábrája alapján)

2.4.2 A Fuzzy ARAM algoritmusa

A bemenő adatokat célszerű normalizálni, mielőtt a rendszerbe kerülnek, ezzel megakadályozható a felesleges kategóriák kialakulása. Erre a legalkalmasabb a komplemens kódolás, amely a normalizálás során nem változtatja meg a bemeneti vektor elemei közötti különbségeket. A komplemens kódolást alkalmazva az F_1^a mező egy A bemeneti mintája a következőképpen néz ki:

$$A = (a, a^c) = (a_1, \dots, a_M, a_1^c, \dots, a_M^c)$$

ahol $a_i^c = 1 - a_i$. Hasonlóan, az F_1^b mező bemeneti mintája egy

$$B = (b, b^c) = (b_1, \dots, b_N, b_1^c, \dots, b_N^c)$$

vektor, ahol $b_i^c = 1 - b_i$. Ezen kívül jelölje x^a és x^b az F_1^a és az F_1^b mező aktivitási vektorait. Jelölje y az F_2 aktivitási vektorát. Az F_2 mező minden kategóriát reprezentáló csomópontja két adaptív súllyal van összeköttetésben, a w_j^a és a w_j^b súlyokkal. Kezdetben minden kategória *szabad* és minden súly 1-gyel egyenlő. Miután egy kategóriát kódolásra kiválasztunk *foglalttá* válik.

A Fuzzy ARAM működését az alábbi paraméterek határozzák meg:

$\alpha > 0$, a *választási paraméter*

$\beta \in [0, 1]$, a *tanulási paraméter,* ami a tanulási sebességet szabályozza

$\rho \in [0, 1]$, a *vigilancia paraméter*, ami a létrejövő kategóriák számát és finomságát szabályozza

$\gamma \in [0, 1]$, a *részesedési paraméter*, ami a két ART modul közötti egyensúlyt szabályozza

1. lépés: Tegyük fel, hogy a rendszerben még nincsenek kategóriák és a bemenetre egy A, B mintapár érkezik. Az A minta kerül az F_1^a réteg bemenetére, a B minta az F_1^a réteg bemenetére, és ez ott a neuronok valamilyen aktivitási szintjeként jelentkezik. Most aktiválnunk kell az F_2 mező neuronjait, amit a T_j választási függvényt segítségével teszünk meg:

$$T_j(I) = \gamma \frac{|A \wedge w_j^a|}{\alpha_a + |w_j^a|} + (1 - \gamma) \frac{|B \wedge w_j^b|}{\alpha_b + |w_j^b|}$$

ahol a $\wedge$ a fuzzy ÉS operátort jelenti, amelyet a következőképpen számolunk:

$$(x \wedge y)_i = \min(x_i, y_i)$$

A képletben szereplő fuzzy |·| normát pedig az alábbi egyenletből kapjuk:

$$|x| = \sum_{i=1}^{N} |x_i|$$

2. lépés: A második lépésben kiválasztjuk a F_2 mező neuronjai közül azt amelyiknek a legnagyobb az aktivitása, tehát itt érvényesül a *győztes-mindent-visz* szabály. A leginkább aktív neuron fogja a kiválasztott kategóriát jelképezni. Ezt a kiválasztott kategóriát J-vel jelöljük. Azt a kategóriát fogjuk kiválasztani, amire teljesül az alábbi egyenlet:

$$T_J = \max \{ T_j : j = 1, \dots M \}$$

Abban az esetben, ha több maximális T_j is létezik, akkor azt választjuk, amelyiknek a legkisebb az indexe.

3. lépés: A harmadik lépésben megvizsgáljuk, hogy a rendszer rezonál-e. A rendszer stabilitása miatt az egész rendszer csak akkor fog rezonálni, ha mind az ART_a modul, mind az ART_b rezonál. Abban az esetben, ha nincs rezonancia, akkor folytatódik a keresés egy jobb mintapár után, vagy létrehoz egy új kategóriát, amelybe elhelyezi a bemeneti mintapárt. Azt mondjuk, hogy a rendszer *rezonál*, ha az m_J^a és m_J^b találatfüggvények megfelelnek a vigilancia kritériumának, vagyis:

$$\mathrm{m_J^a} = \frac{|\mathrm{A} \wedge \mathrm{w_J^a}|}{|\mathrm{A}|} \geq \rho_a \quad \text{és} \quad \mathrm{m_J^b} = \frac{|\mathrm{B} \wedge \mathrm{w_J^b}|}{|\mathrm{B}|} \geq \rho_b$$

Ekkor megkezdődik a tanulás, amit lentebb tárgyalunk. Ellenkező esetben a T_J értékét a bemenet feldolgozásának idejére –1-re állítjuk (egyszóval töröljük), és ezzel a keresés során megakadályozzuk az ismételt kiválasztását. A T_J törlése után a 2. lépéssel folytatjuk az algoritmust.

4. lépés: Amennyiben bekövetkezett a rezonancia vagy kiválasztottunk tanulásra egy új kategóriát, akkor végbemegy a súlyok módosulása. A tanulás során a w_J^a és w_J^b súlyvektort az alábbi összefüggések alapján frissítjük:

$$w_J^{a(új)} = \beta_a \left(A \wedge w_J^{a(régi)} \right) + (1 - \beta_a) w_J^{a(régi)}$$

$$w_J^{b(új)} = \beta_b \left(B \wedge w_J^{b(régi)} \right) + (1 - \beta_b) w_J^{b(régi)}$$

A zajos minták hatékony tanulása érdekében érdemes a β_a és β_b *tanulási paramétert* 1-re állítani, ha a J kategória szabad, és $\beta_a < 1$, $\beta_b < 1$-re állítani, ha már foglalt.

Találatkövetés: Az ARTMAP neurális hálózatban is használt találatkövetési szabály a kódtömörítés maximalizálásához hasznos. Minden inputnál a ρ_a vigilancia paraméter megegyezik a $\overline{\rho_a}$ alapvigilancia értékével. Ha az F_2 kategóriamezőben törlődik egy csomópont, akkor a ρ_a értékét addig csökkentjük, amíg éppen nagyob nem lesz az m_J^a találatfüggvény értékénél. Innentől a keresés a felülbírált vigilanciával folytatódik. [2]

3. A webalkalmazás

A webalkalmazás célja annak szimulációja, hogy az emberi agy hogyan reagál a külvilágból érkező különböző üzenetekre. Ez a fejezet a program részletes ismertetésén kívül tartalmazza a program felhasználói és adminisztrátori dokumentációját, valamint egy referenciaként használható fejlesztői leírást is.

3.1 A program működése

Az alkalmazás a számítások elvégzéséhez a Survey ART neurális hálózat modellt használja, amely egy kognitív információfeldolgozásra létrehozott mesterséges modell, amely Fuzzy ART és Fuzzy ARAM neurális hálózatokat használ. [3]

3.1.1 A Survey ART áttekintése

A Survey ART modell ún. *alrendszerekből* áll, amelyek között Fuzzy ART és Fuzzy ARAM neurális hálózatok biztosítják a kapcsolatot. Minden alrendszer egy S = (t, n, m) hármasként írható le, ahol *t* az alrendszer típusa, *n* jelenti a neurális hálózat prototípusaiban a *jellemvonások* számát (vagyis a prototípusvektor elemeinek számát), az *m* paraméter pedig a hálózatban a kialakítható kategóriák maximális száma. Minden alrendszer a következő típusok közül pontosan egybe tartozik: *általános*, *értékelő*, *döntési* alrendszer. Ezeknek a szerepéről a későbbiekben lesz szó.

Minden S_i alrendszerhez (amelynek a paraméterei n_i, m_i), tartozik egy Fuzzy ART neurális hálózat n_i, m_i paraméterekkel, tehát ebben a Fuzzy ART hálóban a prototípusvektorok n_i dimenziósak, és a hálózat maximálisan m_i kategóriát képes kialakítani. Továbbá tegyük fel, hogy S_i és S_j tetszőleges típusú Survey ART alrendszerek, a hozzá tartozó paraméterek pedig m_i, n_i és m_j, n_j. Ekkor minden ilyen párhoz (ahol az (S_i, S_j) pár ugyanaz, mint az (S_j, S_i) pár) tartozik egy asszociatív Fuzzy ARAM neurális hálózat, amelynek a két ART modulja m_i, n_i valamint m_j, n_j paraméterekkel rendelkezik. Tehát összefoglalva ez annyit jelent, hogy ha az alrendszerek száma *k*, akkor *k* db Fuzzy ART

neurális hálózat és $\frac{k(k+1)}{2}$ db Fuzzy ARAM neurális hálózat alakul ki. Ezen felül minden ART hálózat saját *vigilancia*, *tanulási* és *választási* paraméterrel rendelkezik, ARAM hálózatok esetén pedig mindkét ART modulhoz tartozik egy-egy az előbbi paraméterekből, továbbá az ARAM hálózat rendelkezik egy *részesedési* paraméterrel is, amely tanulásnál a két ART modul súlyát határozza meg.

A Survey ART modellben az alrendszerekhez tartozó ART hálózatok, valamint az alrendszerpárokhoz tartozó ARAM hálózatokban lévő ART modulok a *komplemens kódolást* nem alkalmazzák, viszont tanulásnál a *gyors tanulás, lassú felejtés* technikáját használják, vagyis üres kategória kiválasztása esetén a tanulási paraméter automatikusan 1-re áll. Az ARAM hálózatokban nincsen *találatkövetés*.

3.1.2 Tanulás

A Survey ART hálózat az m = (t, r, A, B) formájú üzeneteket képes megtanulni, ahol *t* az üzenet típusa, amely lehet *egyszerű üzenet* vagy *ok-okozati összefüggést kifejező üzenet.* Az *r* jelöli az üzenet forrásának megbízhatóságát, lehet: *megbízható*, *semleges* vagy *nem megbízható*. A *t* és az *r* a későbbiekben a tanulás paramétereit fogja befolyásolni. Az üzenet leglényegesebb része az *A* és *B* részüzenet. Minden részüzenet két adatot tartalmaz, egyrészt megjelöl egy alrendszert a Survey ART modellben, amelynek továbbítani kell ezt a részüzenetet, és tartalmazza az üzenetet hordozó kódvektort, amelynek dimenziója meg kell hogy egyezzen az alrendszer többi prototípusvektorának dimenziójával. Jelöljük az A részüzenethez tartozó alrendszert S_a-val, a hozzá tartozó vektort pedig c_a-val, a B részüzenethez tartozó alrendszert és vektort pedig S_b-vel és c_b-vel. Ezeknek a paramétereknek az ismeretében a tanulás a következőképpen zajlik:

1. Az *m* üzenetből meghatározzuk, hogy a részüzenetek melyik két alrendszert érintik. Ezek a fenti jelölést használva az S_a és S_b alrendszerek lesznek.
2. Az S_a alrendszerhez tartozó Fuzzy ART neurális hálóhoz továbbítjuk az első részüzenet kódvektorát, c_a-t. A neurális hálózat ezt a vektort kategorizálja (vagyis besorolja valamelyik létező kategóriába, vagy létrehoz számára egy teljesen új kategóriát) és megadja a kategória prototípusát p_a-t. Ennek a Fuzzy ART hálózatnak a

vigilancia, tanulási és választási paramétereit a Survey ART modellben az S_a alrendszer ART hálózatához megadott alapértékek alkotják.

3. Az S_b alrendszerhez tartozó ART neurális hálóhoz továbbítjuk a második részüzenet kódvektorát, c_b-t. A neurális hálózat ezt a vektort is kategorizálja és megadja a kategória prototípusát p_b-t. A Fuzzy ART hálózat vigilancia, tanulási és választási paramétereit itt is az alapértékek határozzák meg.
4. Ezután a p_a és p_b prototípusvektorokat továbbítjuk az S_a és S_b alrendszer közötti asszociációkat kezelő Fuzzy ARAM neurális hálózatnak, ahol paraméterek értékét a következő eljárással határozzuk meg: Jelölje ennek az ARAM neurális hálózatnak a két ART moduljához tartozó vigilancia és tanulási paramétereket ρ_a, β_a, és ρ_b, β_b. Ezeket a paramétereket az üzenet részeként megadott *t* és *r* a következőképpen befolyásolja:

   [1/1] Ha az üzenet *egyszerű üzenet* és az üzenet forrása *megbízható*, akkor a tanulási paraméter mindkét részüzenet esetén az alapértékhez képest nő:

   $\rho_a' = \rho_a$ $\qquad$ $\rho_b' = \rho_b$

   $\beta_a' = \beta_a + \frac{1-\beta_a}{3}$ $\qquad$ $\beta_b' = \beta_b + \frac{1-\beta_b}{3}$

   [1/2] Ha az üzenet *egyszerű üzenet* és az üzenet forrása *semleges*, akkor a paraméterek értéke nem változik meg:

   $\rho_a' = \rho_a$ $\qquad$ $\rho_b' = \rho_b$

   $\beta_a' = \beta_a$ $\qquad$ $\beta_b' = \beta_b$

   [1/3] Ha az üzenet *egyszerű üzenet* és az üzenet forrása *nem megbízható*, akkor a tanulási paraméter mindkét részüzenet esetén az alapértékhez képest csökken:

   $\rho_a' = \rho_a$ $\qquad$ $\rho_b' = \rho_b$

   $\beta_a' = \beta_a - \frac{\beta_a}{3}$ $\qquad$ $\beta_b' = \beta_b - \frac{\beta_b}{3}$

   [2/1] Ha az üzenet *ok-okozati összefüggést kifejező üzenet* és az üzenet forrása *megbízható*, akkor az első részüzenet esetén a vigilancia és a tanulási paraméter nő, a második részüzenet esetén a paraméterek nem változnak:

$$\rho_a' = \rho_a + \frac{1-\rho_a}{3} \qquad \rho_b' = \rho_b$$

$$\beta_a' = \beta_a + \frac{1-\beta_a}{3} \qquad \beta_b' = \beta_b$$

[2/2] Ha az üzenet *ok-okozati összefüggést kifejező üzenet* és az üzenet forrása *semleges*, akkor az első részüzenet esetén a vigilancia nő, a második részüzenet esetén a paraméterek nem változnak:

$$\rho_a' = \rho_a + \frac{1-\rho_a}{3} \qquad \rho_b' = \rho_b$$

$$\beta_a' = \beta_a \qquad \beta_b' = \beta_b$$

[2/3] Ha az üzenet *ok-okozati összefüggést kifejező üzenet* és az üzenet forrása *nem megbízható*, akkor az első részüzenet esetén a vigilancia nő a tanulási paraméter pedig csökken, a második részüzenet esetén a paraméterek nem változnak:

$$\rho_a' = \rho_a + \frac{1-\rho_a}{3} \qquad \rho_b' = \rho_b$$

$$\beta_a' = \beta_a - \frac{\beta_a}{3} \qquad \beta_b' = \beta_b$$

5. Ezután az S_a és S_b alrendszer közötti ARAM hálózat a módosított paramétereket figyelembe véve megtanulja és kódolja a prototípusok közötti asszociációt, vagyis (p_a, p_b)-t.

3.1.3 Kikérdezés

A Survey ART neurális hálózathoz intézett kérdés egy *c* kódvektorból áll, amelyet valamelyik *S* alrendszerhez intézünk. Továbbá kiválasztunk két alrendszert, amelyek közül az egyiket értékelésre, a másikat pedig a végső válasz megadására (vagyis döntésre) fogunk használni. Jelöljük az *értékelő* alrendszert S_e-vel (*evaluation subsystem*) a *döntési* alrendszert S_c-vel (*choice subsystem*).

1. A *c* kódvektor átmegy az adott *S* alrendszerhez tartozó ART hálózaton, ahol a hálózat kategorizálja és megmondja, hogy mi ennek a kategóriának a prototípusa (ilyenkor tanulás egyáltalán nem történik). Legyen ez a prototípus *p*.

2. Megvizsgáljuk, hogy van-e közvetlen kapcsolat a *p* prototípus és az S_c *döntési* alrendszer valamelyik kategóriája között. Ezt úgy tehetjük meg, hogy az S és S_c alrendszerek közötti kapcsolatot tároló ARAM hálózatban megnézzük, hogy van-e olyan kategória, amely a *p* prototípusra asszociál. Amennyiben van, abban az esetben ez lesz a Survey ART-hoz intézett kérdésre a válasz, tehát megállhatunk. Ha a döntési alrendszerben nincs megfelelő asszociáció a *p* prototípusra vonatkozóan, akkor a következő lépéssel folytatjuk.
3. Sorban az összes *általános* típusú alrendszerben megpróbálunk asszociációkat keresni a *p* prototípusra. A válaszokat az A halmaz jelöli. Ha $a_i \in A$, akkor ez azt jelenti, hogy az S_i alrendszerrel sikerül asszociációt találni, és a *p*-re ebben az alrendszerben az a_i prototípusvektor asszociált. Ha $a_j \notin A$, akkor az S_j alrendszer nem adott választ, tehát az ARAM neurális hálózatban nincs kapcsolódó kategória.
4. Az *általános* alrendszerek válaszát továbbítjuk az S_e *értékelő* alrendszerbe, ahol szintén asszociációkat keresünk. Jelölje az értékelő alrendszer által adott asszociációkat az E halmaz. Ha $e_i \in E$, akkor az értékelő alrendszer az S_i alrendszerből származó a_i prototípusra az e_i értékelést adta, ellenkező esetben nem adott választ.
5. Az értékeléseket tartalmazó E halmaz elemeit átlagoljuk, és ez a kombinált aktivációs vektor továbbmegy az S_c *döntési* alrendszerbe, ami megpróbál megfelelő asszociációt találni, és amennyiben talál, ez lesz a feltett kérdésre a végső válasz.

