Artificial Neural Network What-If Theory

Article · October 2015

DOI: 10.4018/IJISSC.2015100104

2 authors:

Massimo Buscema
University of Colorado
268 PUBLICATIONS **2,519** CITATIONS

SEE PROFILE

William Tastle
Ithaca College
64 PUBLICATIONS **383** CITATIONS

SEE PROFILE

Artificial Neural Network What-If Theory

Paolo Massimo Buscema, Semeion Research Institute, Rome, Italy & Department of Mathematical and Statistical Sciences, CCMB, University of Colorado, Denver, CO, USA

William J Tastle, Ithaca College, Ithaca, NY, USA

ABSTRACT

Data sets collected independently using the same variables can be compared using a new artificial neural network called **A***rtificial neural network* **W***hat* **I***f* **T***heory, AWIT. Given a data set that is deemed the standard reference for some object, i.e. a flower, industry, disease, or galaxy, other data sets can be compared against it to identify its proximity to the standard. Thus, data that might not lend itself well to traditional methods of analysis could identify new perspectives or views of the data and thus, potentially new perceptions of novel and innovative solutions. This method comes out of the field of artificial intelligence, particularly artificial neural networks, and utilizes both machine learning and pattern recognition to display an innovative analysis.*

Keywords: Artificial Intelligence, Artificial Neural Network, Auto-Encoders, Data Set, What-If Theory

INTRODUCTION

A neural network typically takes a single set of data, partitions it into two non-overlapping subsets, and uses one subset to train the neural network such that the underlying behaviors of the data are identified while not overly training it such that the "noise" is treated as a component of the behavior. The other dataset is then inputted though the network to identify the actual patterns, outcomes, etc. based on the previously identified ideal behavior. In this way relationships can be discovered in large medical datasets, corporate transaction databases, law enforcement databases, and the like. In each case it is the single ideal or principal dataset itself that is used to identify relationships.

It is postulated in this paper that the method of artificial neural network (ANN) analysis can be extended to identify behaviors that could be identified as an approximation to another dataset. That is to say, given two separate datasets composed of identical variables for which data has been collected at another time, or using some different form of data collection, or data collected from a different population, how close is one dataset to the other. An example of the benefit of using this kind of ANN analysis is to consider the variables associated with a particular disease. Experts have identified that a particular dataset is composed of the characteristics that typify the disease under study. In another dataset that is comprised of identical variables, the data collected

DOI: 10.4018/IJISSC.2015100104

from a different population that might have been given some form of treatment. Using this new method, it will be shown that it is possible to determine the proximity of the treated group with the classical dataset.

In short, using one dataset that is defined as being the ideal standard containing the relationships necessary to measure desired outcomes, another dataset can be compared to determine its degree of closeness. This opens up the possibility of providing one population with some special effect or treatment as in "what if we do *x* with/to this population?" We can determine the degree of closeness of the second or treated dataset with the original standard. It is in this spirit of performing "what-if" analysis that this method is called **A**rtificial Neural Network **W**hat **I**f **T**heory (AWIT).

GENERAL THEORY

Using an auto-encoder ANN we will approximate the implicit function of any dataset during the learning phase and to assign a fuzzy output to any new input during the recall phase. A fuzzy output is a value in the range [0..1] in which zero means complete absence or non-membership in the output and a one means complete membership. Any other value indicates the degree of partial membership.

Recent research has improved the auto-encoder ANNs in order to optimize a deep learning process (Hinton, Osindero & Teh, 2006) or to select the fundamental hidden features of a large dataset (Le, et. al., 2012) or to reduce the dimensionality of data (Hinton & Salakhutdinov, 2006; Raina, Madhavan & Ng, 2009; Raiko, Valpola & LeCun, 2012). Other approaches have tried to use auto-encoders as unsupervised ANNs able to perform supervised tasks (Larochelle & Bengio, 2008; Bengio, 2009).

We have chosen to look at the auto-encoders from a different point of view: we define the testing phase of a trained auto-encoder as **the interpretation** of a dataset (traditionally referred to as the Testing Dataset) using the logic present in *another* dataset (the Training Dataset). We define this point of view a seminal approach for a new theory, named AWIT (**A**rtificial Neural Networks **W**hat **I**f **T**heory).

The algorithm to implement this new approach to data analysis follows:

- Let DB1 and DB2 be two different datasets with the same types of variables but possessing different records;
- The function f() is a non-linear function optimally interpolating DB1 by means of an auto-encoder ANN consisting of one hidden layer:

$$x_{DB1} = f(h, v^*) \tag{1}$$

$$h = g(x_{DB1}, w^*) \tag{2}$$

  v^*, w^* = parameters to be estimated by ANN,
  x = input variables.
- The dataset DB2 is rewritten using the ANN trained on the DB1dataset:

$$x'_{DB2} = f\left(h, v\right) \tag{3}$$

$$h = g(x_{DB2}, w) \tag{4}$$

v, w = Trained parameters estimated b ANN using DB1
x' = output variables.

The output x'_{DB2} represents how each variable of the DB2 dataset is reformulated using the "logic" of the DB1 dataset.

Figure 1 shows a prototype of the auto-encoder ANN.

The Conceptual Meaning of AWIT

For purposes of illustration, consider a dataset DB1 composed of four variables describing the length and width of petals and sepals of M flowers. For many external reasons we know that the sizes of these flowers are very precise. Consequently, we assume this dataset to be the "gold standard" or "idealized dataset" for this type of flower.

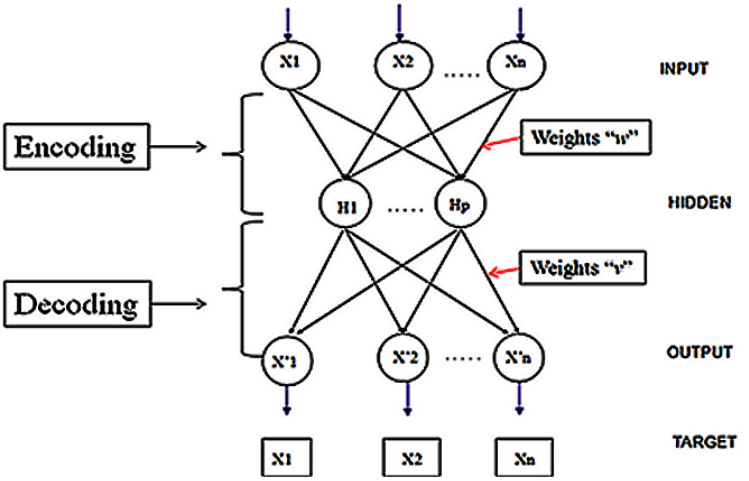
Now consider a completely separate dataset DB2 composed of another sample, P, of the same type of flower, but this time we do not know the degree to which their sizes (length and width of petals and sepals) correct reflect the standard. Perhaps in this case there is some suspicion that P is a variation of M and a botanist desires to how close or similar to M is the sample P.

Therefore, we train an auto-encoder neural network to define the *hyper-surface* of the DB1 dataset. A hyper-surface is a generalization of the concept of a hyperplane. The auto-encoder is able to approximate the many-to-many associations among the variables and after the training phase is complete we know the position of each record contained in the gold standard DB1 dataset on the hyper-surface, with an error approximating zero.

At this point we are able to describe to what degree each flower contained in the DB2 dataset approximates the hyper-surface determined by the DB1 ideal dataset.

In fact, if we use as input each record of the test DB2 dataset into the trained auto-encoder and compare the resulting input vector with the correspondent output vector, we have an expected outcome: the greater the similarity of the output from the input, the greater the likelihood that the

Figure 1. Multi Layer Perception Auto Excoder



flowers of the test DB2 dataset possess the same logic (numerical distribution of its variables) of the flowers coming from idealized dataset. This is also apparent in the reverse: the greater the differences between the output and the input, the greater the flowers of the test DB2 dataset have a different numerical distribution among its variables with respect to the flowers of the idealized dataset.

The trained auto-encoder, in fact, tries to reformulate every new input in terms of an output vector that is more compatible with the sizes of the flowers that was learned from the idealized dataset. In other words, the auto-encoder will be forced to identify a new input on its hyper-surface defined during its previous learning phase.

The differences between each input and each output of the test flower dataset shows the degree to which the auto-encoder has found regularity or dissonance in the new record with respect to its relationship with the idealized dataset. In addition, if we are reasonably sure that the dataset DB1 is well done and the numerical values of its records are correct, we can use the trained auto-encoder to reformulate and correct the sizes of each flower in DB2.

This opens the possibility of calculating a distance measure whereby the proximity of an unknown dataset from an idealized dataset can be determined, perhaps to the point of providing a percentage of accuracy for a particular target.

AWIT Applied to a Toy Example

Let us image a toy dataset representing the terms (the first two inputs) and the results (the third input) of some summations. We have generated 10,000 random examples of correct floating point numbers between 0 and 20 (see Table 1). Now we train an ANN auto-encoder to learn this dataset in order to abstract the implicit logic contained in the summations (column C).

After the training phase:

1. We resubmit each training record as an input vector to the ANN to verify how much of its output is correctly predicted.
2. To the trained ANN we now input two new records whose summation is intentionally wrong:

```
2   1   4
1   3   6
```

Clearly, 2 + 1 = 4 (?) and 1 + 3 = 6 (?) are incorrect. In this case it is important for the ANN to reformulate the output according to the logic contained in the summation column that it had previously learned.

The auto-encoder was trained for 14 000 epochs (RMSE=0.00322813) and with 12 hidden units. When the same two incorrect summations were again inputted to the ANN correction, the output was very precise (see Table 2).

