Artigo Técnico

Revista de Informática Teórica e Aplicada

Non-Supervised Sensory-Motor Agents Learning

Raul Sidnei Wazlawick Dr. Eng.

Universidade Federal de Santa Catarina
Departamento de Informatica e Estatística
Laboratório de Software Educacional EDUGRAF
raul@edugraf.ufsc.br

UFSC-CTC-INE
Cx. Postal 476 Campus Trindade
88040-900 Florianópolis, SC Brazil

Antônio Carlos da Rocha Costa Dr. Sc.

Universidade Federal do Rio Grande do Sul
Instituto de Informática
Departamento de Informática Teórica
rocha@inf.ufrgs.br

Instituto de Informática-UFRGS
Caixa Postal 15064
91501-970 Porto Alegre, RS Brazil

Abstract

This text initially discusses the distribution of neurons in a neural network with non-supervised learning. A proposal for creation and destruction of neurons based on the activity related to the concepts being recognized is introduced. This model, called *activity schema,* is used to define a sensory-motor agent as a collection of activity schemata, each one representing a sensory-motor concept. The sensorial characteristic of a sensory-motor concept is represented by a neuron that acts as a recognizer of observable things (Piaget calls these things *observables*). The motor characteristic is given by an action vector defined by the schema on the agent's environment and by a goal vector that the schema tries to reach. The activity schema permits a useful distribution of neurons in a conceptual space, creating concepts based on action and sensation, and not only on sensation. Such approach is inspired

in the theory of the Swiss psychologist and epistemologist Jean Piaget, and intends to make explicit the account of the processes of continuous interaction between sensory-motor agents and their environments when agents are producing cognitive structures.

A four-page version of this paper will be published by the International Conference on Neural Networks and Genetic Algorithms ICANNGA'95, to be held in Alès, France.

1 Introduction

The notion of an *autonomous agent* plays a central role in contemporaneous research on Artificial Intelligence [DEM 93]. An autonomous agent is a system with decision capacity about the goals that will orient its action in a given environment.

There are two main autonomous agents classes: the so called *cognitive (or symbolic) agents*, and the *reactive (or non-symbolic) agents*. Cognitive agents are based on symbolic processing mechanisms taken from traditional artificial intelligence systems. Reactive agents are based on alternative computational mechanisms like neural networks, analogic processing, etc.

The alternative approach using autonomous agents based on hybrid mechanisms has not been extensively explored [DEM 93], and the question of how to proceed for combining symbolic and non-symbolic mechanisms is an up-to-date research topic.

Our work [WAZ 93] [COS 93] is oriented towards Jean Piaget's Genetic Psychology and Epistemology [PIA 76]. According to that perspective, each of the two kinds of mechanisms (symbolic and non-symbolic) of an autonomous agent are coordinated in a very particular way, which is determined by the so-called *equilibration mechanism*. The characteristic of this kind of coordination is that the symbolic mechanism results from an elaborated construction, called *reflexive abstraction* [PIA 76], based on elements given by non-symbolic mechanism.

This paper presents the structures and processes that may form a basic level for non-symbolic processing for hybrid autonomous agents. These processes and structures are believed to provide adequate mechanisms for reflexive abstraction.

In order to maintain the terminology of genetic epistemology, the non-symbolic level will be called *sensory-motor level*. Correspondingly, the autonomous agents belonging only to this level will be called *sensory-motor autonomous agents*.

The sensory-motor mechanism description is the body of this paper, and it was originally defined in [WAZ 93]. It's based on Kohonen's non-supervised neural networks [DAY 90] and on genetic algorithms [HOL 75], that were combined for reproducing the main structure of the sensory-motor level, called *sensory-motor schema* [PIA 76]. The sensory-motor schema are also called *activity-schema*.

The schema mechanism differentiates from today's culture on neural networks. The nets are basically reactive mechanisms, whose evolution is stimulated by provided entries. The schema mechanism uses the nets as sensors, and keeps them able to evolve reactively (in the case of weights adjustments), but also introduces the possibility of active evolution through genetic algorithms (in the case of schemata reproduction). Essential to the schema mechanism is that both kinds of schema

evolution processes take the agent's interaction process with its environment into account.