3.1.4 A példaadatbázis

Az alkalmazás adatbázisába felvittünk bizonyos példaadatokat, amelyen a működést szemléltetni fogjuk. Ezek a példa alrendszerek a következők:

#	alrendszer megnevezése	
1.	different groups of people	különböző embercsoportok
2.	financial consequences	pénzügyi következmények
3.	employment consequences	foglalkoztatási következmények
4.	cultural consequences	kulturális következmények
5.	safety consequences	biztonsági következmények
6.	descriptions	jellemzések
7.	persons	személyek
8.	evaluations	értékelések
9.	choices	döntések

Az egyes alrendszerekhez tartozó kezdeti kategóriák pedig az alábbiak:

#1	different groups of people (különböző embercsoportok)		
	a kategória szöveges reprezentációja		a kategóriához tartozó kódvektor
1.	immigrants	bevándorlók	[1,1,0,0,0,0,0,0,0,0]
2.	immigrants from Europe	európai bevándorlók	[1,1,0,0,0,0,0,0,1,0,0]
3.	immigrants from Africa	afrikai bevándorlók	[1,1,0,0,0,0,0,0,0,1,0]
4.	refugees	menekültek	[1,0,1,0,0,0,0,0,0,0]
5.	economic refugees	gazdasági menekültek	[1,0,1,0,0,1,0,0,0,0]
6.	political refugees	politikai menekültek	[1,0,1,0,0,0,1,0,0,0]
7.	foreigners	külföldiek	[1,0,0,1,0,0,0,0,0,0]
8.	tourists	turisták	[1,0,0,1,1,0,0,0,0,0]

#2	financial consequences (pénzügyi következmények)		
	a kategória szöveges reprezentációja		a kategóriához tartozó kódvektor
1.	income reductions	bevételcsökkenés	[1,1,0,0,0,0,0,0,0,0]
2.	pension reductions	nyugdíjcsökkenés	[1,1,1,0,0,0,0,0,0,0]
3.	more tax revenues	több adóbevétel	[1,0,0,0,0,1,1,0,0,0]

#3	employment consequences (foglalkoztatási következmények)		
	a kategória szöveges reprezentációja		a kategóriához tartozó kódvektor
1.	decrease of jobs	munkahelyek csökkenése	[1,1,1,0,0,0,0,0,0,0]
2.	new jobs	új munkahelyek	[1,0,0,0,0,0,0,0,1,1]

#4	cultural consequences (kulturális következmények)		
	a kategória szöveges reprezentációja		a kategóriához tartozó kódvektor
1.	destruction of the culture	a kultúra pusztulása	[1,1,1,0,0,0,0,0,0,0]
2.	destruction of the educational system	az oktatási rendszer leromlása	[1,1,0,1,0,0,0,0,0,0]
3.	enrichment of the culture	a kultúra gazdagodása	[1,0,0,0,0,0,1,1,0,0]

#5	safety consequences (biztonsági következmények)		
	a kategória szöveges reprezentációja		a kategóriához tartozó kódvektor
1.	more crimes	magasabb bűnözés	[1,1,1,0,0,0,0,0,0,0]
2.	more riots	több tüntetés/lázongás	[1,1,0,1,0,0,0,0,0,0]

#6	descriptions (jellemzések)		
	a kategória szöveges reprezentációja		a kategóriához tartozó kódvektor
1.	strange	furcsa	[1,1,1,0,0,0,0,0,0,0]
2.	misbehaving	rosszul viselkedő	[1,1,0,1,0,0,0,0,0,0]
3.	dangerous	veszélyes	[1,1,0,0,1,0,0,0,0,0]
4.	friendly	barátságos	[1,0,0,0,0,1,1,0,0,0]

#7	persons (személyek)		
	a kategória szöveges reprezentációja		a kategóriához tartozó kódvektor
1.	I	én	[1,1,1,0,0,0,0,0,0,1]
2.	he	ő	[1,0,0,1,1,0,0,0,0,1]
3.	politician	politikus	[1,0,0,0,0,1,1,0,0,0]
4.	scientist	tudós	[1,0,0,0,0,0,0,1,1,0]

#8	evaluations (értékelések)		
	a kategória szöveges reprezentációja		a kategóriához tartozó kódvektor
1.	good	jó	[1,1,1,0,0,0,0,0,0,0]
2.	useful	hasznos	[1,1,0,1,0,0,0,0,0,0]

3.	advantageous	előnyös	[1,1,0,0,1,0,0,0,0,0]
4.	bad	rossz	[1,0,0,0,0,1,1,0,0,0]
5.	useless	haszontalan	[1,0,0,0,0,1,0,1,0,0]
6.	disadvantageous	hátrányos	[1,0,0,0,0,1,0,0,1,0]

#9	choices (döntések)		
	a kategória szöveges reprezentációja		a kategóriához tartozó kódvektor
1.	stop	megállít	[1,1,1,0,0,0,0,0,0,0]
2.	not allow	nem enged	[1,1,0,1,0,0,0,0,0,0]
3.	allow	enged	[1,0,0,0,0,1,1,0,0,0]
4.	promote	támogat	[1,0,0,0,0,1,0,1,0,0]

Kezdetben két üzenetcsoport van a rendszerben:

1. üzenetcsoport: `Background knowledge` (háttérismeretek)

#	Ar.	Bal oldali részüzenet	Ar.	Jobb oldali részüzenet
1.	#2	income reductions	#8	bad
2.	#2	pension reductions	#8	bad
3.	#2	more tax revenues	#8	good
4.	#3	decrease of jobs	#8	bad
5.	#3	new jobs	#8	good
6.	#4	destruction of the education system	#8	bad
7.	#4	destruction of the culture	#8	bad
8.	#4	enrichment of the culture	#8	good
9.	#5	more crimes	#8	bad
10.	#5	more riots	#8	bad
11.	#6	dangerous	#8	bad
12.	#6	strange	#8	bad
13.	#6	misbehaving	#8	bad
14.	#6	friendly	#8	good
15.	#8	good	#9	allow
16.	#8	bad	#9	not allow
17.	#8	useful	#9	promote
18.	#8	useless	#9	stop

Ezt az üzenetcsoportot a tesztalanyokkal úgy fogjuk megtanítani, hogy az asszociációkat tökéletesen megtanulják, ezért itt az üzenet típusának és forrásának nincsen jelentősége, mivel a tanulást nem befolyásolja. Tehát ezek az üzenetek a tesztalanyok számára egyfajta egységes értékrendet jelképeznek, mivel azt mondják meg, hogy mi jó és mi rossz.

2. üzenetcsoport: `Message group #1`

#	Ar.	Bal oldali részüzenet	Típus*	Ar.	Jobb oldali részüzenet	Forrás**
1.	#1	refugees	↔	#6	dangerous	-
2.	#1	refugees	↔	#6	strange	-
3.	#1	foreigners	↔	#6	dangerous	+

4.	#1	foreigners	↔	#6	strange	+
5.	#1	refugees	↔	#6	friendly	+
6.	#1	tourists	↔	#6	friendly	+
7.	#1	tourists	↔	#6	strange	–
8.	#1	immigrants	→	#4	destruction of the culture	+
9.	#1	immigrants	→	#4	enrichment of the culture	+
10.	#1	immigrants	→	#2	more tax revenues	0
11.	#1	immigrants	→	#5	more crimes	0
12.	#1	immigrants	→	#4	destruction of the educational system	–
13.	#1	immigrants	→	#3	new jobs	–
14.	#1	immigrants	→	#4	enrichment of the culture	–
15.	#1	refugees	→	#3	decrease of jobs	+
16.	#1	refugees	→	#3	decrease of jobs	+
17.	#1	economic refugees	→	#3	new jobs	0
18.	#1	economic refugees	→	#9	not allow	–
19.	#1	political refugees	→	#2	more tax revenues	0
20.	#1	political refugees	→	#3	new jobs	0
21.	#1	political refugees	→	#9	allow	+
22.	#1	foreigners	→	#3	decrease of jobs	+
23.	#1	foreigners	→	#4	destruction of the culture	+
24.	#1	tourists	→	#3	decrease of jobs	0
25.	#1	tourists	→	#2	more tax revenues	0
26.	#1	tourists	→	#2	more tax revenues	0

* ↔ egyszerű üzenet
→ ok-okozati összefüggést kifejező üzenet

** \+ megbízható forrásból származó üzenet
0 semleges forrásból származó üzenet
– nem megbízható forrásból származó üzenet

A második üzenetcsoportban már fontos szerepet játszik az üzenet típusa és a forrás megbízhatósága is, mivel ezek jelentősen befolyásolni fogják a tanulási folyamatot.

3.2 Hardver- és szoftverkörnyezet

Az alkalmazás nem igényel speciális célhardvert. Szerveroldalon a webszerver és adatbázisszerver számára ajánlott egy kifejezetten ipari felhasználásra szánt szervergépet használni. A szoftver kifejlesztése során igyekeztünk ingyenes, nyílt forráskódú programokkal dolgozni. A szerveroldali számítógépnek a következő komponensekkel kell rendelkeznie:

- *Webszerver*: például Apache, IIS, lighttpd, nginx, vagy bármilyen egyéb HTTP/1.1-es protokollt támogató webszerver.
- *Szerveroldali szkriptnyelv*: az adatok feldolgozását PHP szkriptek végzik, így a PHP legalább 5.x verziója elengedhetetlen, a *php_mysql* kiterjesztéssel.
- *Adatbázisszerver*: az adatok tárolása a MySQL adatbázisszerverrel történik, ahol legalább 5.x verzió megléte szükséges, továbbá be kell kapcsolni az InnoDB adatbázismotort.
- *Operációs rendszer*: Windows XP, 2003, Vista, Linux, Unix, vagy bármely más operációs rendszer, amelyen a fenti komponensek működésre bírhatók.

Kliensoldalon egy korszerű, *JavaScript-et* és *sütiket* támogató böngészőn kívül más szoftver nem szükséges. A tesztelés során az alkalmazást az alábbi böngészőkkel próbáltuk ki sikeresen:

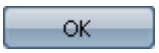
- Firefox 2
  `http://www.mozilla.com/firefox`
- Internet Explorer 6 és 7
  `http://www.microsoft.com/windows/products/winfamily/ie`
- Opera 9
  `http://www.opera.com`

A fejlesztés során a különböző böngészők közötti inkompatibilis DOM (Dokumentum Objektum Modell) és JavaScript implementációk miatt, felhasználtuk az ingyenesen használható *Ext JS* JavaScript könyvtár 2.0.2-es verzióját (`http://extjs.com`), amely továbbá a grafikus felület bizonyos komponenseit (pl. ablakozás) is biztosítja.

3.3 Felhasználói leírás

A felhasználói felület kialakításánál próbáltunk az ablakos rendszereknél már megszokott konvenciókhoz alkalmazkodni. Az alkalmazás beépített helyérzékeny súgót tartalmaz, amelyet a programban mindenhol a 

gomb megnyomásával érhetünk el. A különböző műveletek elvégzésére szolgáló gombok esetén, ha a gomb felirata után '…' látható, akkor a gombra kattintva újabb párbeszédablak jelenik meg, amelyben további beállításokat adhatunk meg, vagy megerősíthetjük a döntésünket (például törlésnél). Ellenkező esetben (vagyis ha a gomb felirata után nincsen három pont), akkor a művelet azonnal, megerősítés nélkül végrehajtódik. A legtöbb párbeszédpanelen a következő gombok találhatók meg:



Az OK gomb elfogadja a párbeszédpanelen végrehajtott változtatásokat és bezárja az ablakot.

A Cancel (Mégse) gomb a párbeszédpanelt a változtatások mentése nélkül zárja be.

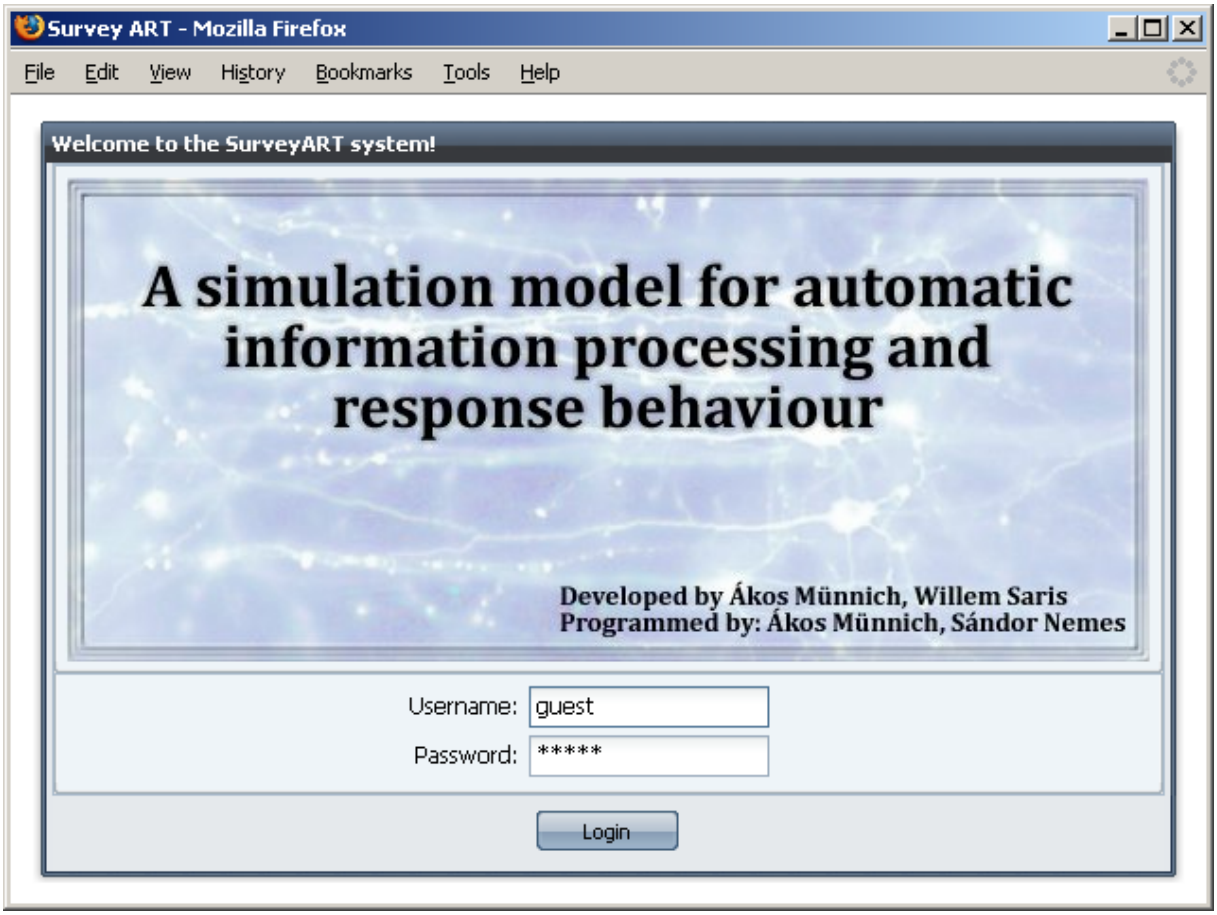
A Help (Súgó) gomb megjeleníti a párbeszédpanel elemeire vonatkozó helyérzékeny súgót.

Egyes ablakokban, amelyek nem tartalmaznak menthető adatokat (vagyis kizárólag információt jelenítenek meg), a Cancel (Mégse) és az OK gomb helyett a Close (Bezárás) gomb található.


Egyes ablakok jobb felső sarkában megtalálható a teljesméret gomb, amellyel az ablakot kinagyíthatjuk úgy, hogy a böngésző teljes ablakát kitöltse.


Az ablakok jobb felső sarkában található X gomb az ablak típusától függően funkcionálisan megegyezik a Cancel (Mégse) vagy a Close (Bezárás) gombbal.

A webalkalmazás címét a böngészőbe beírva, a következő nyitólap fogadja a felhasználót:



Itt lehet a megfelelő felhasználónévvel és jelszóval bejelentkezni a rendszerbe. Az alkalmazás két előre beállított felhasználót tartalmaz:

Felhasználónév	Jelszó	Leírás
`guest`	`guest`	Ezzel a felhasználónévvel belépve a példaadatbázis adatai töltődnek be, így könnyen kipróbálható a rendszer.
`empty`	`empty`	Ez a felhasználónév szintén a rendszer kipróbálására szolgál, azonban ebben az esetben a felhasználó egy üres adatbázist kap.