Table 1. Legend: "A B C" means "A + B = C"

Incorrect Input			Correct ANN Output			Real Sum	Approx Error
A+	B=	C	A +	B =	C		
2	1	4	2.256485	1.156551	3.359304	3.413036	-0.0537318
1	3	6	1.43497	3.353027	4.775997	4.787997	-0.01199973

Table 2. A view of the dataset of 10,000 correct summations

Records	A+	B=	C
1	0.666925	2.62489	3.291815
2	9.548238	4.761469	14.30971
3	8.046946	0.790004	8.836949
4	6.637369	1.240659	7.878027
5	7.869632	6.041184	13.91082
6	9.22611	0.947578	10.17369
7	8.82944	5.420059	14.2495
8	6.814585	3.506861	10.32145
9	7.677471	1.798331	9.475801
10	1.412681	1.013844	2.426525
11	3.752713	0.284032	4.036744
12	8.321301	5.690818	14.01212
13	6.013295	7.127772	13.14107
14	9.413612	4.33E-02	9.456888
15	2.567739	1.128992	3.696731
16	8.916303	9.516163	18.43246
…	…	…	…
10000	2.706205	7.542206	10.24841

Briefly, in this example the ANN reformulates the terms and the results of the implicit summation in order to generate a more precise result. The ANN works in this way only because it is driven by the previous experience derived from the correct training dataset. We can say that the ANN in this case has also been shown to make simple abstractions.

More generally we can say that an auto-encoder ANN is able to learn the implicit logic of its training dataset and it is also able to adapt to this logic new testing data.

AWIT applied to Nursery dataset. A nursery database is provided from UCI and was derived from a hierarchical decision model originally developed to rank applications for nursery schools. It was used during several years in 1980's when there was excessive enrollment in these schools in Ljubljana, Slovenia, and the rejected applications frequently needed an objective explanation. The final decision depended on three sub-problems:

1. Occupation of parents and b. the child's nursery,
2. Family structure and financial standing, and
3. The social and health picture of the family.

The model was developed within expert system shell for decision making DEX[1].

It is a classic dataset for use in classification analysis: 8 attributes, 5 classes and 12 960 records.

The 8 attributes are expressed as **true** versus **false** values using crisp logic (see Table 3).

The five classes have the records distribution shown in Table 4.

We unfold the attributes into 27 separate input variables and consider the five classes to be an input extension of the attributes. Consequently, we will analyze a dataset of 32 binary variables and 12 960 records.

Table 3. Attributes of the Nursery database

Attribute Values	
parents	usual, pretentious, great pretentious
has nursery	proper, less proper, improper, critical, very critical
form	complete, completed, incomplete, foster
children	1, 2, 3, more
housing	convenient, less convenient, critical
finance	convenient, inconvenient
social	non-problematic, slightly problematic, problematic
health	recommended, priority, not recommended

Table 4. Records distribution for five classes

Class	N	N[%]
Not recommended	4320	(33.333%)
Recommended	2	(0.015%)
Very recommended	328	(2.531%)
Priority	4266	(32.917%)
Special priority	4044	(31.204%)

In order to evaluate the capability of AWIT to project alternative scenarios we proceed as follows (Figures 4a and 4b):

1. Select from the database only the 4,320 records classified as class 1 (Not Recommended), and use this sub sample as the "training set" ;
2. Select from the database the two records classified as class 2 (Recommended), and use this little sub sample as the "testing set";
3. Train an auto-encoder using the training set (Not Recommended records); a Back Propagation algorithm with 16 hidden units and 1 072 weights reached, after 1 856 epochs, a RMSE = 0.01258119;
4. Input each one of the two records of the testing set (Recommended) to the trained ANN;
5. Our expectation is to show that the ANN, trained only on the Not Recommended records, reformulate the input of the two "Recommended" records into typical "Not Recommended" ones.

Tables 5 and 6 synthesize the ANN input/output transformations. Comments:

1. In both cases the ANN transforms the crisp logic of the database (Input) into a **fuzzy output**, a value contained in the closed unit interval [0..1];
2. In both cases the ANN changes the original input vector of individual records, classified as class 2, into an output vector completely compatible with the records belonging to class 1 which is the class of all records of the training set.

Table 5. This table shows how the ANN reformulates the input of the first record ("Recommended") using a fuzzy membership grade in order to fit in the class 1 ("Not Recommended")

Subject 1				
Num#	**Variables Name**	**Input**	**Output**	
1	Usual_Parents_occupations	true	true at	100%
2	Pretensious_Parents_occupations	false	false at	100%
3	Great_Pretensious_Parents_occupations	false	false at	100%
4	Child_Nursery_Proper	true	true at	100%
5	Child_Nursery_Less_Proper	false	false at	100%
6	Child_Nursery_Improper	false	false at	100%
7	Child_Nursery_Critical	false	false at	100%
8	Child_Nursery_Very_Critical	false	false at	99%
9	Form_of_Family_Complete	true	true at	100%
10	Form_of_Family_Completed	false	false at	100%
11	Form_of_Family_Incompleted	false	false at	100%
12	Form_of_Family_Foster	false	false at	98%
13	*Children_1*	*true*	*true at*	*63%*
14	Children_2	false	false at	100%
15	Children_3	false	false at	100%
16	Children_more	false	false at	100%
17	*Housing_Cond_Convenient*	*true*	*false at*	*75%*
18	Housing_Cond_Less_Convenient	false	false at	96%
19	Housing_Cond_Critical	false	false at	97%
20	Family_Finance_Convenient	true	true at	100%
21	Family_Finance_Inconvenient	false	false at	100%
22	Social_Cond_NonProblematic	true	true at	100%
23	Social_Cond_SlightlyProblematic	false	false at	100%
24	Social_Cond_Problematic	false	false at	100%
25	*Health_Cond_Recommended*	*true*	*false at*	*100%*
26	Health_Cond_Priority	false	false at	100%
27	*Health_Cond_NotRecommended*	*false*	*true at*	*100%*
28	*NotRecommended_For_Nursery*	*false*	*true at*	*100%*
29	*Recommended_For_Nursery*	*true*	*false at*	*100%*
30	VeryRecommended_For_Nursery	false	false at	100%
31	Priority_For_Nursery	false	false at	100%
32	SpecialPriority_For_Nursery	false	false at	100%

Table 6. This table shows how the ANN reformulates the input of the second record ("Recommended") with a fuzzy membership grade in order to fit into the class 1 ("Not Recommended")

Subject 2				
Num#	**Variables Name**	**Input**	**Output**	
1	Usual_Parents_occupations	true	true at	100%
2	Pretensious_Parents_occupations	false	false at	100%
3	Great_Pretensious_Parents_occupations	false	false at	100%
4	Child_Nursery_Proper	true	true at	100%
5	Child_Nursery_Less_Proper	false	false at	100%
6	Child_Nursery_Improper	false	false at	100%
7	Child_Nursery_Critical	false	false at	100%
8	Child_Nursery_Very_Critical	false	false at	99%
9	Form_of_Family_Complete	true	true at	100%
10	Form_of_Family_Completed	false	false at	100%
11	Form_of_Family_Incompleted	false	false at	100%
12	Form_of_Family_Foster	false	false at	97%
13	*Children_1*	*true*	*false at*	*51%*
14	Children_2	false	false at	100%
15	Children_3	false	false at	100%
16	Children_more	false	false at	100%
17	*Housing_Cond_Convenient*	*true*	*true at*	*53%*
18	Housing_Cond_Less_Convenient	false	false at	100%
19	*Housing_Cond_Critical*	*false*	*false at*	*80%*
20	Family_Finance_Convenient	true	true at	100%
21	Family_Finance_Inconvenient	false	false at	100%
22	*Social_Cond_NonProblematic*	*false*	*true at*	*58%*
23	*Social_Cond_SlightlyProblematic*	*true*	*true at*	*67%*
24	Social_Cond_Problematic	false	false at	100%
25	*Health_Cond_Recommended*	*true*	*false at*	*100%*
26	Health_Cond_Priority	false	false at	100%
27	*Health_Cond_NotRecommended*	*false*	*true at*	*100%*
28	*NotRecommended_For_Nursery*	*false*	*true at*	*100%*
29	Recommended_For_Nursery	true	false at	100%
30	VeryRecommended_For_Nursery	false	false at	100%
31	Priority_For_Nursery	false	false at	100%
32	SpecialPriority_For_Nursery	false	false at	100%

3. In both cases the ANN shows a very high specificity: the input- output transformations (grey background in the tables) are different according to the specific input features of each record;

In Table 7 and 8 we repeat the same experiment using another ANN (with the same structure as the first one), using as the training set the 4044 records of class 5 ("Special priority"): the target this time is to understand how the same two testing records of the class 2 ("Recommended") has to be modified to fit into the class 5.

These two independent experiments illustrate some further properties of AWIT; AWIT can simulate the changes that any input record must undergo in order to fit into the typical features of the training dataset learned previously by an ANN. AWIT can make this projection with high accuracy and implies an ability to consider the specificity of each new input vector. Consequently, AWIT, after its training phase, generates an open set of possible prototypes.

The Exploration of the AWIT Implicit Function and the "Re-Entry" Algorithm

The purpose of this section is to define an algorithm able to explore the hyper surface that the ANN approximates during the learning phase. The ANN, during the training phase, tries to approximate the implicit function, $f(x)=0$ present in the training data. Consequently, after the training phase the ANN has found the function in the form of parameters (weights) approximating the hyper surface representing the dataset. This means that a new input vector forces the ANN to generate an output as a hyper point that lies on this hyper surface. But we do not know, at any given moment, how close this hyper point is to a local minimum on this hyper surface. To get this knowledge we need to calculate the gradient in each small area of the hype surface or define a new algorithm able to explore this hyper surface automatically from any starting point. We have chosen this second path and have defined a new algorithm named the "Re-Entry" algorithm (Buscema, 1998).

We regard the answer derived from the ANN during the recall process as "an information input re-entry process" in output for a number of steps autonomously decided by the same ANN according to the type of input that the ANN is processing. This theoretical device allows the assertion of two important methodological concepts and the reaching of a practical advantage. The first point of the method is in reacting to a stimulus, the ANN not only has to handle the information internally, but it also has to perceive its own handling of that information. This means the information produced by the new input is permitted to iterate internally until such time that the information is no longer integrated (reformulated) with the previous information that the Network has codified in its own weights during the training phase.

The iterative algorithm of Re-entry is very simple: it works at every cycle *(n)* of the ANN Recall to minimize the energy between the input vector and the output vector (see Figure 2).

At each cycle (n) we compare the actual output with the output of the previous cycle (n-1); if they are different (within a small interval of tolerance) we use the output of cycle (n) as new input for the next cycle (n+1).

Because all of the input vectors during the training and the testing phase are scaled between 0 and 1, the necessary convergence of this technique is based on the fixed point theorem.