Sections 2 and 3 discuss neural networks with non-supervised learning, and their problems from the construtivist perspective. It's assumed that the reader has the basic knowledge on this field. Section 4 characterizes in detail the activity schemata. Section 5 defines autonomous sensory-motor agents and shows an adequate way of seeing its interaction processes with their environments. The last section points the main aspects of this work, and indicates future work.

2 Distribution of Neurons in Non-Supervised Neural Networks

Every weight in a non-supervised competitive neural network [DAY 90] need to be initialized before the training process. It's common practice to initialize these weights with uniformly distributed random values. However, this procedure may cause serious difficulties during the training process.

Normally, the observable patterns (entries) to the network aren't uniformly distributed, but concentrated in specific regions of the space. Thus, some neurons may be so distant from any pattern that they will never be considered winners of the competition. These neurons may be discarded.

On the other side, on regions with great concentration of patterns, the number of neurons may not be enough for differentiate the several categories of patterns. Thus, the neurons distribution may not be able to reflect the actual structure of the observable space.

One possible solution is to distribute the neurons accordingly to the density of the entry groups. Since usually it's not possible to know this distribution before the net's training, methods were developed to approximate this effect during the training of the network.

3 Existing Solutions

One solution to the problem of initial distribution of neurons is the so called *convex combination method* [DAY 90]. In this method, every weight is defined with the same value $1/\sqrt[2]{n}$, where n is the number of elements on the entry vectors. This makes the weights vectors to be normalized and to be coincident. Moreover, to each element *xi* from the entry is assigned the value $\beta x_i + ((1/\sqrt[2]{n}) * (1-\beta))$. Initially, b is a very small value, such that all entries are near to $1/\sqrt[2]{n}$. As the net is trained, b is incremented until the limit of 1.0 is reached. This makes all entries to diverge, and finally reach their actual values, carrying with them the respective neurons.

The convex combination method works well, but it makes the learning slower. Furthermore, the method works only in the beginning of the training process, and don't predict any possible changes in the distribution of entries. That is, when the value of b reaches 1.0, the method stops working. If the distribution of entry vectors changes after this point, the method becomes innocuous.

Another method consists in adding some noise (random values) to the entries. This makes the entries to move in such a way to eventually capture a weight vector. This method also works well, but it's still slower than the convex combination method.

A third method initiates with random weights, but in the initial stages it adjusts every weight in the network, and not only the winner. This moves all the weight vectors to the regions of greater concentration of entry patterns. As the training proceeds, only the vectors near to the winner are adjusted. This adjusting vicinity becomes more restrict until only the winner is adjusted.

A fourth method gives to each neuron a value k, that corresponds to the number of times it was activated. If the number of iterations of the network is denoted by t, and the number of competitive neurons is n, then one can wait that each neuron is activated k/t times in average. This is given by the following equation: $k/t = 1/n$.

If the value k/t for a specific neuron is greater than $1/n$, then this neuron was activated more times than the average. A neuron at this situation can be put under a temporary threshold that reduces its winning probability. Thus, other neurons may be trained.

All methods described here are effective only at the beginning of the net's training. They become innocuous as the training proceeds. If the environment conditions are modified, with new concepts arising and old ones being recombined, then the net loses its adaptation capacity. From the point of view of the sensory-motor agents, it means that the agent become insensible to variations on the environment organization.

The solution to this problem consists in varying the number of neurons in the net by dynamic creation and destruction of neurons (recognized concepts) accordingly to the variation of the effective concepts.

This model can be implemented by destroying the least activated neurons and creating new ones on regions covered by the most activated neurons. The tendency is to reduce the variance on the number of entries associated to each neuron. Nevertheless, this approach is based only in quantitative information about the number of occurrences of the entries.

However, this is not the kind of strategy observed in categorizations done by humans. The number of exemplars associated to a concept is not the only significant factor. There are concepts with very few observed exemplars, like, for example, Sun, Moon, whale, panda and diamond. There are also concepts with a number much greater of observed exemplars, like insect, star, etc. There are also concepts without any exemplar (for instance, unicorn), but these ones don't have foundation on the sensory-motor level; they are encountered only at symbolic level.

The Activity Schema combines concepts recognition from neural networks with the search mechanism from genetic algorithms. This combined mechanism is used to build neural networks capable of self-adaptation in changing environments. It enables not only the movement of vectors in the entry space but also the variation on the quantity of recognized concepts.