Ha a bejelentkezés sikeres, akkor a 

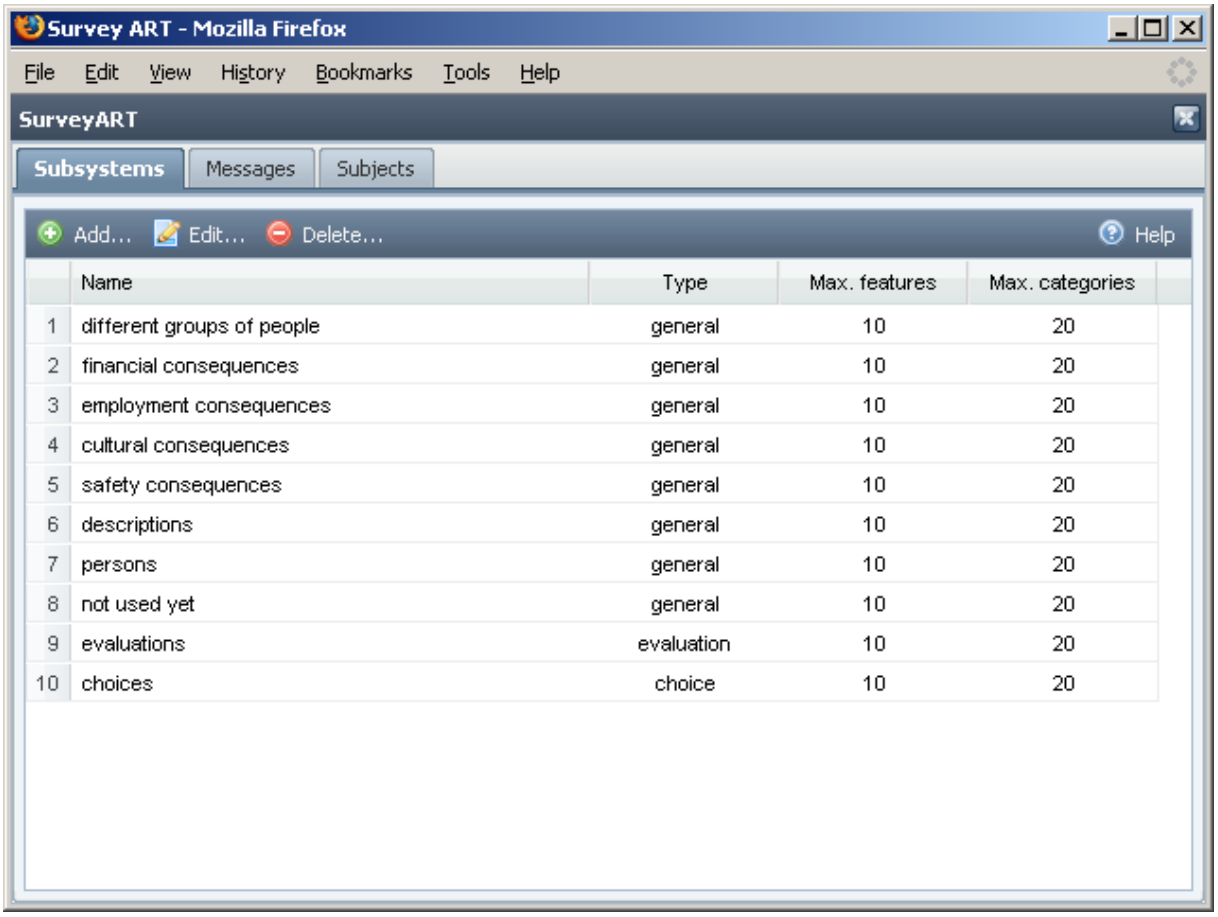
gombra kattintva megjelenik a rendszer főablaka, amelyen kezdetben a 

fül aktív.

3.3.1 Alrendszerek

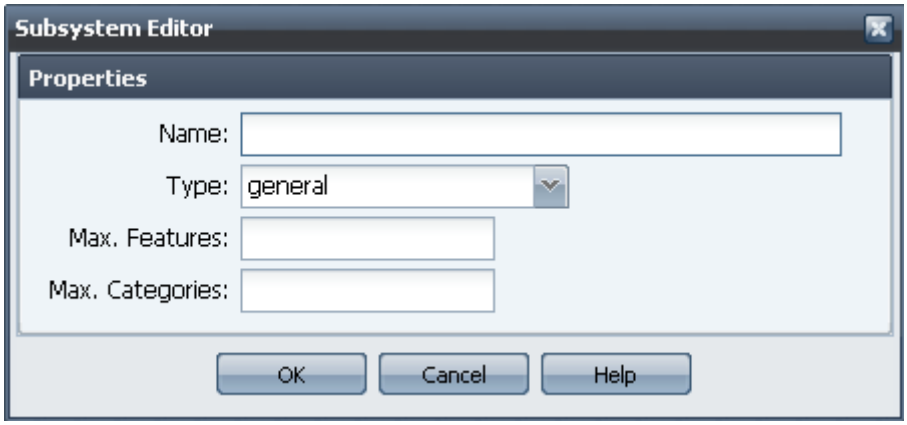
Az alkalmazásban a 

fülre kattintva érhetjük el azt a panelt, ahol az alrendszerek felvitelét, szerkesztését és törlését elvégezhetjük. Egy táblázatos formában látjuk az alrendszer nevét, típusát, az alrendszer kódvektorainak a dimenzióját (a jellemvonások számát), valamint az alrendszerhez kapcsolódó neurális hálóban tárolható kategóriák maximális számát.



Közvetlenül a táblázat felett található a három szerkesztőgomb:

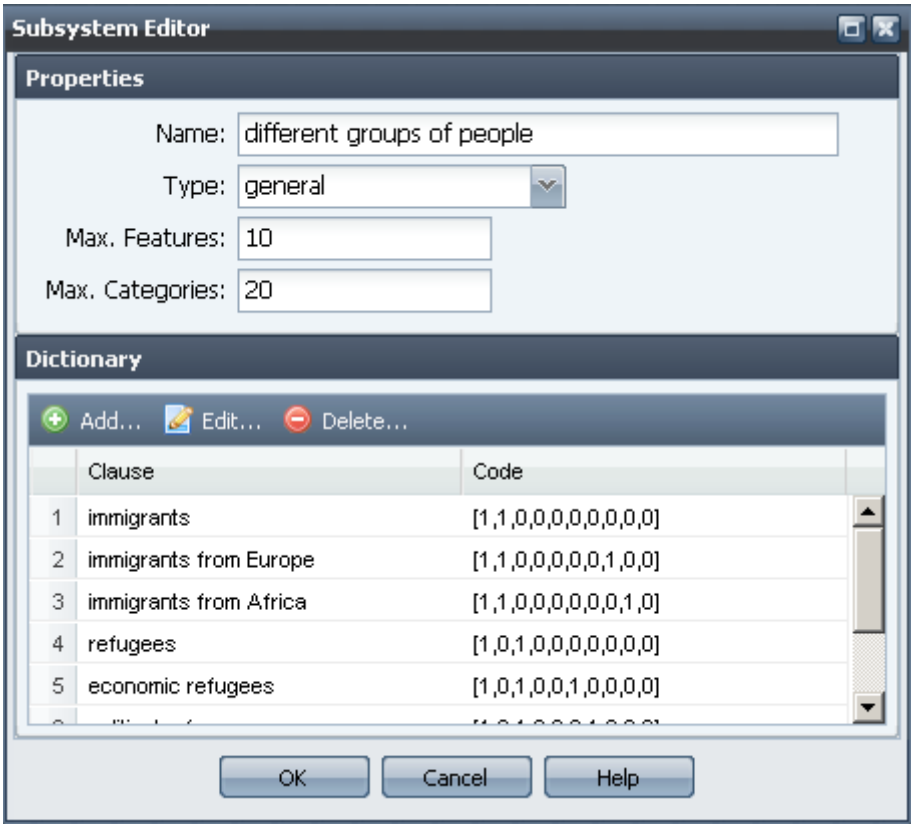
Ezzel a gombbal vehetünk fel újabb alrendszert. A gomb megnyomására az alábbi párbeszédpanel jelenik meg:



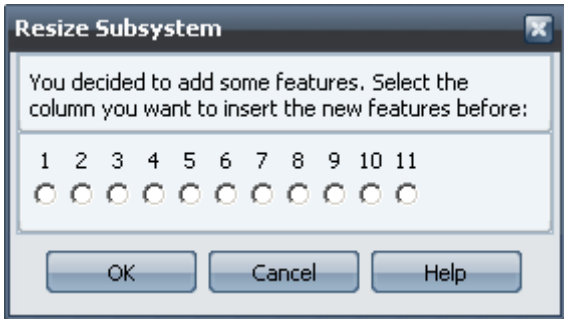
Itt megadhatjuk az alrendszer nevét, kiválaszthatjuk a típusát, valamint beállíthatjuk a jellemvonások maximális számát (vagyis a kódvektor dimenzióját) és a neurális hálózatban létrehozható kategóriák maximális számát. Az alrendszerek az alábbi típusokból kerülhetnek ki:

- `General` (általános alrendszer): Ezeket az alrendszereket fogjuk a leggyakrabban használni. A tesztalanyok kikérdezése során azt vizsgáljuk majd, hogy az általános alrendszerek között milyen asszociációk alakultak ki.
- `Evaluation` (értékelő alrendszer): A kikérdezés során az általános alrendszerek között kialakult asszociációkat az értékelő alrendszerbe vezetjük, ahol értékelés után a kimenő kódvektorokat kombináljuk.
- `Choice` (döntési alrendszer): Az értékelő alrendszer kimenetét a döntési alrendszerbe vezetve ez az alrendszer fogja megadni a feltett kérdésekre a végső választ.

Ezt a gombot választva szerkeszthetjük az alrendszer alapvető tulajdonságait, ezen felül itt rendelhetjük egymáshoz az alrendszerben a kódvektorokat a kezelhetőséget megkönnyítő szöveges reprezentációval.



Amennyiben úgy döntünk, hogy a kódvektorok dimenzióját növeljük, meg kell adnunk, hogy az új értékek hova kerüljenek. Vagyis ha például a dimenziót 10-ről 20-ra növeljük, az alábbi ablak jelenik meg:



Itt kiválaszthatjuk, hogy a vektor hanyadik eleme elé szeretnénk beszúrni az új értékeket. Ha a dimenziók számát csökkenteni próbáljuk, akkor viszont ki kell törölnünk az összes kódvektorból bizonyos elemeket. Ha például a fenti szerkesztőablakban 10-ről 4-re szeretnénk csökkenteni a dimenziószámot, akkor pontosan 6 db elemet kell megjelölnünk:

A törlés gombbal törölhetjük az alrendszereket. Ezt megtehetjük egyenként, vagy a <Shift> illetve a <Ctrl> billentyű segítségével több alrendszert is kijelölhetünk egyszerre. Az alrendszer törlésével az adatbázisból törlődik az összes hozzá tartozó kódvektor-szöveg pár, valamint az összes olyan üzenet, amely az adott alrendszerre hivatkozik.

3.3.2 Üzenetek

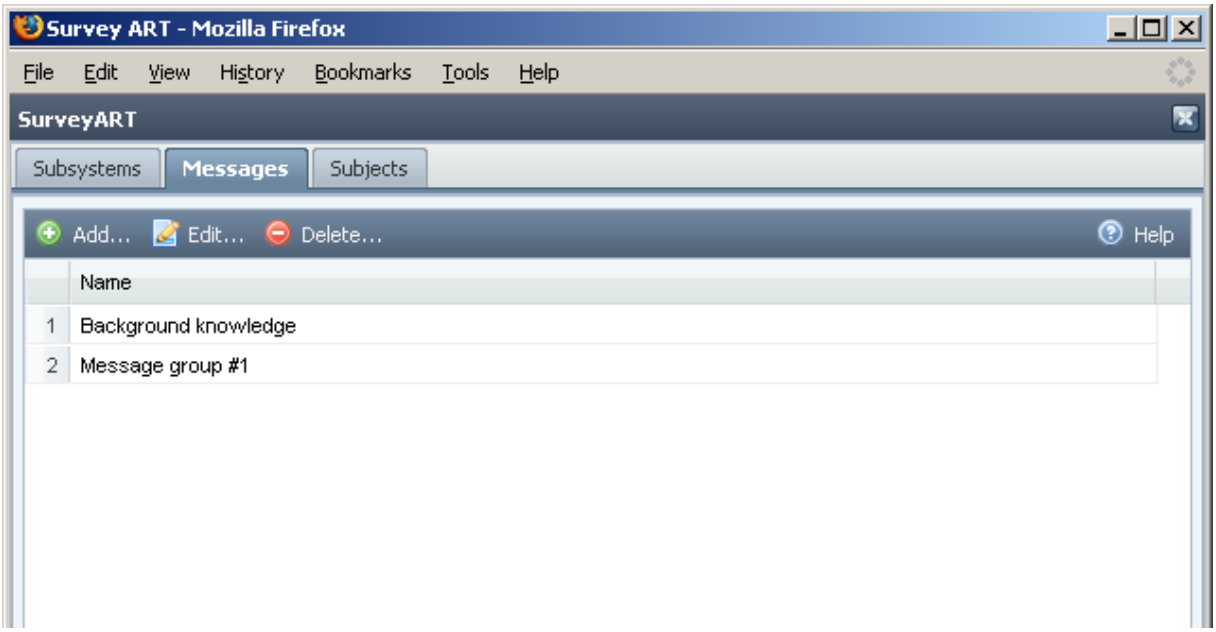
Az üzeneteket tartalmazó panelt a 

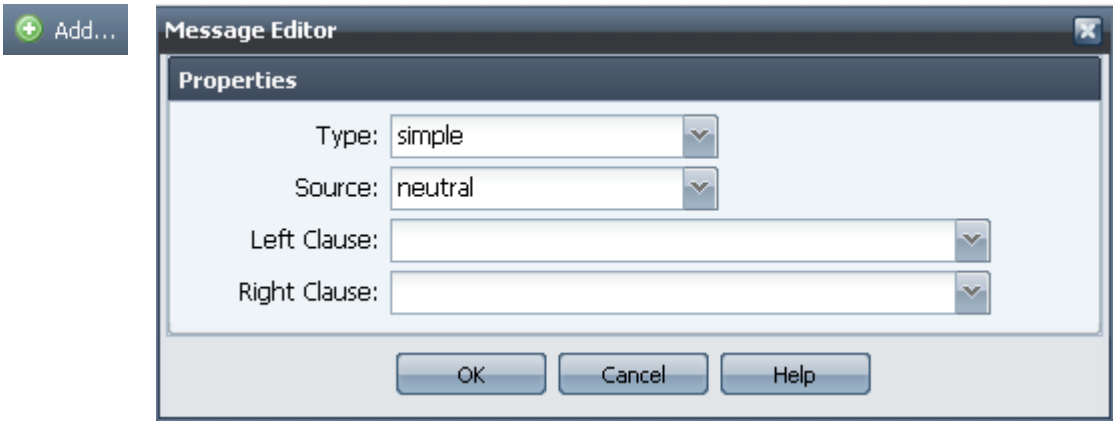
fülre kattintva érhetjük el. Itt az üzeneteket tartalmazó üzenetcsoportokat látjuk. Az 

, 

és 

gombokkal módosíthatjuk az üzenetcsoportokat. A szerkesztőpanelen az üzenetcsoport nevének módosításán kívül elvégezhetjük maguknak az üzeneteknek a felvitelét is.





Új üzenet felvitelekor meg kell határozni, hogy az üzenet milyen típusú lesz. Két típus közül választhatunk:

- `Simple` (egyszerű üzenet): Egyszerű üzeneteknél az üzenet mindkét fele egyforma súllyal számít, és az üzenet forrásának megbízhatósága is egyformán befolyásolja az üzenet mindkét felét a tanulás során.
- `Causality` (okozati összefüggés): Ha az üzenet ok-okozati összefüggést ír le, akkor az üzenet első fele (az „ok") jobban fog számítani a tanulás során.

Az üzenet forrását is itt választhatjuk ki a legördülő menüből:

- `Trusted` (megbízható): A megbízható üzeneteket a tesztalanyok jobban megtanulják, vagyis tanulás során a tanulási paraméter értéke magasabb lesz az alapértéknél.
- `Neutral` (semleges): Semleges megbízhatóságú üzenetnél a tanulási paraméter értéke nem változik meg a tanulás során.
- `Distrusted` (nem megbízható): Ha a tesztalanyhoz nem megbízható üzenet érkezik, akkor kevésbé tanulja meg az üzenet tartalmát, vagyis a tanulási paraméter csökkenni fog az alapértékhez képest.

Az üzenetet alkotó bal és jobb részüzeneteket a legördülő menüből választhatjuk ki, ahol szerepel az összes alrendszer összes kódvektorához

tartozó szöveges reprezentáció.

A szerkesztés gombra kattintva az előző párbeszédpanelhez hasonlót kapunk, ahol átírhatjuk az üzenet tulajdonságait.

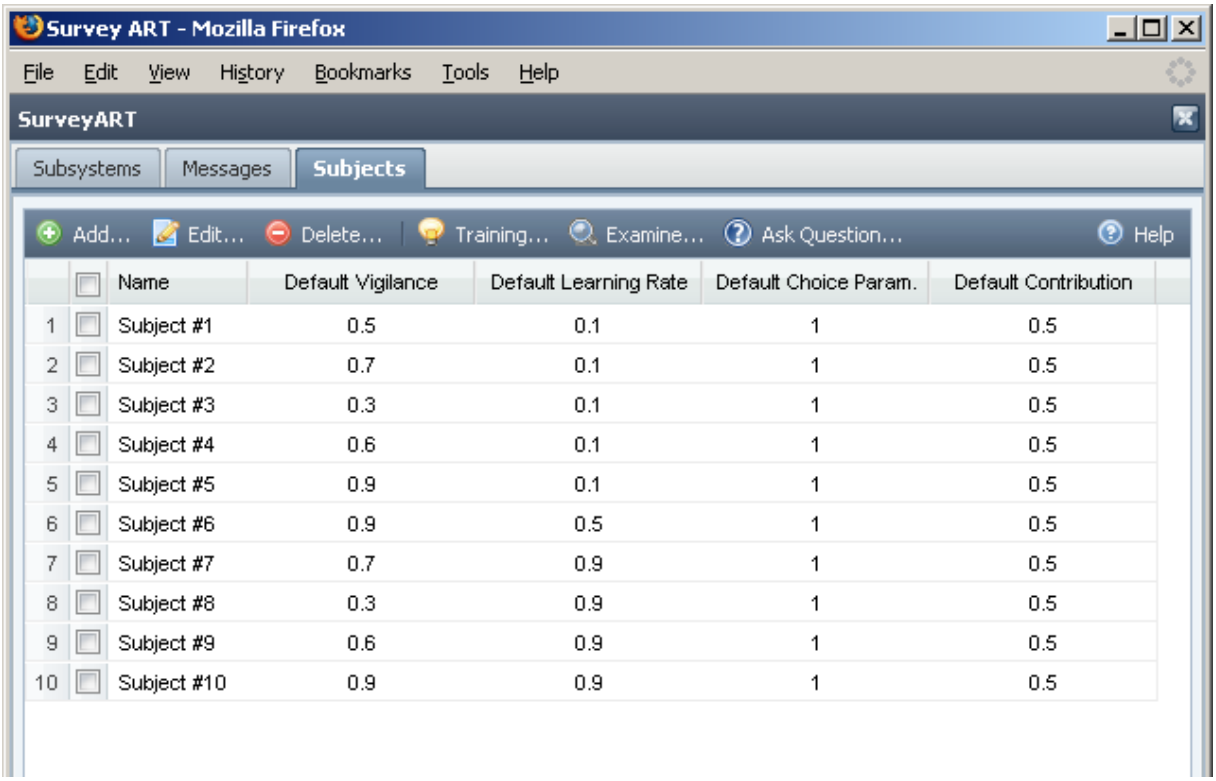
Az üzenetek sorrendjét ezekkel a gombokkal állíthatjuk. Az aktuálisan kijelölt üzenetet mozgathatjuk fel és le. Ha több üzenet van kijelölve akkor csak a legelsőre vonatkozik a mozgatás.