In terms of dynamical systems we write:

$$Y(t + 1) = F(X(0), Y(t), W, V)$$

where X(0) is the initial impulse and "W" and "V" are the trained weights matrices.

Table 7. This table shows how the ANN reformulates the input of the first record ("Recommended") with a fuzzy membership grade in order to fit into class 5 ("Special Priority")

Subject 1				
Num#	**Variabless Name**	**Input**	**Output**	
1	Usual_Parents_occupations	true	true at	100%
2	Pretensious_Parents_occupations	false	false at	100%
3	Great_Pretensious_Parents_occupations	false	true at	100%
4	*Child_Nursery_Proper*	*true*	*true at*	*94%*
5	Child_Nursery_Less_Proper	false	false at	100%
6	Child_Nursery_Improper	false	false at	100%
7	Child_Nursery_Critical	false	true at	100%
8	*Child_Nursery_Very_Critical*	*false*	*false at*	*94%*
9	Form_of_Family_Complete	true	true at	99%
10	Form_of_Family_Completed	false	false at	100%
11	Form_of_Family_Incompleted	false	false at	100%
12	Form_of_Family_Foster	false	false at	100%
13	*Children_1*	*true*	*true at*	*98%*
14	Children_2	false	false at	99%
15	Children_3	false	false at	100%
16	*Children_more*	*false*	*false at*	*66%*
17	Housing_Cond_Convenient	true	true at	100%
18	Housing_Cond_Less_Convenient	false	false at	100%
19	Housing_Cond_Critical	false	false at	100%
20	*Family_Finance_Convenient*	*true*	*true at*	*95%*
21	*Family_Finance_Inconvenient*	*false*	*false at*	*96%*
22	*Social_Cond_NonProblematic*	*true*	*true at*	*62%*
23	*Social_Cond_SlightlyProblematic*	*false*	*false at*	*94%*
24	Social_Cond_Problematic	false	false at	99%
25	Health_Cond_Recommended	true	true at	100%
26	Health_Cond_Priority	false	false at	100%
27	Health_Cond_NotRecommended	false	false at	100%
28	NotRecommended_For_Nursery	false	false at	100%
29	*Recommended_For_Nursery*	*true*	*false at*	*100%*
30	VeryRecommended_For_Nursery	false	false at	100%
31	Priority_For_Nursery	false	false at	100%
32	*SpecialPriority_For_Nursery*	*false*	*true at*	*100%*

Table 8. This table shows how the ANN reformulates the input of the second record ("Recommended") with a fuzzy membership grade in order to fit into class 5 ("Special Priority")

Subject 2				
Num#	**Variables Name**	**Input**	**Output**	
1	Usual_Parents_occupations	true	true at	100%
2	Pretensious_Parents_occupations	false	false at	100%
3	*Great_Pretensious_Parents_occupations*	*false*	*true at*	*100%*
4	Child_Nursery_Proper	true	true at	86%
5	Child_Nursery_Less_Proper	false	false at	100%
6	Child_Nursery_Improper	false	false at	100%
7	Child_Nursery_Critical	false	true at	100%
8	*Child_Nursery_Very_Critical*	*false*	*false at*	*97%*
9	Form_of_Family_Complete	true	true at	100%
10	Form_of_Family_Completed	false	false at	100%
11	*Form_of_Family_Incompleted*	*false*	*false at*	*97%*
12	Form_of_Family_Foster	false	false at	100%
13	*Children_1*	*true*	*true at*	*93%*
14	Children_2	false	false at	100%
15	Children_3	false	false at	100%
16	*Children_more*	*false*	*true at*	*70%*
17	Housing_Cond_Convenient	true	true at	99%
18	Housing_Cond_Less_Convenient	false	false at	100%
19	Housing_Cond_Critical	false	false at	100%
20	*Family_Finance_Convenient*	*true*	*true at*	*96%*
21	Family_Finance_Inconvenient	false	false at	96%
22	Social_Cond_NonProblematic	false	false at	100%
23	Social_Cond_SlightlyProblematic	true	true at	100%
24	Social_Cond_Problematic	false	false at	100%
25	Health_Cond_Recommended	true	true at	99%
26	Health_Cond_Priority	false	false at	99%
27	Health_Cond_NotRecommended	false	false at	100%
28	NotRecommended_For_Nursery	false	false at	100%
29	*Recommended_For_Nursery*	*true*	*false at*	*100%*
30	VeryRecommended_For_Nursery	false	false at	100%
31	Priority_For_Nursery	false	false at	100%
32	*SpecialPriority_For_Nursery*	*false*	*true at*	*100%*

Figure 2. Code for auto encoder

```
w = TrainedEncodingWeightsMatrix;
v = TrainedDecodingWeightsMatrix;
// For the meaning of "w" and "v" see Figure 1.
n = 0;
Input(n) = NewInput;
Output(n) = 0;
do
{
    n = n + 1;
    AutoEncoderRecall( Input(n), Output(n), w, v );
    Input(n) = Output(n);
}while( Output(n) ≠ Output(n − 1) )
```

This simple mechanism allows the auto-encoder *to interpret the interpretation* that it is provided to the external stimuli and to stabilize its own answer just when it has "digested" the external input through a reflection on its own activity. The Re-entry configures a *Meta Activity* in the ANN, that is, a *higher control* activity is brought to bear on its own activities. In this sense the name Re-entry does not imply a negative analogy with the concept of Re-entry drawn by Edelman in his Neural Groups Selection theory (Edelman, 1992).

There is also a practical advantage in using Re-entry when one uses this ANN in Recall mode: the ANN *forces its interpretation* of the data as much as possible against an unknown input. This allows it to read the emergent system of many stimuli against which those ANNs which are not provided a Re-entry technique cannot generate a stable output, because they do not know the gradient of the hyper surface on which their output is located.

The Re-Entry algorithm is part of AWIT because by means of this algorithm is possible to navigate into the hyper surface of the dataset previously learned by the auto-encoder. This navigation is permitted because the auto-encoder and the Re-Entry technique work in tandem on the trained dataset as a Dynamic Content Addressable Memory: the new inputs are the user questions and the final outputs are the fuzzy answers of the system.

Analogous algorithms were presented at the end of 1990 (McClelland, 1981, 1995; McClelland, J. L. & Rumelhart, 1988; Grossberg, 1980), but they work by **maximizing** the energy in a hyper cube of Boolean values. In other words, they do not consider the hyper surface on which the training data points lie; these algorithms try to maximize the true value of each variable of the dataset, according to the internal and external constraints (weights matrix and external input). In any case, their final output will tend to be a binary vector whose logic is crisp.

The Re-Entry algorithm, instead, uses the external input as fixed coordinates and the weights matrices of the trained data points as a navigation system to find out the closest local minimum of the hyper surface defined by the auto-encoder. Therefore, when the trained auto-encoder receives a perturbation, it works looking for an energy **minimization**. Consequently, the Re-Entry algorithm and the ANN auto-encoder will provide both fuzzy and crisp outputs, according to the situation.

Further, when the external input is represented with a vector already present in the training data, the classic constraints satisfaction algorithms will try to maximize their outputs all the same. The Re-Entry algorithm, instead, will stop itself from the beginning, because it recognizes spontaneously the external input as a vector already learned (a local minimum of its hyper- surface).

The Re-Entry algorithm, therefore, transforms the assigned dataset in a Content Addressable Memory (CAM). Using the trained weights, Re-Entry algorithm formulates prototypical question to the assigned dataset like:

Define the degree of association of a specific variable "x" (or a combination "x1", "x2", "...") of the dataset and all the other variables.

The Re-Entry algorithm will answer to any of these questions as a dynamical system, and its answer trajectory will make part of the answer. The Re-Entry algorithm may be used in AWIT in two different modes:

1. Impulsive mode: the external input is presented only at the beginning;
2. Forced mode: the external input is presented at any Re-Entry cycle.

These two modalities respond to two different cognitive expectations of the human operator: to check the depth of the memory traces of the external input by itself, or to force the system to associate the other variables with the variables activated by the external input.

We can also distinguish **four types of prototypical questions** for which the Re-Entry algorithm is able to give answers:

1. **Virtual** Question: a combination of variables whose simultaneous activation is structurally possible but such combination is not implemented in the training dataset;
2. **Impossible** Question: a combination of variables whose simultaneous activation is structurally impossible in the training dataset (that is: exclusive options of the same macro –variable);
3. **Incorrect** Question: a new combination of variables whose simultaneous activation is very close to a record of the training dataset;
4. **Profiling** Question: a small combination of variables that the Re-Entry algorithm need complete to design a record prototype.

Re-Entry algorithm allows to AWIT to be presented as a theory of the dynamical possible worlds implicit in a dataset.

Application of the "Re-Entry" Algorithm to a Real Dataset

To show the power of Re-Entry algorithm we have trained an Auto-Encoder with 27 hidden units with the complete Nursery dataset (32 input-Output variables and 12 960 records). We have posed to the Auto-Encoder some prototypical question like "Define the prototype of the Not Recommended Request for Nursery". After 16 cycles of Re-Entry, the Auto Encoder presents the prototype of Table 9.