The main difference between the schema mechanism and the other methods is that the former is not restricted to use only quantitative information about vectors distribution for deciding where to allocate neurons. The activity schema is based on the piagetian notion of sensory-motor schema, that associates action and perception [PIA 71], and uses a criterion of action success to orient the creation and destruction of neurons.

4 Activity Schemata

4.1 Sensorial Signals and Entries

A *signal* is defined as a rational value between 0 and 1. Establishing a precision degree p as the number of decimal digits that can be represented by these values, one can define an *alphabet* (finite set of symbols) Sp corresponding to the signals that can be represented with p digits after the decimal point. For example: $S2$ = {0.00, 0.01, 0.02, ..., 0.99, 1.00}.

A *generalized signal* is defined as a rational value between -1 and 1. An *alphabet of generalized signals* with precision p is denoted by Gp.

The *degree* of a signal xSp or xGp corresponds to the value of p.

The operation *randomSignal* : *SpSpSpGp* creates a generalized signal with a random distribution defined by three indexes: $i1$, $i2$ and $i3$. Let *random* : [-1,1] be a random rational function uniformly distributed on the interval [-1,1], then *randomSignal* = $li1.li2.li3.$(*if ABS*(*random*) > $i1$ *then* ($i2$ * *random*) *else* ($i3$ * *random*)).

A sensorial entry is defined as a signal string from Sp. Let e be a sensorial entry, then $e \in S_p^*$, where S_p^* denotes the Kleene closure of Sp, that is, the set of every entries that can be formed by concatenating a finite number of the elements of the alphabet Sp. The *degree of a sensorial entry* e is the value of p. The *size of a sensorial entry* corresponds to the number of signals in the entry. A sensorial entry e of size n will be denoted here by en. The i-th signal of en will be denoted by e_i^n.

4.2 Genes and Chromosomes

A *genetic alphabet* is defined as a finite set formed by the signals of an alphabet Sp plus a special symbol, represented by #, that denotes an undetermined element. Let A be a genetic alphabet, then $A = Sp\{\#\}$. An element aA is called *gene*. The *degree of a genetic alphabet A* is given by the degree of the signals that compounds the alphabet, that is, the value of p. Given a genetic alphabet A, the signals Sp taken from this alphabet are called *proper genes* and # is called *improper gene*.

The operation *difGene* : $S_p \times A \rightarrow$ ***Nat***, when applied to a proper gene, produces the absolute difference between the value of a signal and a gene, and it produces the constant 1/3 if applied to a improper gene, that is, the average of the difference between two random numbers in the interval [0,1] [RAM 94]. Therefore *difGene* = *ls.lg.*(*if g* = # *then* 1/3 *else ABS*(*s*-*g*)), where *ABS* : ***Int Nat*** is the function *ABS* = *lx.*(*if x* 0 *then x else -x*).

The operation *sumGene* : *AGpA* makes the summation of a gene and a generalized signal. This operation is used to implement the mutation of a gene. If the gene is improper, then the summation consists of the gene itself. That is, *sumGene* = *lg.ls.*(*if g*=# *then* # *else g*+*s*).

The operation that adjusts a gene gA to a signal sSg, limited by a learning value a is given by: *adjustGene*: *ASpSpA*, where *adjustGene* = *lg.ls.la.*(*if g*=# *then* # *else g* + (*g*-*s*)*a/2). The approximation

of g to s is directly proportional to the value of a.. In the limit, when a= 1.0, the new g is defined to the average between the old g and s.

A *chromosome* is a string <$a1$, $a2$, ..., an> whose elements are taken from a genetic alphabet A, where n is the size of the chromosome. Let c be a chromosome of size n, then cAn. The *degree of a chromosome* cA^* is given by the degree of the alphabet A. The *size of a chromosome* is defined by the number of genes that this chromosome bears. A chromosome c of size n will be denoted as $cnAn$.

The *similarity* between a sensorial entry $s^n \in S_n^p$ and a chromosome $c^n \in A^n$ produces a value in the interval [0,1]. This value corresponds to the average of the application of the function *difGene* to the genes of cn and to the signals at the corresponding positions at sn. The function *similarity* : $S_n^p \times A^n \rightarrow$ ***Nat*** is given by: $similarity = ls^n.lc^n.\frac{\sum_{i=1}^{n} difGene(s_i^n, c_i^n)}{n}$ for every natural n.