A törlés gombbal eltávolíthatjuk a kijelölt az üzenetet vagy üzeneteket az adatbázisból.

3.3.3 Tesztalanyok

A rendszer talán legfontosabb pontja magukat a kísérleti alanyokat tartalmazó panel, amely a 

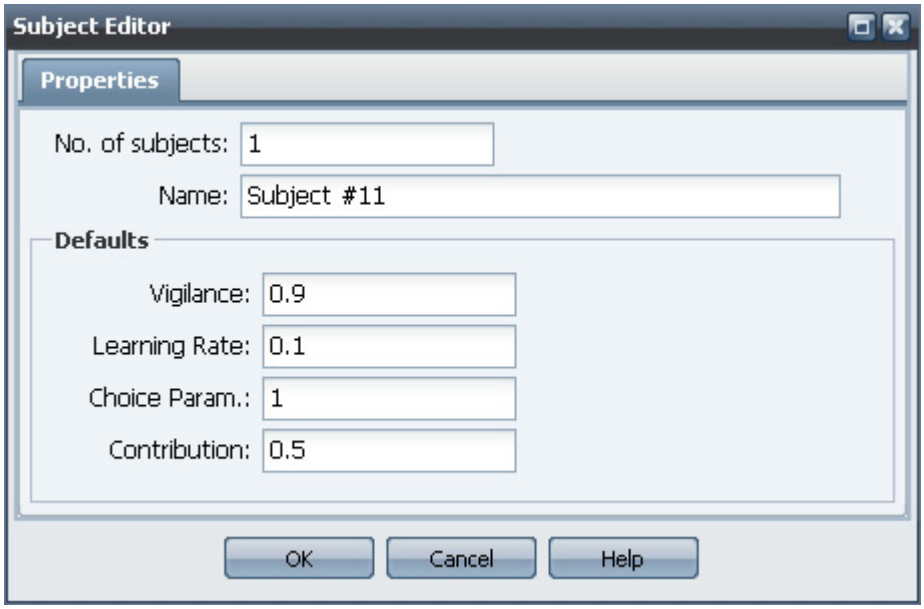
fülre kattintva hívható elő. Ekkor megjelennek az adatbázisban lévő tesztalanyok, és a hozzájuk tartozó alapértelmezett paraméterek (az alapértelmezett vigilancia, tanulási paraméter, választási paraméter és a részesedési paraméter):



	Name	Default Vigilance	Default Learning Rate	Default Choice Param.	Default Contribution
1	Subject #1	0.5	0.1	1	0.5
2	Subject #2	0.7	0.1	1	0.5
3	Subject #3	0.3	0.1	1	0.5
4	Subject #4	0.6	0.1	1	0.5
5	Subject #5	0.9	0.1	1	0.5
6	Subject #6	0.9	0.5	1	0.5
7	Subject #7	0.7	0.9	1	0.5
8	Subject #8	0.3	0.9	1	0.5
9	Subject #9	0.6	0.9	1	0.5
10	Subject #10	0.9	0.9	1	0.5

Itt a tesztalanyok listájának szerkesztésén kívül az alkalmazás egyéb lényeges funkcióit is elérhetjük. Itt módosíthatjuk az alanyok alrendszerekhez és alrendszerpárokhoz tartozó paramétereit, megtekinthetjük a kialakult neurális hálózatok kódvektorait, és az alrendszerpárok között kialakult kapcsolatokat, valamit elindíthatjuk a tanítást és kérdéseket tehetünk fel a tesztalanyoknak.

Új tesztalany hozzáadásánál meg kell adni a tesztalany nevét, amit a rendszer az adatbázisban lévő tesztalanyok száma alapján automatikusan felajánl. Például ha 10 tesztalany már van az adatbázisban, akkor az alkalmazás a „`Subject #11`” elnevezést fogja ajánlani. Itt adhatjuk meg az alapértelmezett paramétereket is, amelyek a későbbiekben alrendszerenként felülbírálhatók. Ha több tesztalanyt is szeretnénk egyszerre felvinni, akkor a „`No. of subjects`” mezőben megadhatjuk a kívánt értéket. Ebben az esetben az alkalmazás automatikusan úgy generálja a neveket, hogy hozzájuk fűz egy sorszámot. Például ha 10 új tesztalanyt szeretnénk létrehozni, és névhez beírjuk, hogy „`Tesztalany`”, akkor a létrejövő tesztalanyok a „`Tesztalany #1`”, „`Tesztalany #2`”, …, „`Tesztalany #10`” nevet fogják kapni.

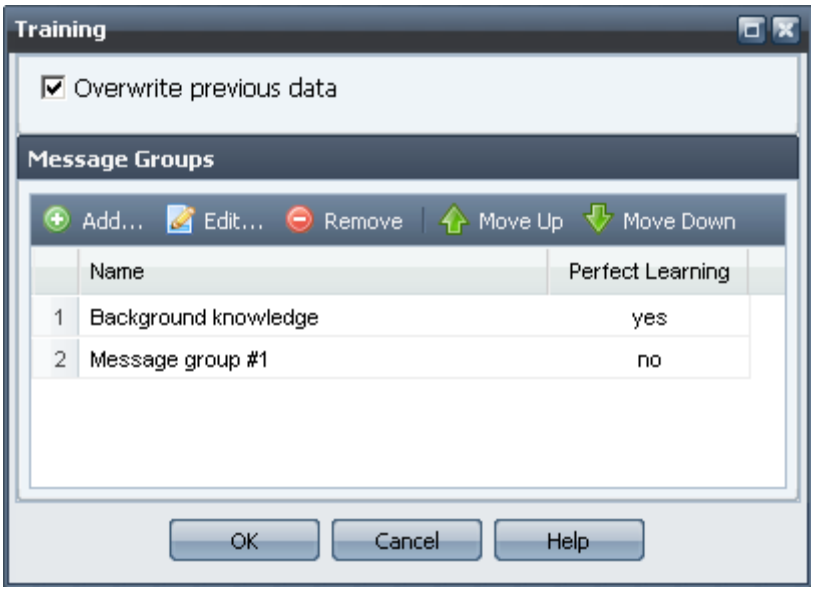


A tesztalany szerkesztésénél az előzőhöz hasonló ablak fogadja a felhasználót, azonban a tesztalanyok számára vonatkozó mező inaktív,

vagyis csak a tesztalany nevére és paramétereire vonatkozó beállításokat tudjuk megváltoztatni. Viszont megjelenik a párbeszédpanelen 2 új fül az ART params és a ARAM params, ahol alrendszerenként és alrendszerpáronként felülbírálhatjuk a tesztalanyok neurális hálózatára vonatkozó paramétereket. Mivel az alrendszerek ART neurális hálózatoknak felelnek meg, ezért ebben az esetben a paraméterek a következők: vigilancia, tanulási paraméter és választási paraméter. Az alrendszerpárok pedig valójában ARAM neurális hálózatok. Mivel az ARAM hálózat egy asszociatív neurális hálózat, ami két ART modulból áll, ezért itt mindkét ART almodul rendelkezik a fenti 3 paraméterrel, továbbá van egy részesedési paraméter, amellyel a két modul közötti súly állítható.

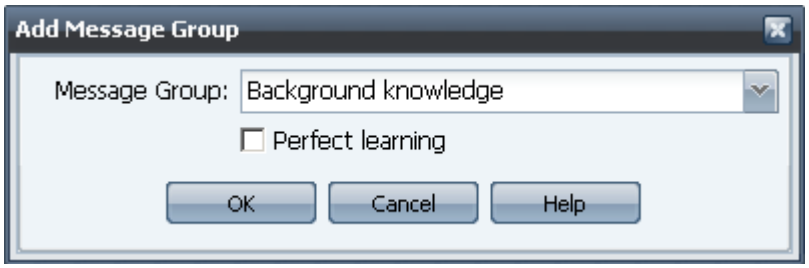
Delete... A törlés gombbal törölhető a kijelölt tesztalany. A <Shift> vagy a <Ctrl> billentyű segítségével több tesztalany is kijelölhető egyszerre.

A tesztalanyok tanításához ki kell jelölni egy vagy több tesztalanyt, majd az eszköztár Training... gombja segítségével kiválaszthatjuk, hogy mely üzenetcsoportokat kívánjuk a kijelölt tesztalanyoknak megtanítani:



Ha az „Overwrite previous data” opciót beállítjuk, akkor a tanítás megkezdése előtt a tesztalanyok által korábban megtanult információk törlődnek.

A hozzáadás gombra kattintva a következő párbeszédpanel jelenik meg:

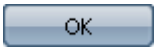


A legördülő listából választhatjuk ki a megtanítandó üzenetcsoportot. A „Perfect learning” opció segítségével azt szabályozhatjuk, hogy tanulás közben figyelembe vegye-e a rendszer az üzenetek típusát és megbízhatóságát. Amennyiben ez az opció ki van kapcsolva, akkor a típust és a forrást figyelembe vesszük, és ez befolyásolja a tanulás mértékét. Bekapcsolt állapotban a tesztalanyok minden üzenetet tökéletesen megtanulnak, ebben az esetben a típusnak és a forrásnak nincsen jelentősége.

A szerkesztés gomb segítségével az aktuálisan kijelölt elem tulajdonságait szerkeszthetjük, pontosan úgy, ahogyan azt fentebb a hozzáadásnál ismertettük.

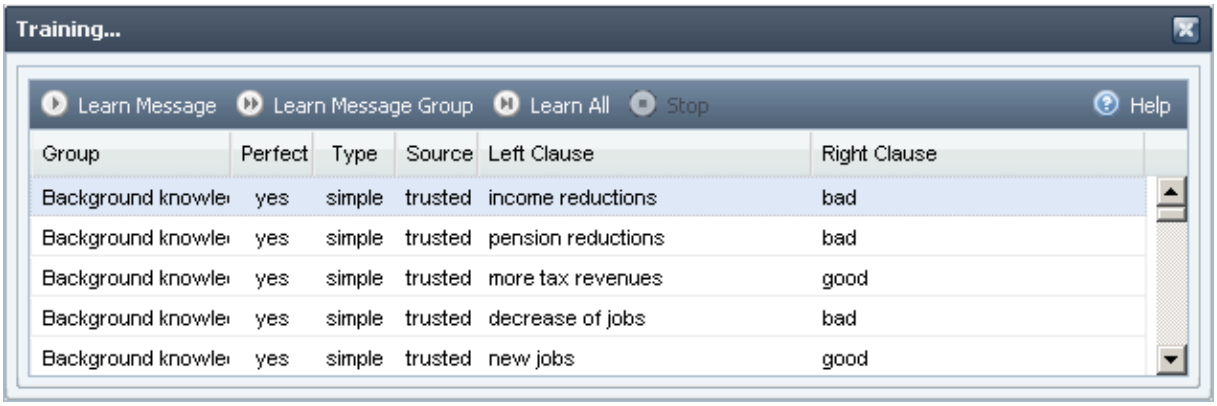
Ezzel a két gombbal szabályozhatjuk a kiválasztott üzenetcsoportok pozícióját a listában, vagyis ezzel befolyásolhatjuk a megtanulandó üzenetek sorrendjét.

Az eltávolítás gomb segítségével a kijelölt üzenetcsoportot vagy üzenetcsoportokat távolíthatjuk el a listából.

Amennyiben a listában szereplő üzenetcsoportokkal és a sorrendjükkel elégedettek vagyunk az 

gombra kattintva az alkalmazás az üzenetcsoportokból összeállítja a

megtanítandó üzenetek listáját, amelyet a képernyő alján megjelenő ablakban tekinthetünk meg:



Amíg az alkalmazás tanulási módban van, vagyis a fent látható tanítás ablak nyitva van, addig a 

gomb inaktív, nem választható ki. A tanítás ablakban az alábbi gombok segítségével indíthatjuk el, illetve állíthatjuk le a tanítást:

A „Learn Message” gombot választva, az előző pontban kiválasztott tesztalanyok a „Training…” ablakban kijelölt üzenetek közül az aktuálisan kijelölt üzenetet tanulják meg.

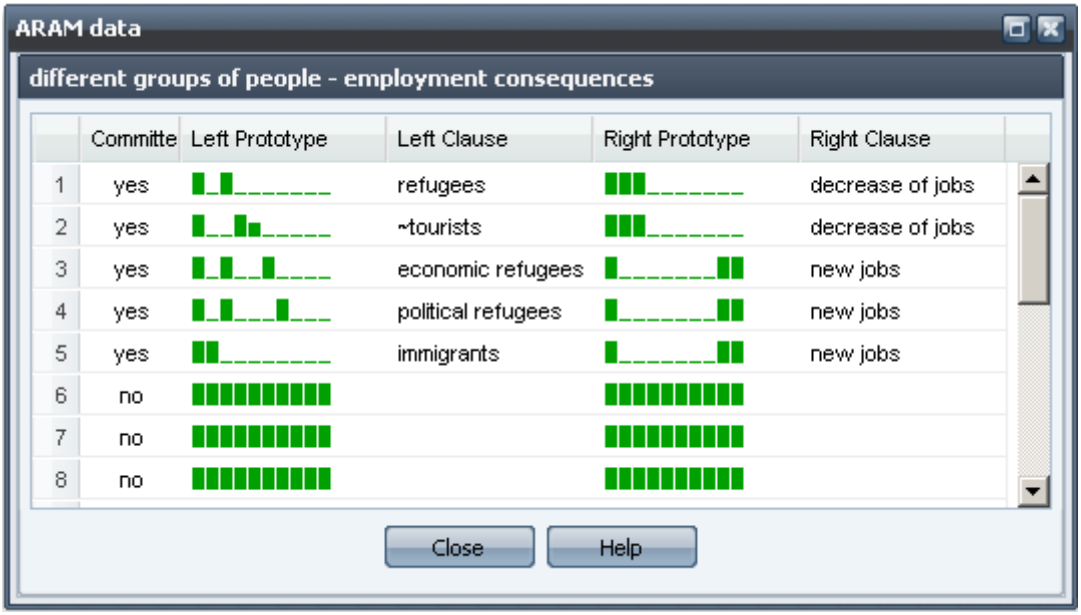
A „Learn Message Group” használatával a tanulás egészen addig folytatódik, amíg az aktuális üzenetcsoport tanulása véget nem ér.

Ezt a gombot választva a tanulási ablakban lévő összes üzenetet megtanulják a tesztalanyok.

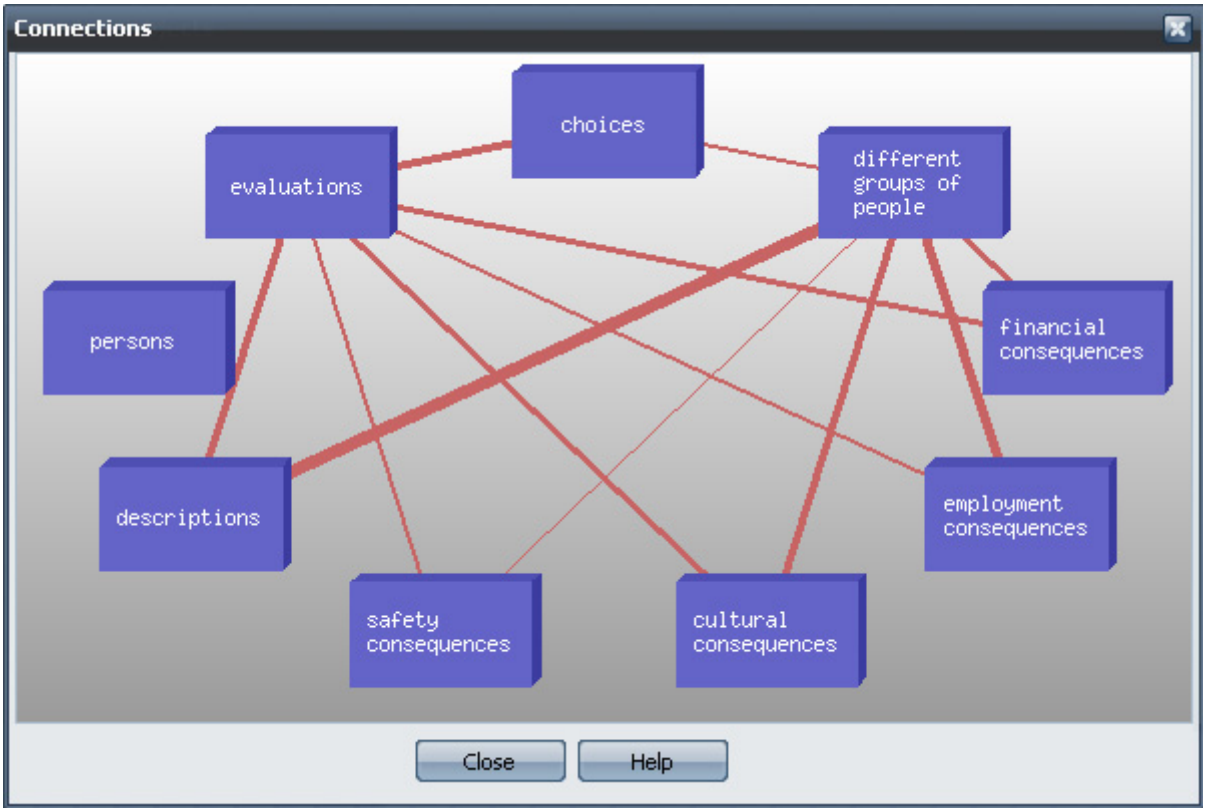
A „Stop” gomb a tanítás megállítására szolgál. A gomb csak akkor aktív, ha a tanulás éppen folyamatban van.