Table 9. Final Prototype of a request "Not Recommended for Nursery" generated by Re-Entry algorithm in 16 cycles

Variables	INPUT	OUTPUT
Usual_Parents_occupations	0	1
Child_Nursery_Improper	0	1
Form_of_Family_Incompleted	0	1
Children_more	0	1
Housing_Cond_Convenient	0	1
Family_Finance_Convenient	0	1
Social_Cond_SlightlyProblematic	0	1
Health_Cond_NotRecommended	0	1
NotRecommended_For_Nursery**	1	1

Figure 3. The re-entry dynamics for the prototype "Not Recommended for Nursery"

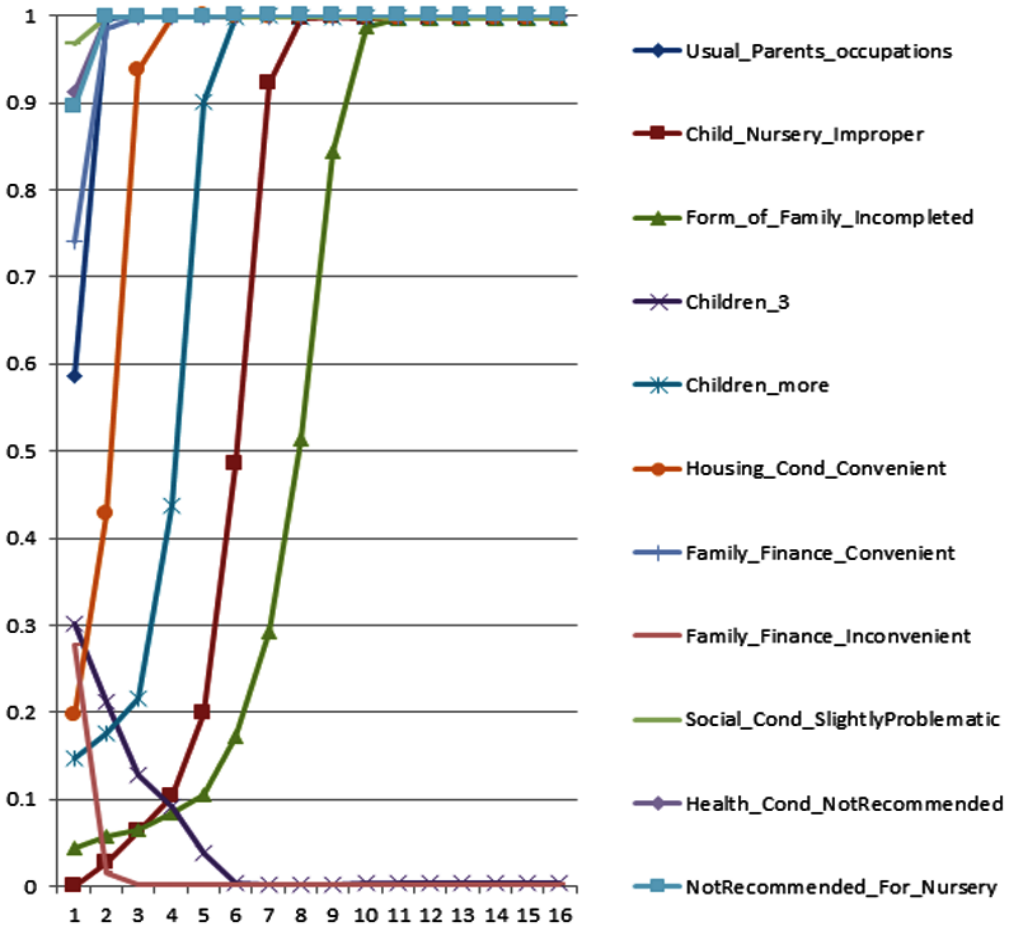


Table 10. Final Prototype of a request "Special Priority for Nursery" generated by Re-Entry algorithm in 14 cycles

Variables	INPUT	OUTPUT
Pretensious_Parents_occupations	0	1
Child_Nursery_Critical	0	1
Form_of_Family_Completed	0	1
Children_2	0	0.99
Housing_Cond_Convenient	0	0.99
Family_Finance_Convenient	0	1
Social_Cond_SlightlyProblematic	0	1
Health_Cond_Priority	0	1
SpecialPriority_For_Nursery**	1	1

In Figure 3 we show the single step by means this prototype was generated dynamically. The trajectory of the answer, of course, is meaningful: the priority of activation and/or inhibition of the different variables enhance the cause-effect relationships among the variables during their negotiation process.

Now to the Auto Encoder we pose the opposite, that is, the prototype of a Request of "Special Priority for Nursery", and Table 10 shows the Auto Encoder answer after 14 cycles of Re-Entry.

The prototype this time is not simply the opposite of the previous one, but it is the reasoned selection of the variables that best fit with the Special Priority condition. Figure 4 shows the dynamic of this answer, where some variables outline a complex and non-monotonic trajectory before to find their stability (final attractor).

Briefly, the Re-Entry algorithm is a technique to transform AWIT in a dynamic scenarios simulator. Re-Entry, in fact, permits to pose "what if" questions to the trained auto-encoder like "how does a system change when it is perturbed in order to fit new input values on its hyper surface?"

These types of questions may have only dynamic and fuzzy answers, because the system needs time to adapt itself to new and emerging situations and its best answers may not necessary be crisp.

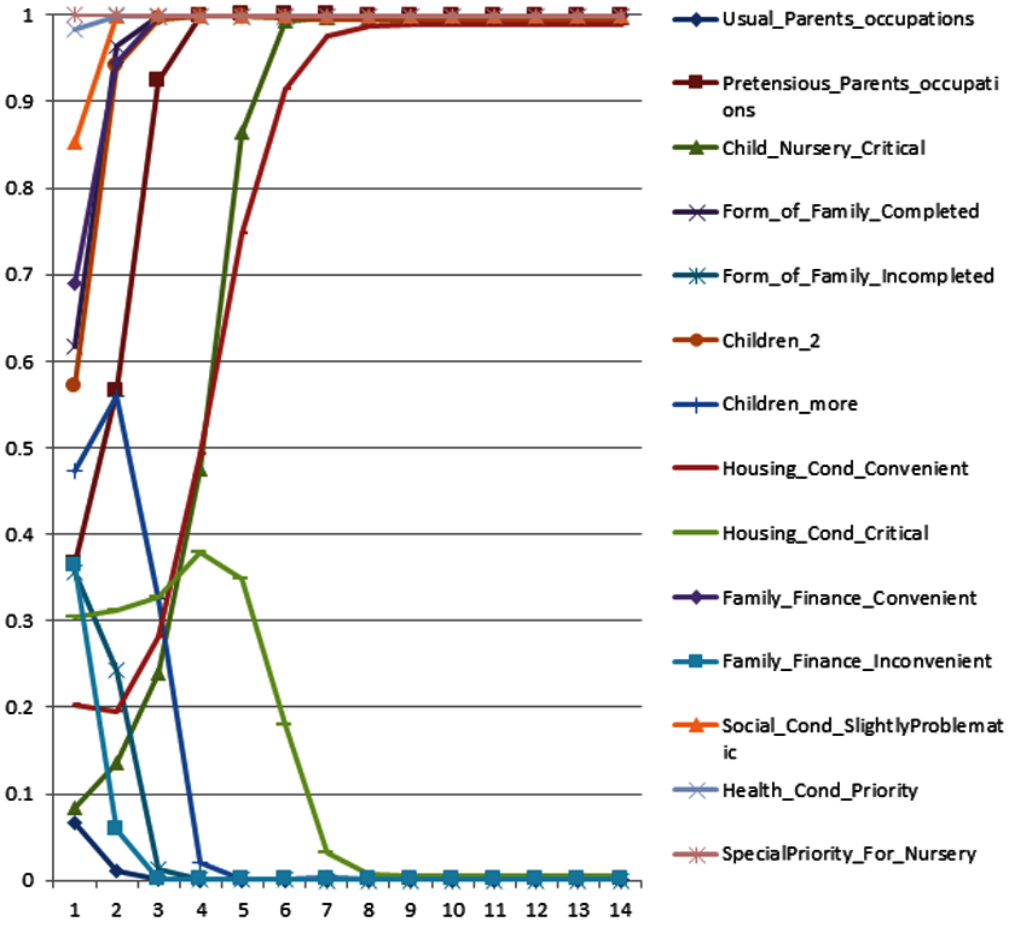
Application of the "Re-Entry" Algorithm to an Artificial Dataset

The Gang dataset (Table 11) is a small dataset already used in scientific literature to enhance the ability of Content Addressable Memory algorithms (McClelland, et. al., 1986, 1988).

The dataset presents five variables, each one composed of a class of options:

Gang = {Jets, Sharks};
Age = {20's, 30's, 40's};
Education = {Junior High, High School, College};
Status = {Married, Single, Divorced};
Profession = {Pusher, Bookie, Burglar}.

Figure 4. The re-entry dynamics for the prototype "Special Priority for Nursery"



After a Simple Pre-Processing the Five Variables are Transformed into 14 Attributes

Table 12 shows an elementary frequency distribution of the 12 attributes into the two main classes (Jets and Sharks). Table 12 also shows how, from the viewpoint of the three professions, the records are equally distributed into the two gangs. Consequently, only a multivariate analysis could establish the eventual degree of specificity of each profession in relation to the two gangs.

We trained an auto-encoder with 14 hidden units up to a RMSE close to 10^{-4}. Then we have used the Re-Entry algorithm to explore the hyper-surface of the auto-encoder weights with different questions:

1. Two prototypical questions: we have **forced** (see section 5) the system to design a profile of an abstract record:
2. To get the typical profile of a Jets member only the Jets variable was activated;
3. To get the typical profile of a Sharks member only the Sharks variable was activated;

Table 11. The Gang dataset

Name	Gang	Age	Educ	Status	Profession
ART	Jets	40	JrHigh	Single	Pusher
AL	Jets	30	JrHigh	Married	Burglar
SAM	Jets	20	College	Single	Bookie
CLYDE	Jets	40	JrHigh	Single	Bookie
MIKE	Jets	30	JrHigh	Single	Bookie
JIM	Jets	20	JrHigh	Divorced	Burglar
GREG	Jets	20	HS	Married	Pusher
JOHN	Jets	20	JrHigh	Married	Burglar
DOUG	Jets	30	HS	Single	Bookie
LANCE	Jets	20	JrHigh	Married	Burglar
GEORGE	Jets	20	JrHigh	Divorced	Burglar
PETE	Jets	20	HS	Single	Bookie
FRED	Jets	20	HS	Single	Pusher
GENE	Jets	20	College	Single	Pusher
RALPH	Jets	30	JrHigh	Single	Pusher
PHIL	Sharks	30	College	Married	Pusher
IKE	Sharks	30	JrHigh	Single	Bookie
NICK	Sharks	30	HS	Single	Pusher
DON	Sharks	30	College	Married	Burglar
NED	Sharks	30	College	Married	Bookie
KARL	Sharks	40	HS	Married	Bookie
KEN	Sharks	20	HS	Single	Burglar
EARL	Sharks	40	HS	Married	Burglar
RICK	Sharks	30	HS	Divorced	Burglar
OL	Sharks	30	College	Married	Pusher
NEAL	Sharks	30	HS	Single	Bookie
DAVE	Sharks	30	HS	Divorced	Pusher

4. A impossible question: we have **forced** the system with two variables mutually incompatible:
5. Because we know that each member may not belong to both gangs, only the variables "Jets" and "Sharks" were activated simultaneously;
6. A virtual question: we have **forced** the system to profile abstract records with attributes ever present in the dataset:
7. No record presents the attributes "College" and "40s" at the same time, but the structure of the dataset allows this combination; so we have activated these two attributes to explore this combination of features, permitted but not instantiated.