Az eszköztár 

gombja segítségével a tanítás előtt, közben és után megtekinthetjük a tesztalanyok neurális hálózatában kialakult állapotokat, értékeket. A gombra kattintva, majd a megnyíló ablakban az alrendszert (ART hálózat esetén) vagy az alrendszerpárt (ARAM hálózat esetén) kiválasztva megtekinthetjük a prototípusvektorokat:

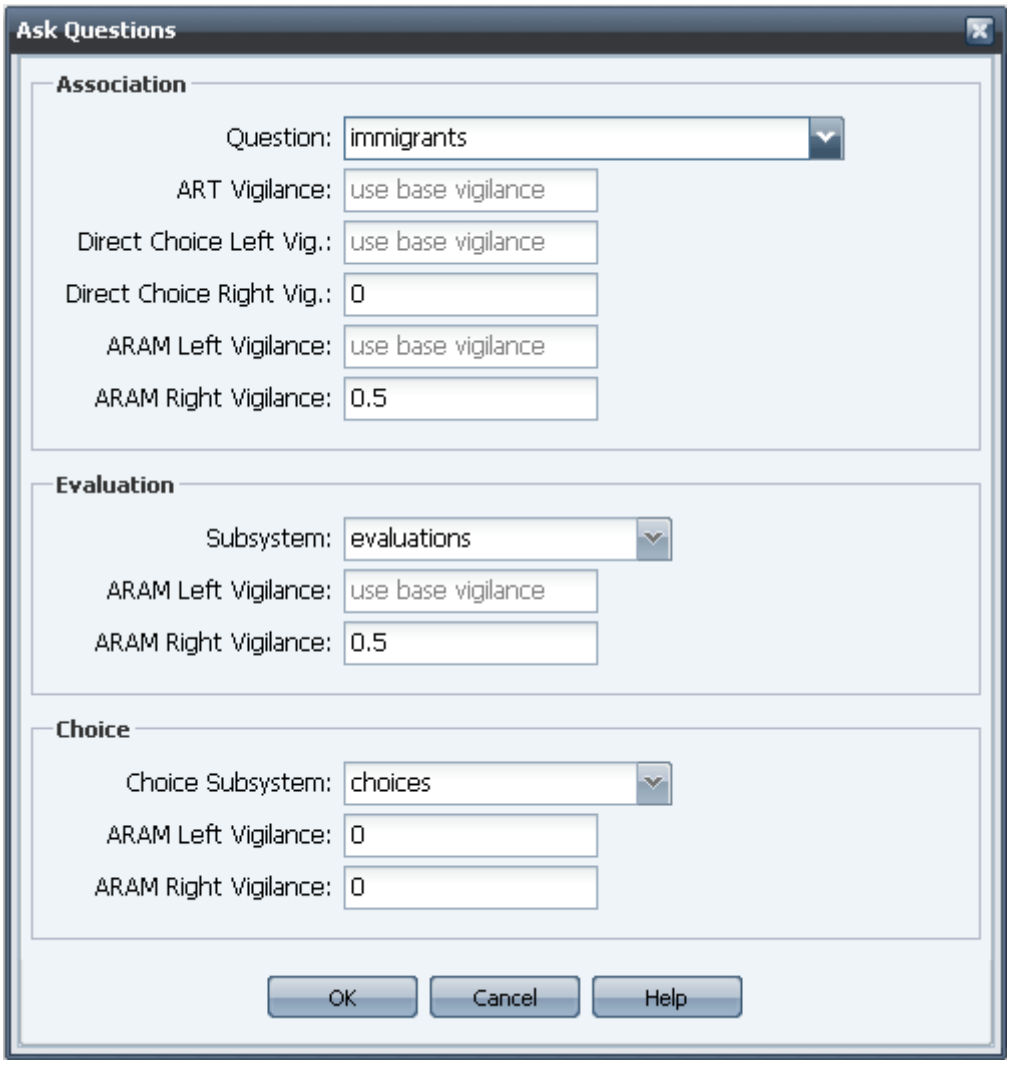


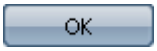
A vektorok értékét a zöld oszlopok jelzik. Minél magasabb egy oszlop, a vektor adott elemének értéke annál nagyobb. Az egeret a zöld oszlopokon pihentetve, a vektor értéke numerikusan is megjelenik. Az ablakban a prototípushoz tartozó szöveges reprezentáció is látható. Ha az adott prototípusnak van megfelelő szöveges reprezentációja, akkor a mellette lévő oszlopban az látható. Ha nincsen pontosan megegyező szöveges reprezentáció, de a tesztalany aktuális alrendszerhez tartozó vigilanciáját felhasználva a szótármodul neurális hálózata talált megfelelő reprezentációt, akkor a szöveges megfelelő előtt egy ~ jel (hullámvonal) látható. Ha így sem sikerült szöveggé alakítani a vektort, akkor a neurális hálózat egy újabb próbát tesz, ezúttal a vigilanciát 0-ra állítva. Ekkor a talált szöveg mellett ~~ jel (két hullámvonal) szerepel.



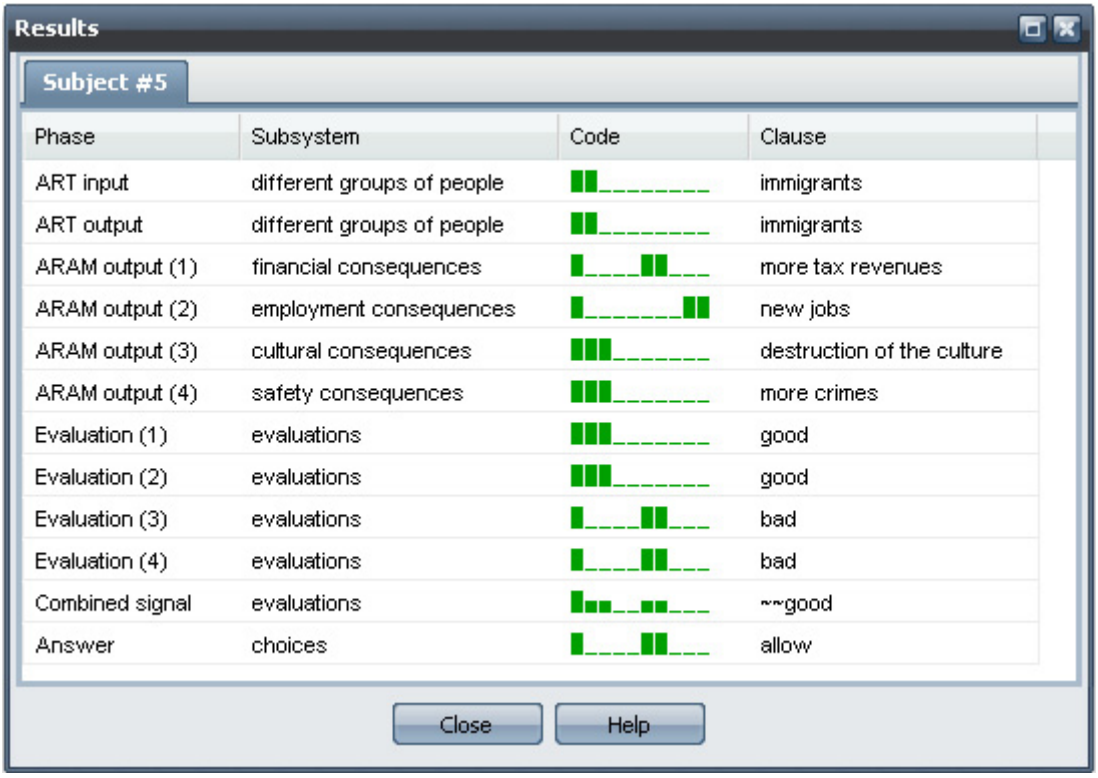
Ha az Examine... gombra megjelenő ablakban az eszköztár Show Connections... gombját választjuk ki, akkor megtekinthetjük az alrendszerek között kialakult kapcsolatokat. A megnyíló ábrán minden alrendszert egy doboz jelképez, amely tartalmazza az alrendszer nevét. A dobozok közötti összeköttetések mutatják, hogy melyik alrendszer melyik másik alrendszerrel alakított ki kapcsolatokat. Minél vastagabb az összeköttetés annál több kapcsolat alakult ki az alrendszerek között.

Végül a tesztalanyok kikérdezését az Ask Question... gombbal végezhetjük el. Kikérdezésnél több tesztalany is kiválasztható egyszerre. A megjelenő ablakban kell megadnunk a kérdést, valamint a neurális hálózat kikérdezéséhez használandó különböző paramétereket. Amennyiben nem adunk meg paramétert, vagyis a beviteli mezőt üresen hagyjuk, abban az esetben automatikusan a tesztalany megfelelő alrendszerhez vagy alrendszerpárhoz tartozó alapértékeit használja a rendszer. Továbbá azt is itt kell eldöntenünk, hogy melyik alrendszert használja az alkalmazás értékelésre és melyiket a végső döntés meghozatalához. Értékelésre természetesen csak olyan alrendszert választhatunk ki, amelynek a típusát korábban `Evaluation`-re állítottuk, döntési alrendszernek pedig csak a `Choice` típusú alrendszerek választhatók.



A kikérdezés során a feltett kérdést a rendszer egy ART neurális hálózaton vezeti át, ahol a tesztalany besorolja egy kategóriába. Ha ennek a kialakult kategóriának van közvetlen kapcsolata a döntési alrendszerrel, akkor a döntési alrendszer egyből megadja a kérdésre feltett választ. Abban az esetben, ha nincs közvetlen kapcsolat, akkor a rendszer megpróbál asszociációt keresni ezzel a kategóriával a többi általános alrendszerben. A többi alrendszer válaszát ezután az értékelő alrendszernek küldjük, amely eldönti, hogy ezek az asszociációk kedvezőek-e vagy sem. A különböző alrendszerekből jövő válasz értékelését ezután összegzi, és a döntési alrendszernek továbbítja. A döntési alrendszer ezután megadja a kérdésre a végső választ. A folyamat során lezajló részeredményeket az 

gomb megnyomása után megjelenő ablakban tekinthetjük meg. Itt látható, hogy melyik fázis részeredményéről és melyik alrendszerről van szó. Látható továbbá a kódvektor és ennek a szöveges megfelelője is.



Tekintsük a következő eredményeket, és nézzük őket végig lépésenként! Az első sor mutatja, hogy feltetettük a tesztalanynak azt a kérdést, hogy mi a véleménye a *bevándorlókról* (*immigrants*). Ez átment az ART neurális hálón, ami nem változtatott a bemenetként kapott adaton. Ez azt jelenti, hogy az éppen vizsgált tesztalany részletes kategóriákat alakított ki az agyában, pontosan tudja, hogy mi az, hogy *bevándorló*. [Amennyiben egy olyan tesztalanyt vizsgáltunk volna, akinek a vigilancia paramétere alacsony, akkor az nem alakított volna ki ennyire finom kategóriákat és lehet hogy a *bevándorlókat* egy másik kategóriával sorolta volna egybe, például azt a választ adta volna, hogy a *bevándorló* nem más mint egy *külföldi* (*foreigners*).] Mivel a kapott üzenetek egyike sem fogalmazott meg ilyen esetre közvetlen döntést, ezért a döntési alrendszerrel ez a kategória nincs közvetlen kapcsolatban. A rendszer megvizsgálta az alrendszereket és a következő kialakított asszociációkat találta:

- pénzügyi következmények: több adóbevétel
- foglalkoztatási következmények: több munkahely
- kulturális következmények: a kultúra pusztulása
- biztonsági következmények: magasabb bűnözés

Ezeket a különböző asszociációkat az értékelő alrendszer értékelte és a következő választ adta:

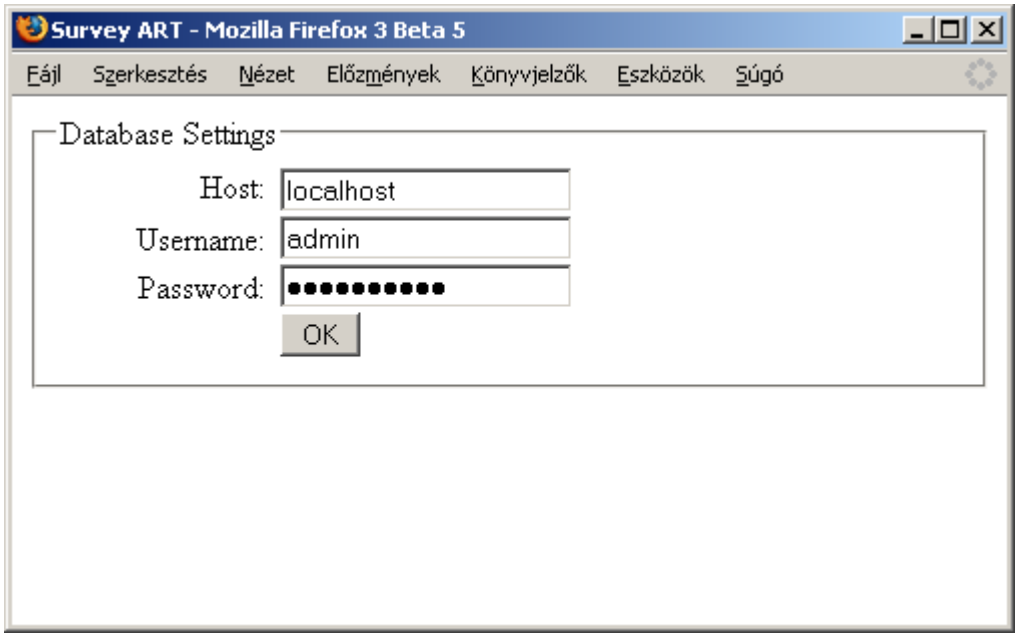
- több adóbevétel: jó
- több munkahely: jó
- a kultúra pusztulása: rossz
- magasabb bűnözés: rossz

Ezek az értékeket kombinálva kapjuk meg a végső értékelést, amely most az esetünkben pont félúton van a jó és rossz között. A rendszer ezt összességében jónak értékelte, és a döntési alrendszer a *bevándorlókra* vonatkozóan pozitív eredményt adott. A kérdés feltevésénél a döntési alrendszer vigilanciáját magasabbra állítva szabályozhatjuk, hogy a rendszer csak akkor adjon választ, ha elér egy bizonyos konfidenciaküszöböt. A példában szereplő esetet tekintve ekkor a rendszer nem tudna megfelelő választ adni, mert a döntés nem elég magabiztos.

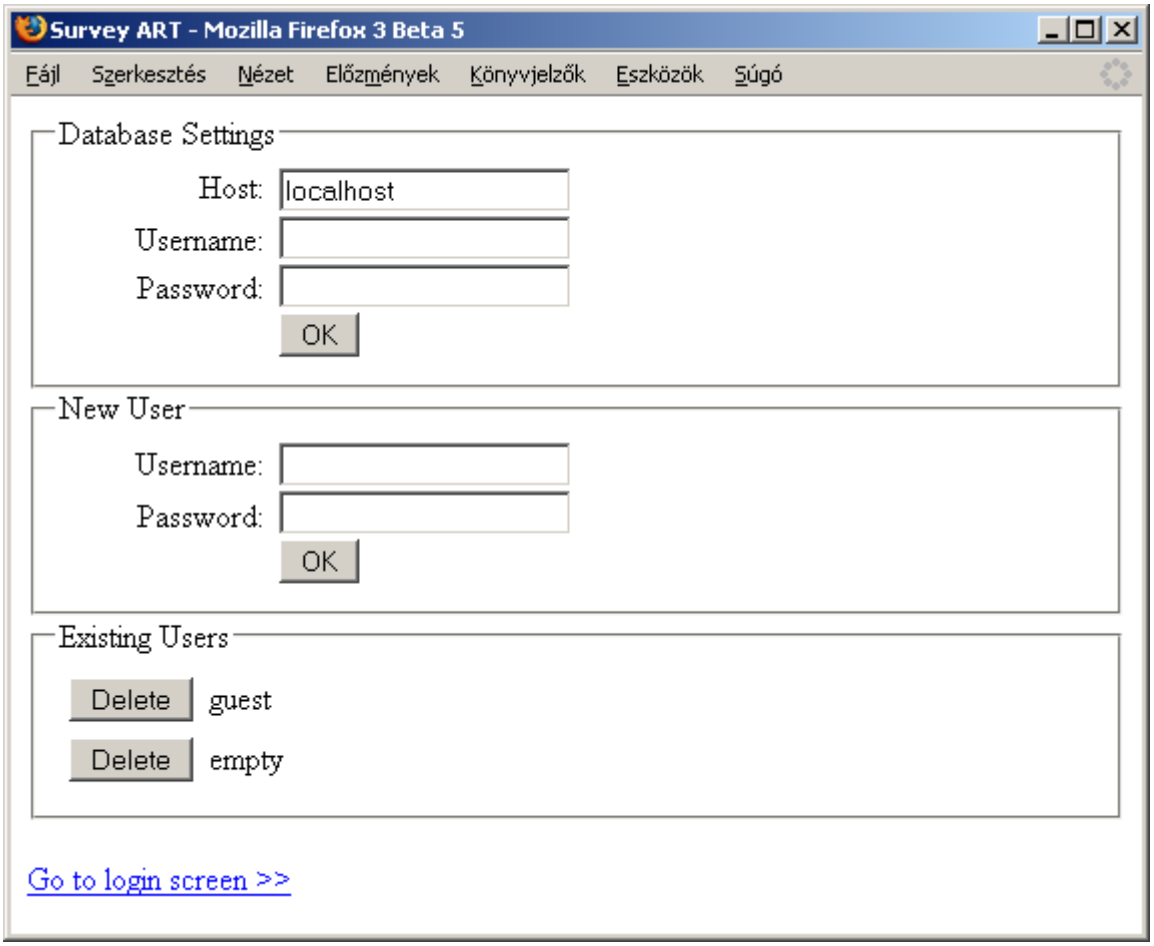
3.4 Adminisztrátori leírás

Feltételezzük, hogy a szerveren a **Szoftverkörnyezet** fejezetben ismertetett komponensek telepítve és konfigurálva vannak (Apache, PHP a php_mysql kiterjesztéssel, MySQL az InnoDB adatbázismotorral). Ekkor az alkalmazás telepítése a szerverre a következő lépésekből áll:

1. A `surveyart-1.0.zip` fájlt tömörítsük ki egy olyan mappába, amely a webszerver segítségével elérhető. Most ebben a példában feltételezzük, hogy az archívum tartalmát a szerverről a `http://localhost/surveyart` címen elérhető mappában helyeztük el.
2. Írjuk be a szerveren egy webböngészőbe azt az URL-t, ahová a szerveren elhelyeztük az alkalmazást, esetünkben most ez a `http://localhost/surveyart` cím. Ekkor bejön egy telepítőképernyő, ahol megadhatjuk az adatbázisszerver címét, és az adatbázis eléréséhez használható nevet és jelszót. Ennek a felhasználónak a megfelelő jogosultságokkal kell rendelkeznie az adatbázisszerveren ahhoz, hogy létrehozza a `surveyart` adatbázist, és ebben a megfelelő táblákat, továbbá a későbbiekben *SELECT*, *UPDATE* és *DELETE* műveleteket hajtson végre a táblán.



3. Ha a megfelelő adatokat adtuk meg, akkor a telepítő kapcsolódik az adatbázisszerverhez, és ha nem létezik a szükséges adatbázis, akkor létrehozza azt. Ezután lehetőségünk van felhasználók hozzáadására:



4. A telepítés befejezése után a „Go to login screen >>” linkkel az újonnan feltelepített rendszer bejelentkezőképernyőjére ugorhatunk és megkezdhetjük az alkalmazás használatát.

3.5 Fejlesztői leírás

3.5.1 Fájlok és mappák

A webalkalmazást alkotó fájlok a szerveren a következő struktúrában helyezkednek el:

classes
`Dictionary.class.php`
`FuzzyARAM.class.php`
`FuzzyART.class.php`
`SurveyART.class.php`
	Ebben a mappában helyezkednek el az alkalmazás által használt objektumorientált PHP osztályok. A *Dictionary* osztály végzi a kódvektorok szöveggé alakítását. A *FuzzyART*, *FuzzyARAM* és *SurveyART* osztályok a korábban már tárgyalt megfelelő neurális hálózatok implementációi.

ext-2.0.2	Az Ext JS 2.0.2 JavaScript könyvtár mappája.
images	Az alkalmazás által használt grafikus elemek.

includes
`config.inc.php`
`dbconn.inc.php`
`init.inc.php`
	Különböző PHP segédállományok:
	A *config.inc.php* állomány tartalmazza az adatbázis-szerverhez használt nevet és jelszót. Ha ez a fájl nem létezik, akkor automatikusan elindul a telepítő, ahol a hiányzó adatok megadhatók.

install
`index.php`
`surveyart.sql`
	Ez a mappa tartalmazza az adatbázis-beállítások megadására szolgáló telepítőt. A *surveyart.sql* fájl hozza létre a *surveyart* adatbázist, amennyiben nem létezik.

`js`	A kliensoldali felhasználói felületet, és a szerverrel történő kommunikációt kezelő JavaScript állományok.
`style`	A felhasználói felület stílusát megadó CSS állományok.
`connections.php`	Az alrendszerek közötti kapcsolatok grafikus megjelenítéséért felelős PHP állomány.
`database.php`	A kliens számára biztosított különböző szolgáltatásokat tartalmazó PHP forrásfájl.

`help.html`	A súgó szövegét tartalmazó HTML fájl.
`index.php`	A webszerveren alkalmazás indítására szolgáló index fájl, amely betölti az indulásnál szükséges JavaScript modulokat.
`login.php` `logout.php`	A be- és kijelentkezést adminisztráló PHP forrásfájlok.

`messages.html`
`subjects.html`
`subsystems.html`
	Az alkalmazás megfelelő JavaScript moduljait igény szerint betöltő HTML segédfájlok.

3.5.2 Kommunikáció

A kommunikáció a kliens és a szerver között JSON (JavaScript Object Notation) formátumú adatokkal történik HTTP protokollon keresztül. Ennek a megvalósítását a különböző böngészőkben az AJAX hívásokhoz használt XMLHttpRequest (XHR) objektum végzi. A szerveren a *database.php* állomány biztosítja a különböző metódusokat, függvényeket a kliens számára, amelyek eléréséhez a kliens a szervernek egy HTTP POST kérést küld, ahol a *method* paraméterben nevezi meg a kívánt metódust.

Példa:

A kliens által a szerverhez intézett HTTP POST kérés:

```
method=getSubsystems
```

A szerver válasza:

```
{
    "rows":3,
    "subsystems":[{
        "id":"1",
        "type":"0",
        "name":"different groups of people",
        "features":"10",
        "categories":"20"
    },{
        "id":"2",
        "type":"0",
        "name":"financial consequences",
        "features":"10",
        "categories":"20"
    },{
        "id":"3",
        "type":"0",
        "name":"employment consequences",
        "features":"10",
        "categories":"20"
    }]
}
```

3.5.3 A szerveroldal szolgáltatásai

A kliens a *database.php* fájlnak küldött kéréseken keresztül a következő függvényeket, eljárásokat veheti igénybe a szerveren:

`getSubsystems()`

Visszaadja az alrendszerek listáját.

Paraméterek: –

`getSubsystemPairs()`

Lekéri az adatbázisból az alrendszerpárokat.

Paraméterek: –

`addSubsystem(name, type, maxfeatures, maxcategories)`

Felvesz az adatbázisba egy új alrendszert.

Paraméterek:

`name`	szöveg	az alrendszer neve
`type`	egész	az alrendszer típusa: 0 - általános, 1 - értékelő, 2 - döntési
`maxfeatures`	egész	a jellemzők száma (tehát, hány dimenziós)
`maxcategories`	egész	a kialakítható kategóriák maximális száma

`editSubsystem(id, name, type, maxfeatures, maxcategories)`

Módosít az adatbázisban egy már meglévő alrendszert.

Paraméterek:

`id`	egész	a módosítandó alrendszer azonosítója
`name`	szöveg	az alrendszer neve
`type`	egész	az alrendszer típusa: 0 - általános, 1 - értékelő, 2 - döntési
`maxfeatures`	egész	a jellemzők száma (tehát hány dimenziós)
`maxcategories`	egész	a kialakítható kategóriák maximális száma

`deleteSubsystem(id)`

Töröl az adatbázisból egy vagy több alrendszert, ilyenkor törlődik az adatbázisból az összes hozzá tartozó kód-szöveg pár és az összes kapcsolódó üzenet is.

Paraméterek:

`id`	tömb	a törlendő alrendszerek azonosítóinak tömbje

`shrinkSubsystemFeatures(id, col)`

Bizonyos jellemzők törlésével leszűkíti a megadott alrendszer kódjait.

Paraméterek:

`id`	egész	a leszűkítendő alrendszer azonosítója
`col`	tömb	az eltávolítandó jellemzők sorszámát tartalmazó tömb, ahol a számozás 1-el kezdődik

`expandSubsystemFeatures(id, num, col)`

A megadott alrendszer kódvektoraiba új jellemzőket szúr be.

Paraméterek:

`id`	egész	a kibővítendő alrendszer azonosítója
`num`	egész	hány új jellemzőt kell beszúrni
`col`	tömb	annak az oszlopnak a sorszáma ami elé be kell szúrni az új jellemzőket

`getSubjects()`

Az adatbázisból lekérdezi a tesztalanyok listáját.

Paraméterek: –

`getParamsART(id)`

Lekéri az adatbázisból a megadott tesztalany alrendszerekre vonatkozó ART neurális hálózat paramétereit.

Paraméterek:

`id`	egész	a tesztalany azonosítója

`setParamsART(id, ssid, vigilance, learning, choice)`

Frissíti az adatbázisban a megadott tesztalany adott alrendszer(ek)re vonatkozó ART paramétereit.

Paraméterek:

`id`	egész	a tesztalany azonosítója
`ssid`	tömb	az alrendszerek azonosítóit tartalmazó tömb
`vigilance`	valós	vigilancia paraméter
`learning`	valós	tanulási paraméter
`choice`	valós	választási paraméter

`getParamsARAM(id)`

Lekéri az adatbázisból a megadott tesztalany alrendszerpárokra vonatkozó ARAM paramétereit.

Paraméterek:

`id`	egész	a tesztalany azonosítója

`setParamsARAM(id, ssid, vigilance1, learning1, choice1, vigilance2, learning2, choice2, contribution)`

Frissíti az adatbázisban a megadott tesztalany adott alrendszerpárra vonatkozó ARAM paramétereit.

Paraméterek:

`id`	egész	a tesztalany azonosítója
`ssid`	tömb	az alrendszerpárok azonosítóinak tömbje, ahol az alrendszerpárok azonosítója az alábbi képlettel számítható ki: $\frac{j\,(j+1)}{2} + i$, ahol az i a kisebbik azonosítójú alrendszer
`vigilance1`	valós	az ARAM bal oldali moduljának vigilancia paramétere
`learning1`	valós	az ARAM bal oldali moduljának tanulási paramétere
`choice1`	valós	az ARAM bal oldali moduljának választási paramétere
`vigilance2`	valós	az ARAM jobb oldali moduljának vigilancia paramétere
`learning2`	valós	az ARAM jobb oldali moduljának tanulási paramétere
`choice2`	valós	az ARAM jobb oldali moduljának választási paramétere
`contribution`	valós	az ARAM moduljainak súlya, vagyis a részesedési paraméter

`addSubject(name, vigilance, learning, choice, contribution, [count])`

Felvesz egy tesztalanyt az adatbázisba.

Paraméterek:

`name`	szöveg	a tesztalany neve
`vigilance`	valós	az alapértelmezett vigilancia paraméter
`learning`	valós	az alapértelmezett tanulási paraméter
`choice`	valós	az alapértelmezett választási paraméter

`contribution`	valós	az alapértelmezett részesedési paraméter
`count`	egész	opcionális paraméter, a létrehozandó tesztalanyok száma

`editSubject(id, name, vigilance, learning, choice, contribution)`

Módosítja egy adott tesztalany adatait az adatbázisban.

Paraméterek:

`id`	egész	a módosítandó tesztalany azonosítója
`name`	szöveg	a tesztalany neve
`vigilance`	valós	az alapértelmezett vigilancia paraméter
`learning`	valós	az alapértelmezett tanulási paraméter
`choice`	valós	az alapértelmezett választási paraméter
`contribution`	valós	az alapértelmezett részesedési paraméter

`deleteSubject(id)`

Törli a megadott azonosítójú tesztalany(oka)t az adatbázisból.

Paraméterek:

`id`	tömb	a törlendő tesztalanyok azonosítóit tartalmazó tömb

`getMessageGroups()`

Az adatbázisból lekéri az üzenetcsoportok listáját.

Paraméterek: –

`addMessageGroup(name)`

Felvesz az adatbázisba egy új üzenetcsoportot.

Paraméterek:

`name`	szöveg	az üzenetcsoport neve

`editMessageGroup(id, name)`

Módosít az adatbázisban egy már meglévő üzenetcsoportot.

Paraméterek:

`id`	egész	a módosítandó üzenetcsoport azonosítója
`name`	szöveg	az üzenetcsoport neve

`deleteMessageGroup(id)`

Töröl az adatbázisból egy vagy több üzenetcsoportot.

Paraméterek:

`id`	tömb	a törlendő üzenetcsoportok azonosítóit tartalmazó tömb

`getMessages(id)`

Lekéri az adatbázisból az adott azonosítójú üzenetcsoporthoz tartozó üzeneteket.

Paraméterek:

`id`	egész	az üzenetcsoportnak az azonosítója, aminek az üzeneteit szeretnénk megkapni

`addMessage(pos, group, type, source, clause1, clause2)`

Felvesz az adatbázisba egy új üzenetet.

Paraméterek:

`pos`	egész	az üzenet pozíciója az üzenetek között
`group`	egész	az üzenetcsoportnak az azonosítója, amelyikbe szeretnénk felvenni az üzenetet
`type`	egész	az üzenet típusa (1 - egyszerű üzenet, 2 - ok-okozati összefüggés)
`source`	egész	az üzenet forrása (–1 - nem megbízható, 0 - semleges, 1 - megbízható)
`clause1`	egész	az üzenethez tartozó 1. részüzenet azonosítója a *dictionary* adatbázistáblában
`clause2`	egész	az üzenethez tartozó 2. részüzenet azonosítója a *dictionary* adatbázistáblában

`editMessage(id, pos, type, source, clause1, clause2)`

Egy már meglévő üzenet szerkesztése az adatbázisban.

Paraméterek:

`id`	egész	a módosítandó üzenetnek az azonosítója
`pos`	egész	az üzenet pozíciója az üzenetek között
`group`	egész	az üzenetcsoportnak az azonosítója, amelyikbe az üzenetet tartozik
`type`	egész	az üzenet típusa (1 - egyszerű üzenet, 2 - ok-okozati összefüggés)
`source`	egész	az üzenet forrása (–1 - nem megbízható, 0 - semleges, 1 - megbízható)
`clause1`	egész	az üzenethez tartozó 1. részüzenet azonosítója a *dictionary* adatbázistáblában
`clause2`	egész	az üzenethez tartozó 2. részüzenet

azonosítója a *dictionary* adatbázistáblában

`deleteMessage(id)`

Töröl egy vagy több üzenetet az adatbázisból.

Paraméterek:

`id`	tömb	a törlendő üzenetek azonosítóit tartalmazó tömb

`getDictionary([id])`

Lekérdezi az adatbázisból a kód-szöveg párokat.

Paraméterek:

`id`	egész	opcionális paraméter, ha meg van adva akkor csak az azonosítóhoz tartozó alrendszernek a bejegyzéseit adja vissza

`addClause(subsystem, word, code)`

Hozzáad a megadott alrendszerhez egy új kód-szöveg párt.

Paraméterek:

`subsystem`	egész	annak az alrendszernek az azonosítója, amelyikhez hozzá kell adni a bejegyzést
`word`	szöveg	a kód szöveges reprezentációja
`code`	tömb	ez a tömb tartalmazza a kódot

`editClause(id, word, code)`

Módosít az adatbázisban egy kód-szöveg párt.

Paraméterek:

`id`	egész	a szótárbejegyzésnek az azonosítója
`word`	szöveg	a kód szöveges reprezentációja
`code`	tömb	ez a tömb tartalmazza a kódot

`deleteClause(id)`

Kitöröl az adatbázisból egy vagy több szótárbejegyzést, ekkor törlődnek az szótárbejegyzést felhasználó üzenetek is.

Paraméterek:

`id`	tömb	a törlendő szótárbejegyzések azonosítóit tartalmazó tömb

`getMessageQueue(subject, group, overwrite)`

Lekéri az adatbázisból a megtanulandó üzenetek listáját.

Paraméterek:

`subject`	tömb	a tesztalanyok azonosítóit tartalmazó tömb
`group`	tömb	a megtanulandó üzenetcsoportokat tartalmazó tömb
`overwrite`	logikai	ha *IGAZ*, akkor kitörli a paraméterként átadott tesztalanyok eddig megtanult adatait

`learnMessage(subject, id, perfect)`

Megtanít a paraméterként átadott tesztalanyoknak egy üzenetet.

Paraméterek:

`subject`	tömb	a tesztalanyok azonosítóit tartalmazó tömb
`id`	egész	a megtanítandó üzenet azonosítója
`perfect`	logikai	ha igaz, akkor az üzenetet tökéletesen megtanulja, és nem veszi figyelembe az üzenet típusát és forrását

`getDataART(subject, subsystem)`

Lekéri az adatbázisból a megadott tesztalany megadott alrendszerre vonatkozó ART prototípusvektorainak értékeit.