Table 12. Frequency distribution of the 12 attributes into the two Gang

Frequency	Jets	Sharks	Jets	Sharks
20s	9	1	60.00%	8.33%
30s	4	9	26.67%	75.00%
40s	2	2	13.33%	16.67%
JH	9	1	60.00%	8.33%
HS	4	7	26.67%	58.33%
COL	2	4	13.33%	33.33%
Single	9	4	60.00%	33.33%
Married	4	6	26.67%	50.00%
Divorced	2	2	13.33%	16.67%
Pusher	5	4	33.33%	33.33%
Bookie	5	4	33.33%	33.33%
Burglar	5	4	33.33%	33.33%
Total	15	12		

8. An incorrect question: we input a wrong record as external input **only at the beginning** (see section 5) of the Re-Entry algorithm to test the ability of the system to correct the input spontaneously:
9. The triplet "Sharks" + "20's" + "Married" is not present in the Gang dataset, and the attribute "20's" is mostly atypical for a Sharks member (only 1 record has this feature). Our expectation is a spontaneous self-correction of the system with a combination of attributes similar to the initial one.

Table 13. Prototypical Attributes of Jets member (in bold)

Attributes	Activation
Jet =1.000**	**0.999987**
Sharks	0.000023
20's	**0.886697**
30's	0.001682
40's	0.009073
JH	**0.957944**
COL	0.000346
HS	0.031924
Single	**0.997109**
Married	0.00014
Divorced	0.005706
Pusher	**0.993779**
Bookie	0.000039
Burglar	0.011095

Figure 5. Membership of each record to the prototype of the Jets member (Jets = Red vs Sharks = Blue)

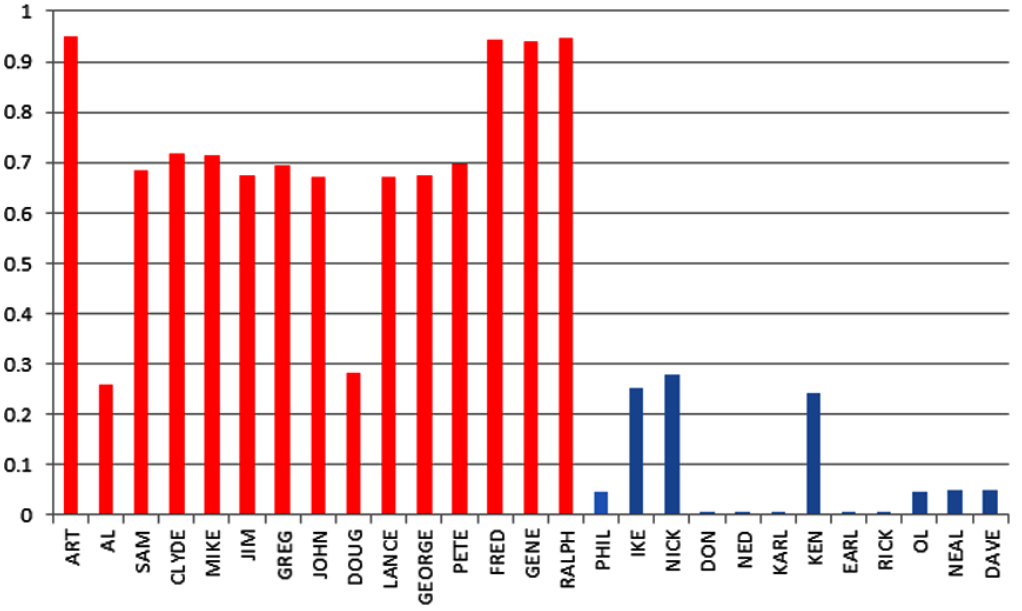
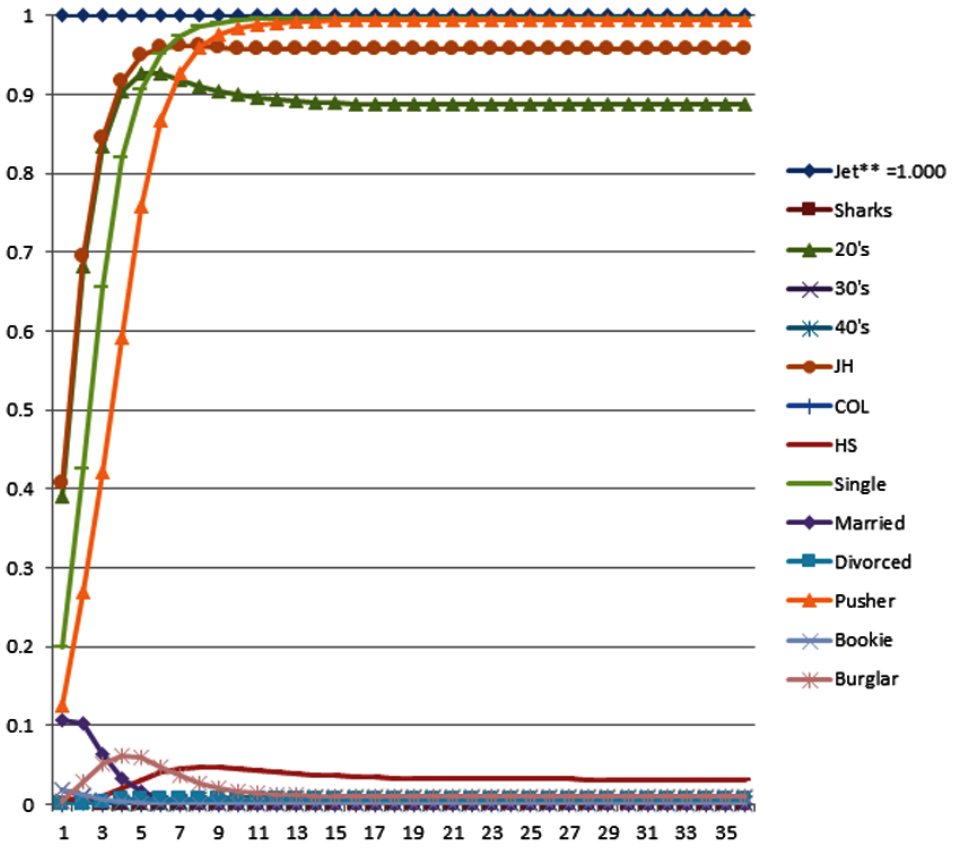


Table 14. Prototypical Attributes of Sharks member (in bold)

Attributes	Activation
Jet	0.000347
Sharks =1.000**	**0.999894**
20's	0.005316
30's	**0.990838**
40's	0.001545
JH	0.00912
COL	0.02965
HS	**0.943527**
Single	0.000025
Married	**0.990061**
Divorced	0.097012
Pusher	0.004884
Bookie	0.001086
Burglar	**0.996202**

Figure 6. Dynamics of the attributes in order to profile the Jets prototype



The Prototype of a Jets Member

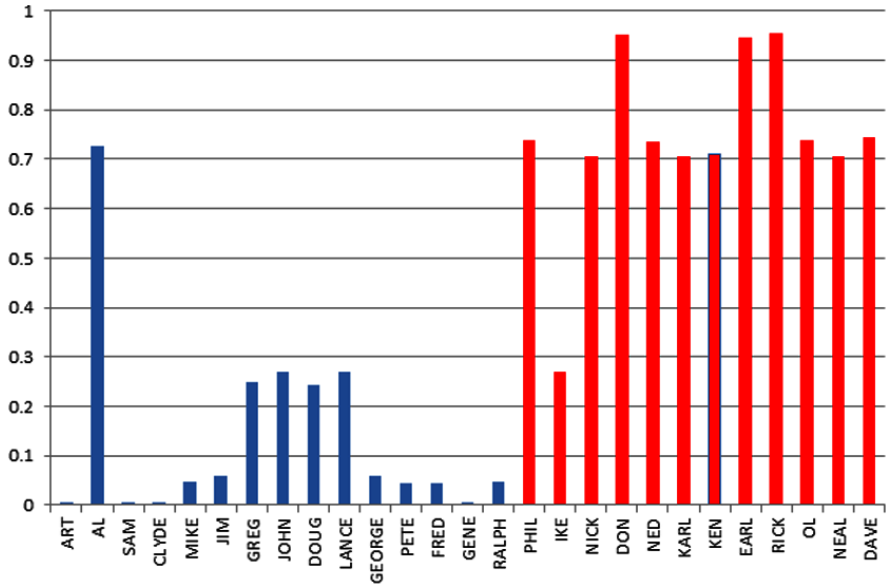
Table 13 shows the prototypical attributes of a Jets member. The more activated attributes in this situation design a profile of the typical Jets member completely abstract from the dataset; in fact the combination of features as "20's" + "Junior School" +"Single" + "Pusher" is not instantiated by any record. All the same the correlated activation of the record is surprising correct (see Figures 5 and 6).

The Prototype of a Sharks Member

Table 14 shows the same for the Sharks gang and also in this case the final result is an abstraction: no record in the Gang dataset presents simultaneously the attributes "Sharks" + "30's" + "High School " +"Married" + "Burglar". But Figure 7 shows how the records activation is coherent with this Sharks prototype (note how "Al" is an outlier in the Jets gang and "Ike" is an outlier in the Sharks gang).

Figure 8 shows how each variable has negotiated its activation value with all the others before the Re-Entry algorithm reaches the attractor. This dynamics are highly nonlinear (refer to the trajectory of the attribute "Divorced") and it moves in a fuzzy set world.