Paraméterek:

`subject`	egész	a tesztalany azonosítója
`subsystem`	egész	az alrendszer azonosítója

`getDataARAM(subject, subsystem1, subsystem2)`

Lekéri az adatbázisból a megadott tesztalany megadott alrendszerpárra vonatkozó ARAM prototípusvektorainak értékeit.

Paraméterek:

`subject`	egész	a tesztalany azonosítója
`subsystem1`	egész	az 1. alrendszer azonosítója
`subsystem2`	egész	a 2. alrendszer azonosítója

`askQuestion(subject, clause, evalSubsystem, choiceSubsystem, vigilance, order)`

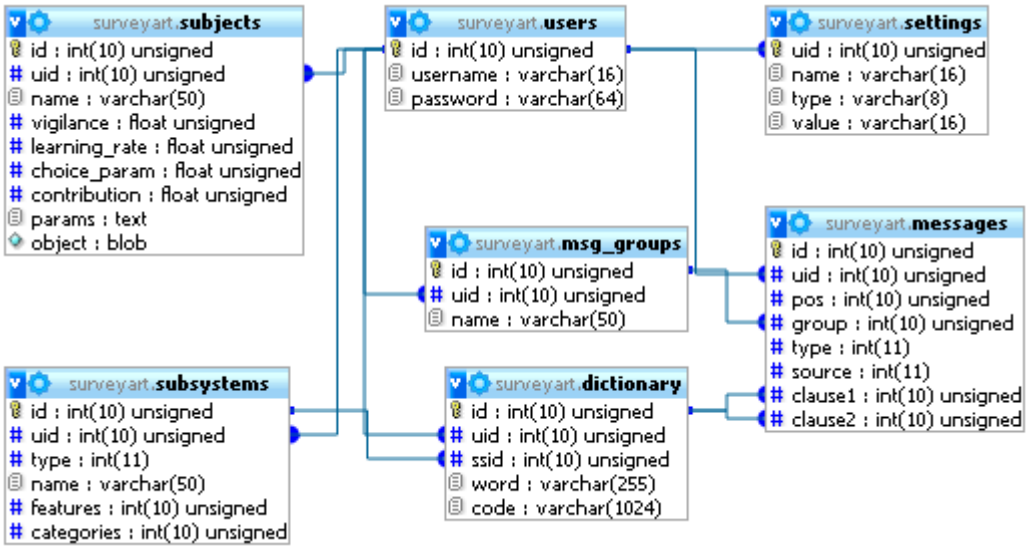
Feltesz a tesztalanynak egy kérdést.

Paraméterek:

`subject`	egész	a tesztalany azonosítója

`clause`	egész	a kérdést alkotó szótárbejegyzés azonosítója
`evalSubsystem`	egész	az értékelésre használandó alrendszer azonosítója
`choiceSubsystem`	egész	a döntésre használandó alrendszer azonosítója
`vigilance`	tömb	a vigilancia paramétereket tartalmazó tömb, amelynek az elemei: 0 - az ART vigilanciája 1 - a döntési alrendszerrel a direkt összeköttetés bal oldali vigilanciája 2 - a döntési alrendszerrel a direkt összeköttetés jobb oldali vigilanciája 3 - az asszociációkeresésnél használt ARAM bal oldali vigilanciája 4 - az asszociációkeresésnél használt ARAM jobb oldali vigilanciája 5 - az értékelő alrendszernél használt ARAM bal oldali vigilanciája 6 - az értékelő alrendszernél használt ARAM jobb oldali vigilanciája 7 - a döntési alrendszernél használt ARAM bal oldali vigilanciája 8 - a döntési alrendszernél használt ARAM jobb oldali vigilanciája

3.5.4 Az alkalmazás adatbázissémája



USERS tábla

Név	Típus	Leírás
id	int(10)	Elsődleges kulcs.
username	varchar(16)	A rendszer felhasználójának neve.
password	varchar(64)	A felhasználó jelszavából előállított hash-függvény, amely a következőképpen jön létre: PASSWORD(MD5(*jelszo*)), ahol a PASSWORD és az MD5 függvény a MySQL beépített függvényei, a *jelszo* pedig a felhasználó által megadott eredeti jelszó.

A *USERS* tábla tartalmazza a rendszer felhasználóit, valamint a belépéshez szükséges jelszó ellenőrzésére használható információkat. Biztonsági megfontolásból a jelszavakat nem tároljuk eredeti formában az adatbázisban, hanem egy hash-függvény segítségével egy kivonatot készítünk belőle. A jelszó ellenőrzésekor a beírt jelszóra is elvégezzük ezt a transzformációt, és ha a beírt jelszóból számított függvényérték megegyezik az adatbázisban tárolttal, akkor a jelszót elfogadjuk.

SETTINGS tábla

Név	Típus	Leírás
uid	int(10)	Elsődleges kulcs, valamint a *USERS* táblára hivatkozó külső kulcs.
name	varchar(16)	A beállítás megnevezése.
type	varchar(8)	A beállítás értékének típusa, amely a következő értékeket veheti fel: *boolean*, *integer*, *float* vagy *string*.
value	varchar(16)	A beállítást meghatározó érték.

A *SETTINGS* tábla tartalmazza a felhasználók különböző beállításait. A beállítások előre definiált név-érték párok, amelyeket a webalkalmazás belső működését befolyásolják. Jelenleg a következő beállítások adhatók meg:

- **divisor** (*float*): Ez a paraméter határozza meg, hogy a Survey ART neurális hálózatban a vigilancia és tanulási paraméter értékeit mennyire befolyásolja az üzenet típusa és a forrás megbízhatósága. A beállítás alapértéke 3, vagyis ilyenkor a tanulási paraméter csökkenésnél β/3-mal fog csökkenni, növekedés esetén pedig (1 – β)/3-mal, ahol β a tanulási paraméter előző értéke.
- **random** (*boolean*): Azt határozza meg, hogy ha az ART és ARAM neurális hálózatokban a maximális aktivációs függvény érték több kategóriára esetén is megegyezik, akkor az első legmagasabbat válassza-e a rendszer, vagy véletlenszerűen

válasszon az egyező értékek közül. Ha a beállítás értéke 0, akkor a rendszer mindig az első legnagyobb aktivációs értékkel rendelkező kategóriát választja, ellenkező esetben pedig a véletlen segítségével választja ki a maximális aktivációs értékű kategóriák közül. Alapesetben a beállítás értéke 0.

- **randomseed** (*integer*): Ha az előző beállítás 1-re van állítva, akkor itt adhatjuk meg azt az értéket, amellyel a véletlenszám-generátort inicializáljuk. Alapértéke: 2008

SUBSYSTEMS tábla

Név	Típus	Leírás
id	int(10)	Elsődleges kulcs.
uid	int(10)	Az alrendszert létrehozó felhasználó azonosítója, a *USERS* táblára hivatkozó külső kulcs.
type	int(11)	Az alrendszer típusa: 0 – *általános*, 1 – *értékelő*, 2 – *döntési*
name	varchar(50)	Az alrendszer megnevezése
features	int(10)	Az alrendszer jellemzőinek maximális száma, ami egyúttal neurális hálózat vektorainak a dimenziószáma.
categories	int(10)	Az alrendszerhez tartozó neurális hálózatban kialakítható kategóriák maximális száma.

A *SUBSYSTEMS* tábla tartalmazza a felhasználók által létrehozott összes alrendszert.

DICTIONARY tábla

Név	Típus	Leírás
id	int(10)	Elsődleges kulcs.
uid	int(10)	A szótárbejegyzést létrehozó felhasználó azonosítója, a *USERS* táblára hivatkozó külső kulcs.
ssid	int(10)	Annak az alrendszernek az azonosítója, amelyikhez ez a szótárbejegyzés tartozik, a *SUBSYSTEMS* táblára hivatkozó külső kulcs.
word	varchar(255)	A kódhoz tartozó szöveges reprezentáció.
code	varchar(1024)	A aktivációs vektor $[a_1, a_2, \ldots, a_n]$ formában.

A *DICTIONARY* tábla tartalmazza az alrendszerekhez tartozó szótárbejegyzéseket, vagyis a kódok bevitelét megkönnyítő kód-szöveg párokat.

MSG_GROUPS tábla

Név	Típus	Leírás
id	int(10)	Elsődleges kulcs.
uid	int(10)	Az üzenetcsoportot létrehozó felhasználó azonosítója, a *USERS* táblára hivatkozó külső kulcs.
name	varchar(50)	Az üzenetcsoport megnevezése.

Az *MSG_GROUPS* tábla az üzenetcsoportokat tartalmazza.

MESSAGES tábla

Név	Típus	Leírás
id	int(10)	Elsődleges kulcs.
uid	int(10)	Az üzenetet létrehozó felhasználó azonosítója, a *USERS* táblára hivatkozó külső kulcs.
pos	int(10)	Az üzenet pozíciója a többi üzenet között. Minél kisebb az értéke az üzenet annál előrébb kerül a többi üzenethez képest.
group	int(10)	Az üzenetcsoportnak az azonosítója, amelybe ez az üzenet tartozik, az *MSG_GROUPS* táblára hivatkozó külső kulcs.
type	int(11)	Az üzenet típusa: 1 – *egyszerű*, 2 – *ok-okozati*
source	int(11)	Az üzenet forrásának megbízhatósága: -1 – *nem megbízható*, 0 – semleges, 1 – *megbízható*
clause1	int(10)	Az üzenet első részét alkotó szótárbejegyzés azonosítója, a *DICTIONARY* táblára hivatkozó külső kulcs.
clause2	int(10)	Az üzenet második részét alkotó szótárbejegyzés azonosítója, a *DICTIONARY* táblára hivatkozó külső kulcs.

A *MESSAGES* táblában vannak az üzenetcsoportokhoz tartozó üzenetek, amelyeket a tesztalanyok a későbbiekben megtanulnak.

SUBJECTS tábla

Név	Típus	Leírás
id	int(10)	Elsődleges kulcs.
uid	int(10)	A tesztalanyt létrehozó felhasználó azonosítója, a *USERS* táblára hivatkozó külső kulcs.
name	varchar(50)	A tesztalany neve.
vigilance	float	A tesztalany alapértelmezett vigilancia paramétere.
learning_rate	float	A tesztalany alapértelmezett tanulási paramétere.
choice_param	float	A tesztalany alapértelmezett választási paramétere.

contribution	float	A tesztalany alapértelmezett részesedési paramétere.
params	text	A tesztalany egyéni alrendszerekre és alrendszerpárokra vonatkozó paraméterei JSON formátumban.
object	blob	A tesztalany Survey ART objektumát tartalmazó mező, a PHP *serialize()* függvénye által előállított szerializált formában.

A *SUBJECTS* táblában vannak a tesztalanyok, a hozzájuk tartozó különböző neurális háló paraméterek, valamint a szerializált formában tárolt Survey ART objektum.

5.3.5 Az algoritmusok implementációja

A Fuzzy ART, Fuzzy ARAM és a Survey ART neurális hálózatokat a PHP5 objektumorientált eszközeit felhasználva programoztuk. Az alábbiakban pszeudokóddal megadjuk az implementáció részleteit.

Fuzzy ART

A Fuzzy ART osztályt a *classes\FuzzyART.class.php* állomány tartalmazza. Az osztály az alábbi publikus metódusokkal rendelkezik:

`learn(I)`

Osztályozza és megtanulja a megadott *I* input vektort.

Paraméterek:

`I`	tömb	a megtanulandó kódvektor

`classify(I, prototype)`

Osztályozza a megadott *I* input vektort anélkül, hogy a neurális hálót módosítaná.

Paraméterek:

`I`	tömb	az osztályozandó kódvektor
`prototype`	logikai	ha *HAMIS*, akkor a metódus a kategória sorszámát adja meg, amit a *getPrototype* metódussal lehet prototípusvektorrá alakítani, ha *IGAZ*, akkor közvetlenül a prototípusvektort adja vissza

`getChoiceParam()`

Ezzel a metódussal kérdezhető le a *választási paraméter* értéke.

Paraméterek: –

`setChoiceParam(choice)`

Ezzel a metódussal állítható be a *választási paraméter* értéke.

Paraméterek:

`choice`	valós	az új választási paraméter

`getLearningRate()`

Ezzel a metódussal kérdezhető le a *tanulási paraméter* értéke.

Paraméterek: –

`setLearningRate(learning)`

Ezzel a metódussal állítható be a *tanulási paraméter* értéke.

Paraméterek:

`learning`	valós	az új tanulási paraméter

`getVigilance()`

Ezzel a metódussal kérdezhető le a *vigilancia paraméter* értéke.

Paraméterek: –

`setVigilance(vigilance)`

Ezzel a metódussal állítható be a *vigilancia paraméter* értéke.

Paraméterek:

`vigilance`	valós	az új vigilancia paraméter

`getParams()`

Ezzel a metódussal egyszerre kérdezhető le a *vigilancia, tanulási* és *választási paraméter*. A visszatérési értéke egy tömb amely a paramétereket az előbb megadott sorrendben tartalmazza.

Paraméterek: –

`setParams(vigilance, learning, choice)`

Ezzel a metódussal egyszerre állítható be a *vigilancia, tanulási* és *választási paraméter*.

Paraméterek:

`vigilance`	valós	az új vigilancia paraméter
`learning`	valós	az új tanulási paraméter

`choice`	valós	az új választási paraméter

`getPrototype(category)`

Megadja a kategória sorszámához tartozó prototípusvektort.

Paraméterek:

`category`	egész	a kategória sorszáma

`isCommitted(category)`

Megadja, hogy az adott sorszámú kategória foglalt-e.

Paraméterek:

`category`	egész	a kategória sorszáma

A tanítás és az osztályozás működése pszeudokóddal:

M	a kategóriák maximális száma
N	a prototípusvektorok dimenziója
súlyok	egy M×N-es mátrix, amely a kategóriákat tartalmazza
foglalt	egy M dimenziós logikai értékeket tartalmazó tömb, amely megadja, hogy melyik kategória foglalt
α	a választási paraméter
β	a tanulási paraméter
ρ	a vigilancia paraméter

(Ezek a változók az osztály mezői, így bármelyik metódusból közvetlenül elérhetőek.)

```
TANÍT(I)
 1  aktiváció ← AKTIVÁL(I)
 2  while IGAZ
 3        do J ← GYŐZTESMINDENTVISZ(aktiváció)
 4           if J = –1
 5              then return NIL
 6           if REZONANCIA(I, J)
 7             then KÓDOL(I, J)
```

```
8                    return súlyok[J]
9                else aktiváció[J] ← –1
```

OSZTÁLYOZ(I)

```
1  régi ← β
2  β ← 0
3  prototípus ← TANÍT(I)
4  β ← régi
5  return prototípus
```

AKTIVÁL(I, M, N, súlyok, α)

```
1  eredmény ← NIL
2  for i ← 1 to M
3      do eredmény[i] ← |I∧súlyok[i]| / (|súlyok[i]|+α)
4  return eredmény
```

GYŐZTESMINDENTVISZ(aktiváció)

```
1  eredmény ← 1
2  for i ← 2 to M
3      do if aktiváció[i] > aktiváció[eredmény]
4            then eredmény ← i
5  return eredmény
```

REZONANCIA(I, J)

```
1  return |I∧súlyok[J]| / |I| > ρ
```

KÓDOL(I, J)

```
1  if foglalt[J] = IGAZ
2     then súlyok[J] ← β(I ∧ súlyok[J]) + (1 – β)súlyok[J]
3     else súlyok[J] ← I ∧ súlyok[J]
4          foglalt[J] ← IGAZ
```

Fuzzy ARAM

A Fuzzy ARAM osztályt a *classes\FuzzyARAM.class.php* állomány tartalmazza. Az osztály az alábbi publikus mezőkkel és metódusokkal rendelkezik:

`A`

Az *A* mező segítségével érthető el az ARAM neurális hálózat A modulja.

`B`

Az *B* mező segítségével érthető el az ARAM neurális hálózat B modulja.

`learn(IA, IB)`

Megtanulja a megadott *IA* és *IB* input vektor közötti asszociációt.

Paraméterek:

`IA`	tömb	a megtanulandó asszociáció első fele
`IB`	tömb	a megtanulandó asszociáció második fele

`associate(IA, IB, prototype)`

Asszociációt keres az *AI* és *IB* vektor között, anélkül, hogy a neurális háló módosulna.

Az *IA* és *IB* paraméterek közül nem lehet mindkettő NULL.

Paraméterek:

`IA`	tömb	az asszociáció első fele vagy NULL
`IB`	tömb	az asszociáció második fele vagy NULL
`prototype`	logikai	ha *HAMIS*, akkor a metódus a kategória sorszámát adja meg, amit a *getPrototype* metódussal lehet prototípusvektorrá alakítani, ha *IGAZ*, akkor közvetlenül a prototípusvektort adja vissza

`getContribution()`

Ezzel a metódussal kérdezhető le a *részesedési paraméter* értéke.

Paraméterek: –

`setContribution(contribution)`

Ezzel a metódussal állítható be a *részesedési paraméter* értéke.

Paraméterek:

`contribution`	valós	az új részesedési paraméter

`getParams()`

Ezzel a metódussal egyszerre kérdezhető le a *vigilancia, tanulási, választási* és a *részesedési paraméter*. A visszatérési értéke egy tömb amely a paramétereket az előbb megadott sorrendben tartalmazza.