Figure 7. Membership of each record to the prototype of the Sharks member (Sharks=Red VS Jets=Blue)



Impossible Question: The Shared Attributes of "Jets" and "Sharks"

In SQL terms there is not an intersection between the set "Jets" and the set "Sharks" because these two attributes are orthogonal in the architecture of Gang dataset. Consequently, the prototypical attributes of an imaginary member belonging to both gangs seems impossible. But the knowledge of his/her combination of features would be useful to recruit new gang members and/or to understand which ones of the actual members are prone to belong to both gangs.

Table 15 shows the combination of attributes typical of an imaginary record that with different degrees of membership may represent this hybrid condition. The shown solution is interesting from a different point of view:

1. The final degree of membership in the two gangs is not the same: the Re-Entry algorithm finds a larger matching with the Sharks gang;
2. Consequently, three of the more activated attributes represent more (but not exclusively) of the Sharks gang than the Jets ("30's", "Married", and "Burglar");
3. One of the more activated attributes ("Junior School") quite exclusively represents the Jets gang;

Figure 9 shows the different membership in the meta-gang "Jets & Sharks" of each record in the dataset: the Jets membership is more selective while the Sharks membership is more spread out (Figure 10).

Figure 8. Dynamics of the attributes in order to profile the Sharks prototype

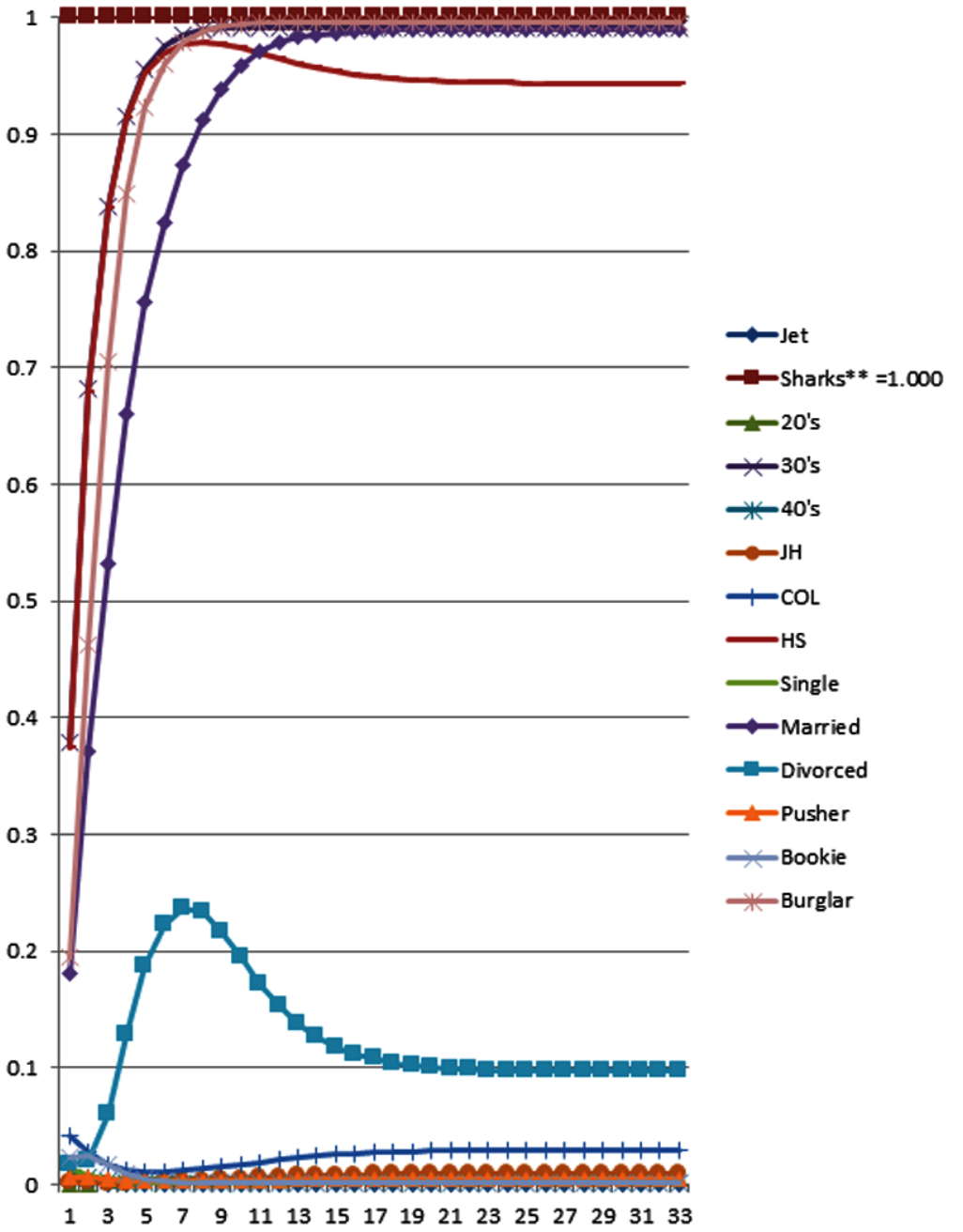


Table 15. Prototypical Attributes of a Jets and Sharks member (in bold)

Attributes	Activation
Jet =1.000**	**0.665672**
Sharks =1.000**	**0.85475**
20's	0.00023
30's	**0.998589**
40's	0.002374
JH	**0.989651**
COL	0.006513
HS	0.00066
Single	0.000082
Married	**0.99793**
Divorced	0.016178
Pusher	0.000265
Bookie	0.001929
Burglar	**0.999204**

Figure 9. Membership of each record to the prototype of the Jets (Blue) and Sharks (Red) meta-gang

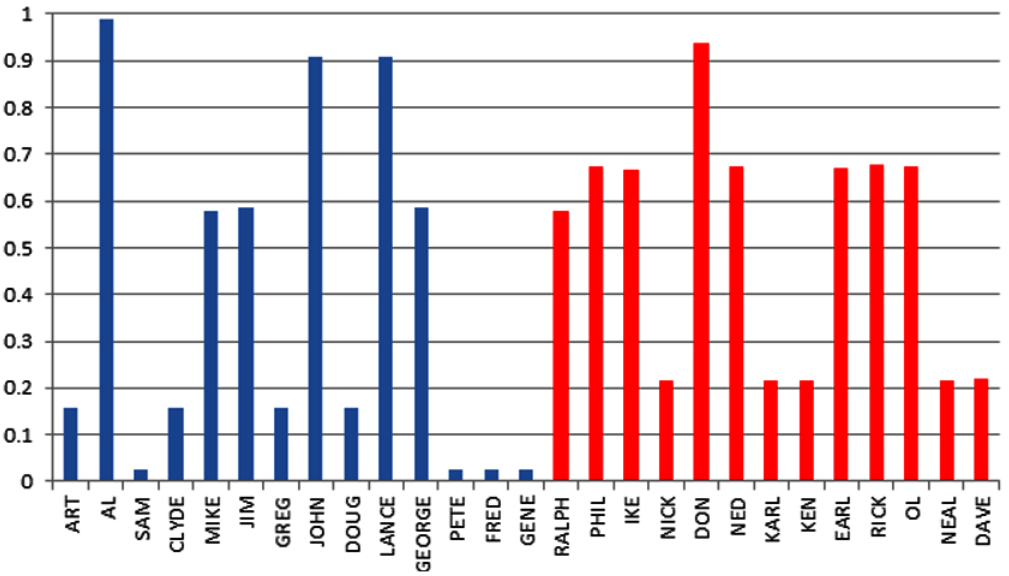
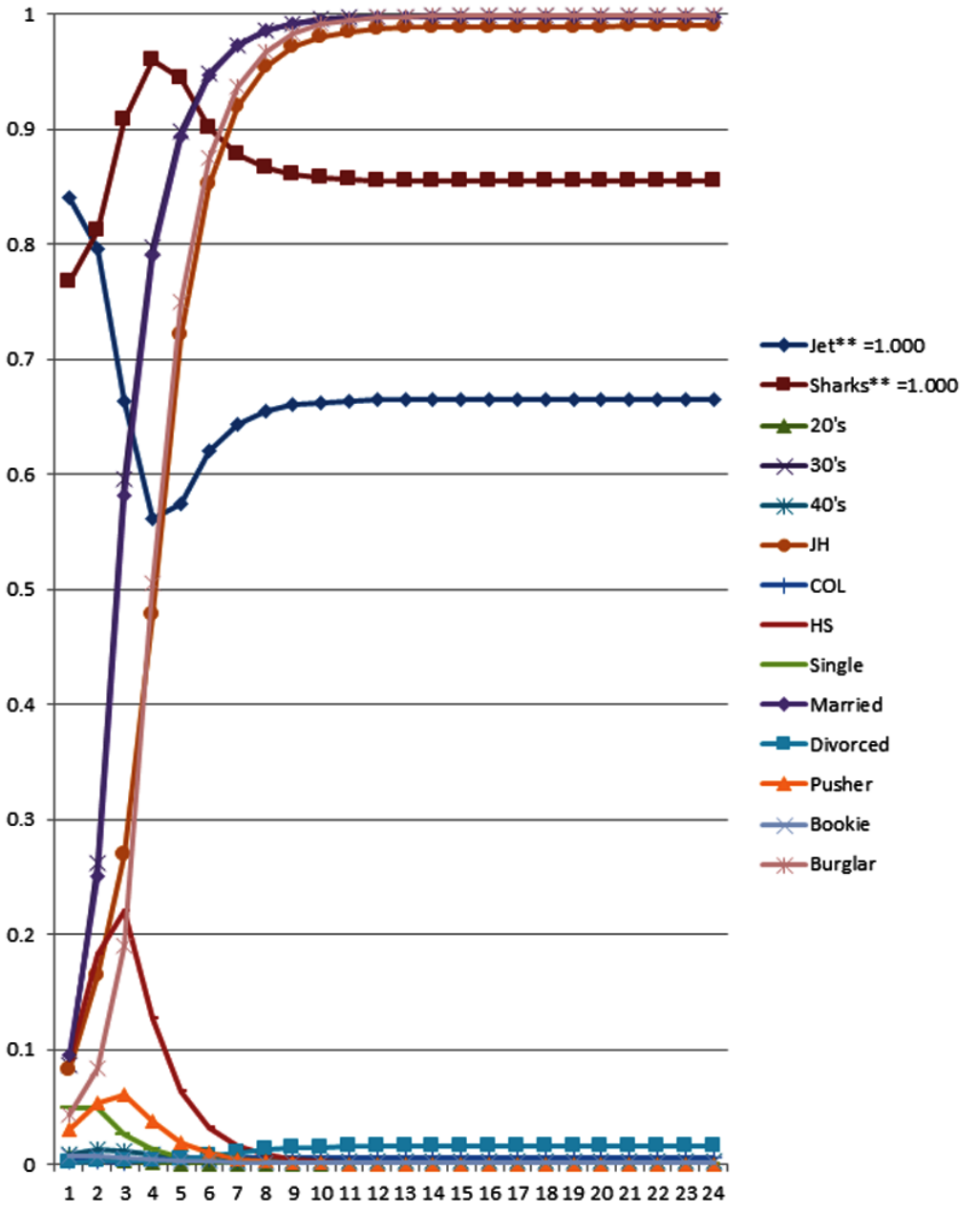


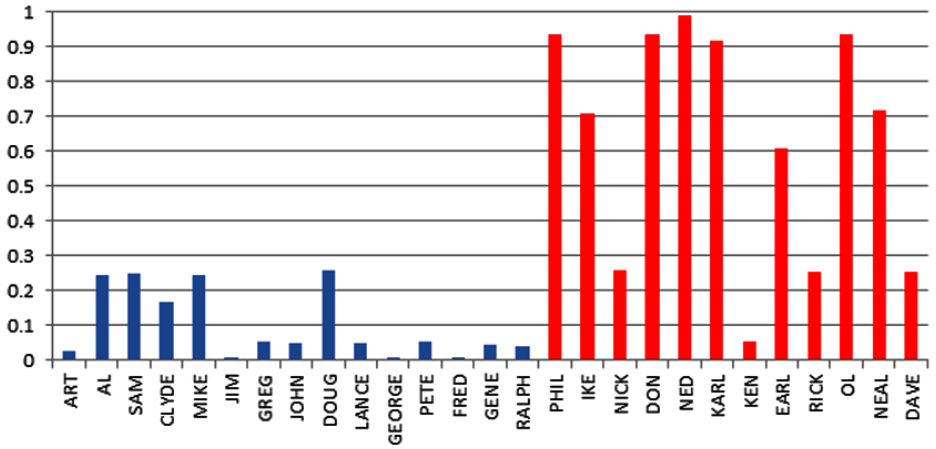
Figure 10. Dynamics of the attributes in order to profile the Jets-Sharks prototype



Virtual Question: A Gang Member "40's" and with "College" Education

There is no gang member that is 40 years old and with a College education in the dataset, but this combination of attributes does not belong to the same macro-variable, and then theoretically it is a non-used possibility.

Figure 11. Membership of each record to the prototype of the new virtual record (Jets=Blue vs Sharks=Red)



The Re-Entry algorithm forces the weights matrix to consider this possibility and Table 16 shows the result.

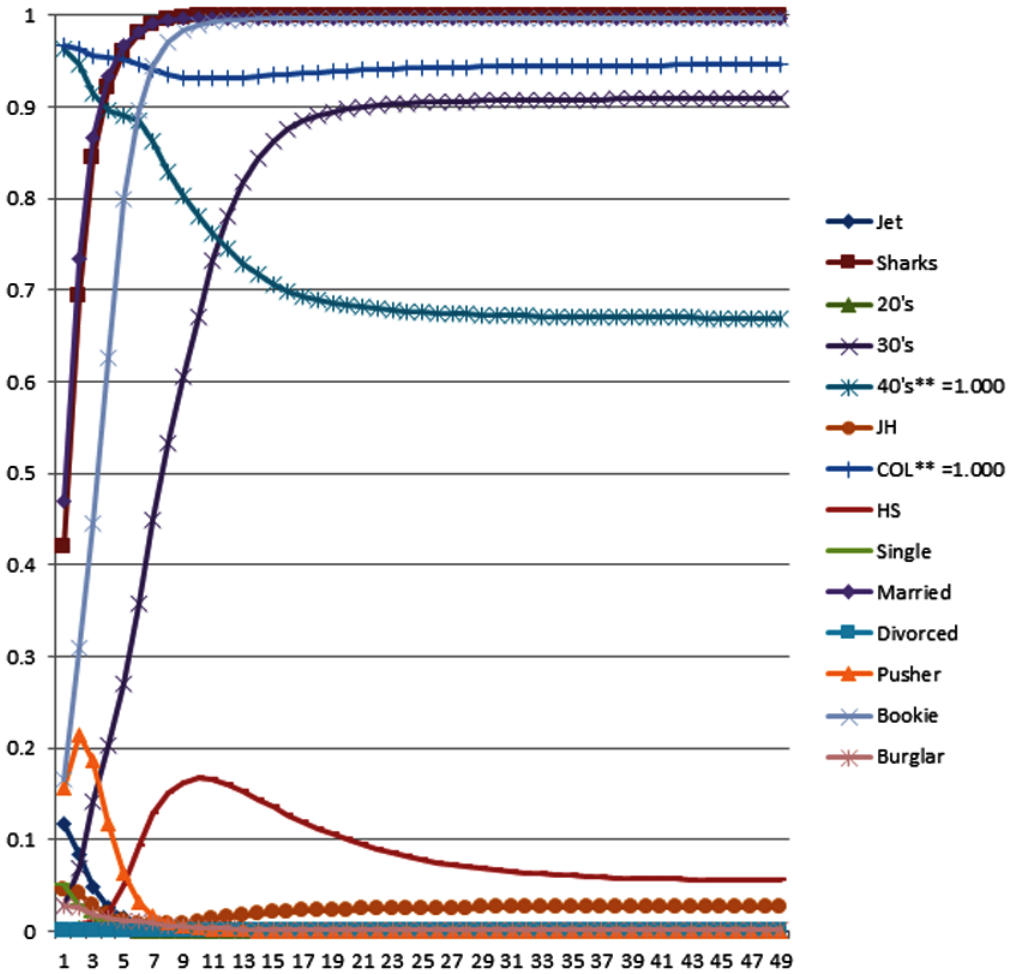
This time the algorithm decreases the activation strength of the attribute "40's" and increases the attribute "30's", showing a non-monotonic behavior. Further, new attributes are activated as "Sharks" and "Married", but also "Bookie" that is not strongly linked to the first two. A new virtual record is created, not present in the original dataset. This new virtual combination is close to some of the dataset records such as "Ned", but "Ned" has no link to "40's" with a fuzzy membership of 0.67.

Figure 11 shows the closer candidates among the actual records for these new attribute combinations, and Figure 12 shows the dynamics through which the new attributes are optimized.

Table 16. Prototypical Attributes of a gang member with "40's" and a "College" education (in bold)

Attributes	Activation
Jet	0.000816
Sharks	**0.999402**
20's	0.000026
30's	**0.908925**
40's ** =1.000**	**0.669465**
JH	0.026373
COL ** =1.000**	**0.945939**
HS	0.055099
Single	0.007278
Married	**0.997502**
Divorced	0.000009
Pusher	0.000786
Bookie	**0.996545**
Burglar	0.002692

Figure 12. Dynamics of the attributes in order to profile the new virtual record



Incorrect Question: A "Sharks" Member with Attributes "20's" and "Married"

"Married" seem to be a typical attribute of Sharks gang, but the attribute "20's" seems to be quite exclusive of Jets (only one Sharks member is "20's").

When the Re-Entry algorithm receives these inputs only at the beginning (Impulsive Mode), it rearrange its trajectory according to an energy minimization criterion. Consequently, when the algorithm is activated the impulsive mode attempts to spontaneously correct mistakes and / or outliers.

Table 17 shows the corrected prototype proposed by the algorithm: the attributes "20's", activated at the beginning is been deleted, while a new attribute ("40's"), which is more consistent with the new prototype, is selected.

Table 17. Corrected Attributes of prototype (in bold)

Attributes	Activation
Jet	0.00057
Sharks =1.000**	**0.999694**
20's =1.000**	**0.004224**
30's	0.001752
40's	**0.992658**
JH	0.001336
COL	0.000913
HS	**0.998323**
Single	0.000054
Married =1.000**	**0.99931**
Divorced	0.002648
Pusher	0.001743
Bookie	0.003626
Burglar	**0.994601**

Figure 13 shows the degree of membership of the actual records to the new prototype. The Sharks members, obviously, are the most activated, but also some of the Jets membership present a similar fuzzy membership.

Figure 14 shows the complex dynamics through which the attributes are changed in order to define a "correct" prototype. It is interesting to note in the figure how the two attributes, "20's" and "40's" cross each other during the process, the first one decreasing and the second one increasing its activation.

Figure 13. Membership of each record to the corrected prototype (Jets=Blue vs Sharks=Red)