Paraméterek: –

`setParams(vigilance, learning, choice, contribution)`

Ezzel a metódussal egyszerre állítható be a *vigilancia, tanulási, választási* és *részesedési paraméter*.

Paraméterek:

`vigilance`	valós	az új vigilancia paraméter
`learning`	valós	az új tanulási paraméter
`choice`	valós	az új választási paraméter
`contribution`	valós	az új részesedési paraméter

`getPrototype(category)`

Megadja a kategória sorszámához tartozó prototípusvektort.

Paraméterek:

`category`	egész	a kategória sorszáma

`isCommitted(category)`

Megadja, hogy az adott sorszámú kategória foglalt-e.

Paraméterek:

`category`	egész	a kategória sorszáma

A tanítás és az asszociáció működése pszeudokóddal:

M_A, M_B	a kategóriák maximális száma az A és a B modulban
N_A, N_B	a prototípusvektorok dimenziója az A és a B modulban
súlyok$_A$, *súlyok*$_B$	egy $M_A \times N_A$-s és egy $M_B \times N_B$-s mátrix, amely az A és B modul kategóriáit tartalmazza
foglalt$_A$, *foglalt*$_B$	egy M_A és egy M_B dimenziós, logikai értékeket tartalmazó tömb, amely megadja, hogy melyik kategória foglalt az A és a B modulban
α_A, α_B	az A és a B modul választási paramétere

β_A, β_B	az A és a B modul a tanulási paramétere
ρ_A, ρ_B	az A és a B modul a vigilancia paramétere
γ	az A és B modul közötti részesedési paraméter

(Ezek a változók az osztály mezői, így bármelyik metódusból közvetlenül elérhetőek.)

TANÍT(I_A, I_B)

```
1   aktiváció ← AKTIVÁL(I_A, I_B)
2   while IGAZ
3         do J ← GYŐZTESMINDENTVISZ(aktiváció)
4            if J = –1
5               then return NIL
6            if REZONANCIA(I_A, I_B, J)
7              then KÓDOL(I_A, I_B, J)
8                   return súlyok_A[J], súlyok_B[J]
9              else aktiváció[J] ← –1
```

ASSZOCIÁL(I_A, I_B)

```
1   if I_A = NIL és I_B = NIL
2     then return NIL
3   if I_A ≠ NIL és I_B = NIL
4     then return A.OSZTÁLYOZ(I_A), I_B
5   if I_A = NIL és I_B ≠ NIL
6     then return I_A, B.OSZTÁLYOZ(I_B)
7   régi_A ← β_A
8   régi_B ← β_B
9   β_A ← 0
10  β_B ← 0
11  prototípus_A, prototípus_B ← TANÍT(I_A, I_B)
12  β_A ← régi_A
13  β_B ← régi_B
14  return prototípus_A, prototípus_B
```

AKTIVÁL(I_A, I_B)

```
1  eredmény ← NIL
2  for i ← 1 to MIN(M_A, M_B)
3      do eredmény[i] ← γ |I_A ∧ súlyok_A[i]| / (|súlyok_A[i]| + α_A) + (1 − γ) |I_B ∧ súlyok_B[i]| / (|súlyok_B[i]| + α_B)
4  return eredmény
```

GYŐZTESMINDENTVISZ(aktiváció)

```
6  eredmény ← 1
7  for i ← 2 to MIN(M_A, M_B)
8      do if aktiváció[i] > aktiváció[eredmény]
9           then eredmény ← i
10 return eredmény
```

REZONANCIA(I_A, I_B, J)

```
1  return |I_A ∧ súlyok_A[J]| / |I_A| > ρ_A  és  |I_B ∧ súlyok_B[J]| / |I_B| > ρ_B
```

KÓDOL(I_A, I_B, J)

```
1  if foglalt_A[J] = IGAZ  és  foglalt_B[J] = IGAZ
2    then súlyok_A[J] ← β_A(I_A ∧ súlyok_A[J]) + (1 – β_A)súlyok_A[J]
3         súlyok_B[J] ← β_B(I_B ∧ súlyok_B[J]) + (1 – β_B)súlyok_B[J]
4    else súlyok_A[J] ← I_A ∧ súlyok_A[J]
5         foglalt_A[J] ← IGAZ
6         súlyok_B[J] ← I_B ∧ súlyok_B[J]
7         foglalt_B[J] ← IGAZ
```

Survey ART

A Survey ARAM osztályt a *classes\SurveyART.class.php* állomány tartalmazza. Az osztály az alábbi publikus mezőkkel és metódusokkal rendelkezik:

`art`

Az osztály *art* mezője egy tömb, amelynek segítségével elérthetők az ART hálózatok. A tömb indexei a *SUBSYSTEMS* adatbázistábla *id* mezői.

`aram`

Az osztály *aram* mezője egy tömb, amelynek segítségével elérhetők az ARAM hálózatok. A tömb indexei a *SUBSYSTEMS* adatbázistábla *id* mezőiből a következőképpen számolhatók: $\frac{j(j+1)}{2} + i$, ahol i és j két alrendszer azonosítója az adatbázisban, és $i \leq j$.

`learn(type, source, message)`

Megtanulja az adott üzenetet.

Paraméterek:

`type`	egész	az üzenet típusa (0 - tökéletes tanulás, 1 - egyszerű üzenet, 2 - ok-okozati összefüggés)
`souce`	egész	az üzenet forrása (–1 - nem megbízható, 0 - semleges, 1 - megbízható), ha a *type* értéke 0, akkor nincsen jelentősége
`message`	tömb	az üzenetet tartalmazó tömb a következő struktúrában:

```
[
  [
    subsystemA,
    [
      [codeA, ρ, β, α],
      ρA, βA, αA
    ]
  ],
  [
    subsystemB,
    [
      [codeB, ρ, β, α],
      ρB, βB, αB
    ],
    Y
  ]
]
```

`ask(ssid, code, general, eval, choice, vigilance, params)`

Feltesz egy kérdést a neurális hálózatnak.

Paraméterek:

`ssid`	egész	annak az alrendszernek az azonosítója, amelyikhez intézzük a kérdést
`code`	tömb	a kérdés kódvektora
`eval`	egész	az értékelő alrendszer azonosítója
`choice`	egész	a döntési alrendszer azonosítója
`vigilance`	tömb	a vigilancia paramétereket tartalmazó tömb, amelynek az elemei: 0 - az ART vigilanciája

		1 - a döntési alrendszerrel a direkt összeköttetés bal oldali vigilanciája 2 - a döntési alrendszerrel a direkt összeköttetés jobb oldali vigilanciája 3 - az asszociációkeresésnél használt ARAM bal oldali vigilanciája 4 - az asszociációkeresésnél használt ARAM jobb oldali vigilanciája 5 - az értékelő alrendszernél használt ARAM bal oldali vigilanciája 6 - az értékelő alrendszernél használt ARAM jobb oldali vigilanciája 7 - a döntési alrendszernél használt ARAM bal oldali vigilanciája 8 - a döntési alrendszernél használt ARAM jobb oldali vigilanciája
`params`	tömb	az alrendszerspecifikus paramétereket tartalmazó tömb

A tanítás és a kikérdezés működése pszeudokóddal:

TANÍT(típus, forrás, alrendszer$_A$, kód$_A$, ρ_A^{ART}, β_A^{ART}, α_A^{ART}, ρ_A^{ARAM}, β_A^{ARAM}, α_A^{ARAM},
alrendszer$_B$, kód$_B$, ρ_B^{ART}, β_B^{ART}, α_B^{ART}, ρ_B^{ARAM}, β_B^{ARAM}, α_B^{ARAM}, γ)

1 **if** típus = 0
2 **then** $\rho_A^{ART} \leftarrow \rho_A^{ARAM} \leftarrow \rho_B^{ART} \leftarrow \rho_B^{ARAM} \leftarrow 1$
3 $\beta_A^{ART} \leftarrow \beta_A^{ARAM} \leftarrow \beta_B^{ART} \leftarrow \beta_B^{ARAM} \leftarrow 0$
4 **elseif** típus = 1
5 **then if** forrás = 1
6 **then** $\beta_A^{ARAM} \leftarrow \beta_A^{ARAM} + \frac{1-\beta_A^{ARAM}}{DIVISOR}$
7 $\beta_B^{ARAM} \leftarrow \beta_B^{ARAM} + \frac{1-\beta_B^{ARAM}}{DIVISOR}$
8 **elseif** forrás = –1
9 **then** $\beta_A^{ARAM} \leftarrow \beta_A^{ARAM} - \frac{\beta_A^{ARAM}}{DIVISOR}$
10 $\beta_B^{ARAM} \leftarrow \beta_B^{ARAM} - \frac{\beta_B^{ARAM}}{DIVISOR}$
11 **elseif** típus = 2

12 **then** $\rho_A^{ARAM} \leftarrow \rho_A^{ARAM} + \frac{1-\rho_A^{ARAM}}{DIVISOR}$

13 **if** forrás = 1

14 **then** $\beta_A^{ARAM} \leftarrow \beta_A^{ARAM} + \frac{1-\beta_A^{ARAM}}{DIVISOR}$

15 $\beta_B^{ARAM} \leftarrow \beta_B^{ARAM} + \frac{1-\beta_B^{ARAM}}{DIVISOR}$

16 **elseif** forrás = –1

17 **then** $\beta_A^{ARAM} \leftarrow \beta_A^{ARAM} - \frac{\beta_A^{ARAM}}{DIVISOR}$

18 $\beta_B^{ARAM} \leftarrow \beta_B^{ARAM} - \frac{\beta_B^{ARAM}}{DIVISOR}$

19 art[alrendszer$_A$].α ← α_A^{ART}

20 art[alrendszer$_A$].β ← β_A^{ART}

21 art[alrendszer$_A$].ρ ← ρ_A^{ART}

22 prototípus$_A$ ← art[alrendszer$_A$].TANÍT(kód$_A$)

23 art[alrendszer$_B$].α ← α_B^{ART}

24 art[alrendszer$_B$].β ← β_B^{ART}

25 art[alrendszer$_B$].ρ ← ρ_B^{ART}

26 prototípus$_B$ ← art[alrendszer$_B$].TANÍT(kód$_B$)

27 **if** alrendszer$_A$ > alrendszer$_B$

28 **then** tmp ← alrendszer$_A$

29 alrendszer$_A$ ← alrendszer$_B$

30 alrendszer$_B$ ← tmp

31 tmp ← prototípus$_A$

32 prototípus$_A$ ← prototípus$_B$

33 prototípus$_B$ ← tmp

34 index ← $\frac{alrendszer_B(alrendszer_B+1)}{2} + alrendszer_A$

35 aram[index].A.α ← α_A^{ARAM}

36 aram[index].A.β ← β_A^{ARAM}

37 aram[index].A.ρ ← ρ_A^{ARAM}

38 aram[index].B.α ← α_B^{ARAM}

39 aram[index].B.β ← β_B^{ARAM}

40 aram[index].B.ρ ← ρ_B^{ARAM}

```
41  aram[index]. γ ← γ
42  aram[index].TANÍT(prototípus_A, prototípus_B)

KÉRDEZ(alrendszer, kód, általános, értékelő, döntési, ρ1, ρ2, ρ3, ρ4, ρ5, ρ6, ρ7, ρ8, ρ9)
 1  art[alrendszer].ρ ← ρ1
 2  prototípus ← art[alrendszer].OSZTÁLYOZ(kód)
 3  válasz ← ASSZOCIÁCIÓ(alrendszer, prototípus, döntési, ρ2, ρ3)
 4  if válasz ≠ NIL
 5     then return válasz
 6  for i ← 1 to hossz[általános]
 7       do asszociációk[i] ← ASSZOCIÁCIÓ(alrendszer, prototípus, i, ρ4, ρ5)
 8          értékelések[i] ← ASSZOCIÁCIÓ(i, asszociációk[i], értékelő, ρ6, ρ7)
 9  kombinált ← ÁTLAG(értékelések)
10  válasz ← ASSZOCIÁCIÓ(értékelő, kombinált, döntési, ρ8, ρ9)
11  return válasz

ASSZOCIÁCIÓ(alrendszer_A, kód_A, alrendszer_B, ρ_A, ρ_B)
 1  kód_B ← NIL
 2  if alrendszer_A > alrendszer_B
 3     then csere ← IGAZ
 4          tmp ← alrendszer_A
 5          alrendszer_A ← alrendszer_B
 6          alrendszer_B ← tmp
 7          tmp ← kód_A
 8          kód_A ← kód_B
 9          kód_B ← tmp
10  index ← alrendszer_B(alrendszer_B + 1)/2 + alrendszer_A
11  aram[index].A.ρ ← ρ_A
12  aram[index].B.ρ ← ρ_B
13  prototípus_A, prototípus_B ← aram[index].ASSZOCIÁL(kód_A, kód_B)
14  if csere = IGAZ
15     then tmp ← prototípus_A
```

```
16          prototípus_A ← prototípus_B
17          prototípus_B ← tmp
18  return prototípus_B
```

4. Összefoglalás

A diplomamunkám első részében röviden betekinthettünk az adaptív rezonancia elméletére épülő neurális hálózatok működésébe. A tanulás során ezek a hálózatok a korábban megtanult információkat úgy védik meg, hogy a inputot összehasonlítják a már meglévő mintákkal, és a meglévő kategóriák súlyai csak akkor módosulnak, ha ez a hasonlóság elér egy bizonyos küszöbszintet, vagyis bekövetkezik egy rezonáns állapot. Amennyiben a bemenet egyik meglévő kategóriára sem hasonlít, akkor a bemeneti minta számára egy teljesen új kategória alakul ki.

Egy gyakorlati alkalmazáson keresztül megmutattuk, hogy az adaptív rezonancia elméletére épülő neurális hálózatok igen sokoldalú modellezőeszközök. A nem felügyelt tanulású Fuzzy ART, a heteroasszociatív Fuzzy ARAM, és az ezeket a hálózatokat felhasználó Survey ART segítségével akár olyan bonyolult szimulációk is végezhetők, amelyek az emberi észlelés, gondolkodás természetét vizsgálják. A többi neurális hálózat modellhez képes óriási előnyt jelent, hogy az adaptív rezonancia elméletére épülő neurális hálózatok esetén a létrejövő kategóriák részletezettsége a vigilancia paraméter segítségével igen finoman állítható.

A programot webalkalmazás formájában készítettük el, amit egyrészt a mai Internetes kor követelményei is megkívántak, másrészt leegyszerűsíti az alkalmazás gyors frissítését és továbbfejlesztését anélkül, hogy a felhasználónak ez bármilyen terhet jelentene. A diplomamunka második részében ismertettük a szoftver működését és implementációjának részleteit, a fejlesztői leírásban pedig mindenre kiterjedő referenciát adtunk, ami elősegíti a program karbantartását és megkönnyíti a további funkciók hozzáadását.

A probléma komplexitása ellenére egy könnyen használható, barátságos felhasználói felülettel rendelkező fejlett rendszert sikerült létrehozni, amely remélhetőleg nagy segítséget nyújt az ismertetett neurális hálózatok tanulmányozásában és továbbfejlesztésében.

Irodalomjegyzék

[1] Carpenter G. A., Grossberg, S., Rosen, D. B. (1991). Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System. *Neural Networks, Vol. 4*, 759-771. oldal.

[2] Tan, A.-H. (1995). Adaptive Resonance Associative Map. *Neural Networks, Vol. 8, No. 3*, 437-446. oldal.

[3] Münnich, Á., Saris, W. (2007). A simulation model for automatic information processing and response behaviour *(kiadatlan kézirat)*.

[4] Fausett, L. (1994). Fundamentals of Neural Networks: Architectures, Algorithms and Applications. *Prentice Hall*, 3-38, 218-287. oldal, ISBN 0-13-334186-0.

[5] McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, 7*, 115 - 133. oldal

[6] Rosenblatt, Frank (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6*, 386-408. oldal

[7] comp.ai.neural-nets FAQ, Part 1 of 7: Introduction
http://www.faqs.org/faqs/ai-faq/neural-nets/part1/

[8] Arbib, M. A. (szerk.) (2002). The Handbook of Brain Theory and Neural Networks. *MIT Press*, 19-25, 87-90. oldal, ISBN 0-262-01197-2.

[9] Weenink, D. (1997). Category ART: A Variation on Adaptive Resonance Theory Neural Networks. *21st Proceedings of the Institute of Phonetic Sciences, University of Amsterdam*, 117-129. oldal.

[10] Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science (publikáció), 11*, 23-63. oldal

[11] Carpenter, G.A. & Grossberg, S. (1987). ART 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics, 26(23)*, 4919-4930. oldal

[12] Carpenter, G.A. & Grossberg, S. (1990). ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks (publikáció), 3*, 129-152. oldal

[13] Carpenter, G.A., Grossberg, S., & Reynolds, J.H. (1991). ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks (publikáció), 4*, 565-588. oldal

[14] Retter, Gy. (2006). Fuzzy, neurális, genetikus, kaotikus rendszerek. *Akadémiai Kiadó*, 338-339. oldal, ISBN 963-05-8353-4.

[15] Arbib, M. A. (szerk.) (2002). The Handbook of Brain Theory and Neural Networks. *MIT Press*, 19-25, 87-90. oldal, ISBN 0-262-01197-2.

[16] Weitzenfeld, A., Arbib, M.A, Alexander, A. (2002). The Neural Simulation Language: A System for Brain Modeling. *MIT Press*, 158-169. oldal, ISBN 0-262-73149-5.

[17] Du, K.-L., Swamy, M. N. S. (2006). Neural Networks in a Softcomputing Framework, *Springer*, 207-215. oldal, ISBN 1-84628-302-7.