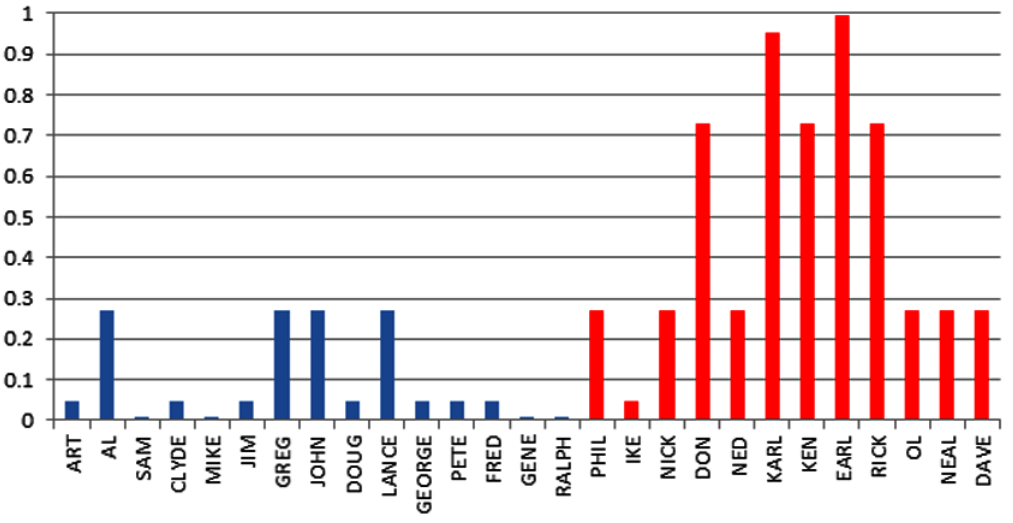
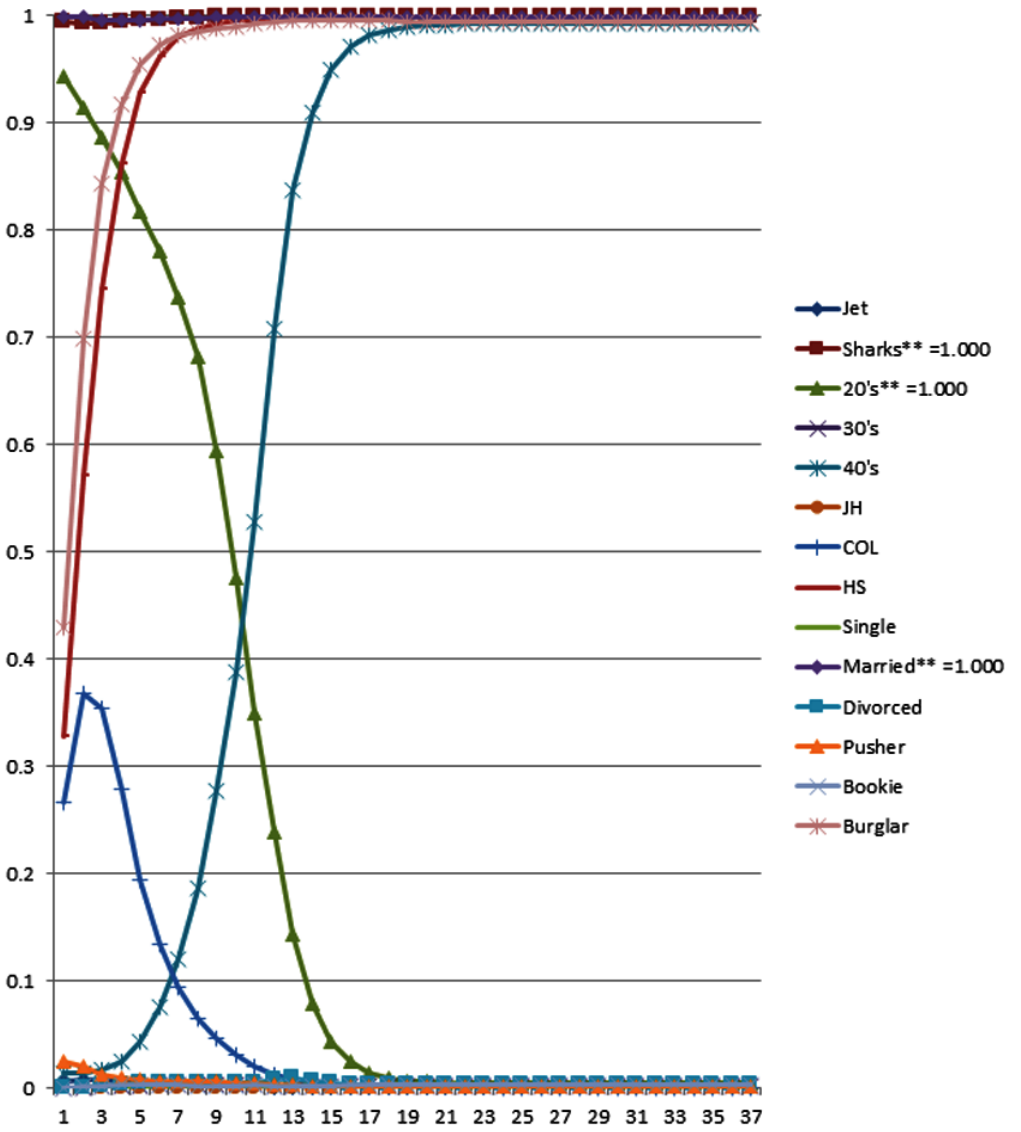


Figure 14. Dynamics of the attributes in order to define a "correct" prototype



DISCUSSION

AWIT is a new neural system composed of an auto-encoder ANN, suitable for approximating the implicit function of the assigned dataset, and a new algorithm, named Re-Entry, through which one can surf inside the hyper-surface of the data, posing different kind of possible question:

1. Profiling questions;
2. Impossible questions;
3. Virtual questions;
4. Incorrect questions.

AWIT, therefore, is a suitable technique for a real "what if" analysis, that is, for dynamic scenarios simulation.

The auto-encoder is able to rewrite any new record utilizing the "logic" of the dataset imputed during its training phase. This capability represents a key point because it allows us to give an approximate answer to question as: "how a generic pattern will be reformulated when will work in a known context but different from the original?"

Let us imagine the cost of healthcare in two different geopolitical regions and let us further suppose that the *cost* per person is similar, but the *quality* of the results is completely different: very high in the first region and low to medium in the second one. AWIT can learn the detailed costs of the health units contained in the virtuous region and then AWIT can reformulate the details costs of each health unit of the region of low quality in the test phase. In this way we can know how the costs of each health unit of low quality have to be reformulated to improve its own quality.

The Re-Entry algorithm, from another perspective, is able to use all the information that the auto-encoder has previously learned. This represents another key point: in fact the Re-Entry algorithm provides an answer to questions such as: "how will a generic system react when some of its parameters are changed and/or forced with specific values?"

Taking again the example from the healthcare system, the problem could be how to calculate the complex side effects generated when we change some costs in a specific health unit. Considering the time in which we live, AWIT could be simultaneously both useful and dangerous.

REFERENCES

Bengio, Y. (2009). Learning Deep Architectures for AI. *Machine Learning*, *2*(1), 1–127. doi:10.1561/2200000006

Bengio, Y. (2009). Classification using Discriminative Restricted Boltzmann Machines. In *Proceedings of the 25th International Conference on Machine Learning*. Helsinki, Finland.

Buscema, M. (1998). Recirculation Neural Networks. *Substance Use & Misuse*, *33*(2), 383–388. doi:10.3109/10826089809115872 PMID:9516734

Edelman, G. M. (1992). *Bright Air, Brilliant Fire: On the Matter of the Mind*. USA: Basic Books.

Grossberg, S. (1980). How does the brain build a cognitive code? *Psychological Review*, *87*(1), 1–51. doi:10.1037/0033-295X.87.1.1 PMID:7375607

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets [MIT Press Journal.]. *Neural Computation*, *18*(7), 1527–1554. doi:10.1162/neco.2006.18.7.1527 PMID:16764513

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, *313*(5786), 504–507. doi:10.1126/science.1127647 PMID:16873662

Le, Q. V., Ranzato, M. C., Monga, R., Devin, M., Chen, K., Corrado, G. S., & Ng, A. Y. et al. (2012). Building High-level Features Using Large Scale Unsupervised Learning. In *Proceedings of the 29-th International Conference on Machine Learning*. Edinburgh, Scotland, UK.

McClelland, J. L. (1981). Retrieving general and specific information from stored knowledge of specifics. *Proceedings of the third annual meeting of the cognitive science society*, 170-172, Berkeley, CA.

McClelland, J. L. (1995). Constructive memory and memory distortions: a parallel-distributed processing approach. In Schacter D. L. (ed), Memory distortion: How minds, brains, and societies reconstruct the past, 69-90. Harvard University Press, Cambridge, MA.

McClelland, J. L., & Rumelhart, D. E. (1988). *Explorations in parallel distributed processing. A handbook of models, programs, and exercises*. Cambridge: MIT Press.

McClelland, J. L., & Rumelhart, D. E. (1988). Interactive Activation and Competition, Chapter 2, Figure 1. In Explorations in parallel distributed processing. A handbook of models, programs, and exercises, MIT Press, Cambridge.

McClelland, J. L., Rumelhart, D. E., & Hinton, G. E. (1986). The Appeal of Parallel Distributed Processing, Chapter 1, Figure 10. In D. E. Rumelhart, J. L. McClelland and the PDP Research Group, Parallel Distributed Processing. Exploration in Microstructure of Cognition, 1, MIT Press.

Raiko, T., Valpola, H., & LeCun, Y. (2012). Deep Learning Made Easier by Linear Transformations in Perceptrons. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS). XX*, Journal of Machine Learning Research: Workshop and Conference Proceedings. La Palma, Canary Islands.

Raina, R., Madhavan, A., & Ng, A. Y. (2009). Large-scale Deep Unsupervised Learning using Graphics Processors. In *Proceedings of the 26th International Conference on Machine Learning*. Montreal, Canada. doi:10.1145/1553374.1553486

ENDNOTE

[1] M. Bohanec, V. Rajkovic: Expert system for decision making. Sistemica 1(1), pp. 145-157, 1990.

Paolo Massimo Buscema is a full Professor and Computer Scientist, as well as an expert in Neural Networks and Artificial Adaptive Systems. He is the Executive Director of the Semeion Research Centre for the Sciences of Communication based in Rome, Italy. The institution is recognized by the Italian Ministry of University and Scientific Research (MIUR). Additionally, Professor Buscema is a faculty member in the Department of Mathematics and Statistics of the University of Colorado (USA), and founder and Member of the Center for Computational and Mathematical Biology (CCMB) at the University of Colorado. He is also Adviser to the Presidency of the Council of Ministers of the Italian Government. From 2003 to 2007 Professor Buscema was an adviser to New Scotland Yard, London, UK. He has designed, constructed and developed new models and algorithms of Artificial Intelligence, is the author of scientific publications on theoretical aspects of Natural Computation, with over 250 titles (scientific articles, essays, and 25 books) and over 35 Software Systems used in many Universities and international research centres. He also holds 22 international patents.

William J. Tastle is Professor of Information Systems in the School of Business at Ithaca College in New York. He is a Senior Member of both IEEE and ACM. Dr Tastle is a Research Fellow of Semeion Research Institute, Rome, Italy, a Research Professor at the University of Iceland, Visiting Scientist at Los Alamos, International Scholar in Milan, and served on many boards. His research interests are in artificial adaptive neural networks, measure theory, analytics of complex systems, and big data analytics.