The operation *crossover* consists in taking a pair of chromosomes $(xn,yn)AnAn$, and combining their genes, obtaining a new pair. The operation *crossover* : $AnAn$***Nat***$AnAn$ takes two chromosomes and a natural value r, such that 1 £ r £ n, what indicates that the crossover point will occur immediately after the r-th gene of the chromosome. If $r = n$ then the crossover operation simply returns the original pair of chromosomes. That operation is defined by: $crossover = lxn.\ lyn.\ lr.$ ($if\ r = n\ then$ (xn , yn) $else$ $< x_1^n, .., x_r^n, y_{r+1}^n, .., y_n^n >, < y_1^n, .., y_r^n, x_{r+1}^n, .., x_n^n >$).

A *vector of mutation* is defined as a string of generalized signals from Gp. Vectors of mutation are created with random signals. A vector of mutation $v \in G_p^*$ is created from the distribution of the function randomSignal and a value t that defines the quantity of signals that the vector will bear. The operation *vmut* : ***SpSpSpNat*** creates a vector of mutation respecting the following restrictions: ($i1*t$) values are generated by a random function on the interval [-$i2$,+$i2$] and ((1-$i1$)*t) values are generated on the interval [-$i3$,+$i3$]. Therefore, $vmut = li1.li2.li3.lt.<randomSignal(i1,i2,i3)$**1**, ..., $randomSignal(i1,i2,i3)$***t*** >.

An acceptable mutation vector must has its values concentrated on the vicinity of 0.0. Thus, a sample acceptable vector would has: $i1$=0.1, $i2$= 1.0, $i3$=0.2, meaning that 0.1 (10%) of the values are in the interval [-1,+1] and 1-0.1 (90%) of the values are in the interval [-0.2,+0.2].

Mutation consists in the application of *sumGene* to the genes of a chromosome $c^n \in A^n$ with the values of a vector of mutation $v^n \in G_p^n$. Formally, *mutation* : $A^n \times G_p^n \rightarrow A^n$ is defined by *mutation* $= lc^n.lv^n.< sumGene(c_1^n, v_1^n), ..., sumGene(c_2^n, v_2^n), sumGene(c_n^n, v_n^n) >$.

The *reproduction of a pair of chromosomes* is an operation that generates a new pair of chromosomes. It's given by the function *reproduction* : $A^n \times A^n \rightarrow A^n \times A^n$ where $reproduction = lc1n.lc2n.(let\ i = random*n,\ let\ v1n = vmut(0.1,1.0,0.2),\ let\ v2n = vmut(0.1,1.0,0.2)\ in\ (let\ (p1n, p2n) = crossover(c1n, c2n, i)\ in\ (mutation(p1n, v1n), mutation(p2n, v2n))))$.

4.3 Neurons

A neuron is defined as a pair <cn,l>, where cn Î An and l Î Sp. The element cn is a chromosome,

that substitutes the weight vector in the basic model of neural nets. The improper genes of that chromosome denote weights that are not applicable. Therefore, the neuron is not sensitive to the entries at those positions. The element *l* corresponds to an activation threshold for the neuron, that is, an excitation level above that the neuron triggers. The domain of neurons of size *n* will be denoted by: *Neuronn = An ´ Sp*.

The operations of *decomposition* of this domain are: *chromosome* : *Neuronn* ® *An* where *chromosome*(<*cn,l*>)=*cn*, and *threshold* : *Neuronn* ® *Sp*, where *threshold*(<*cn,l*>)=*l*.

The *application* of a neuron <*cn,l*> to a sensorial entry *sn* Î S_p^n generates a signal in the interval [0,1], that corresponds to the difference between the similarity of *sn* and *cn* by the value of the threshold *l*. If this difference is negative, the signal produced will be zero. Formally: *application* : S_p^n ´ *Neuronn* ® *Sp*, where *application* = *lsn.lrn.max*(0, *similarity*(*sn, chromosome*(*rn*)) - *threshold*(*rn*)).

The *adjustment* of a neuron to a sensorial entry is given by the adjustment of each of its genes. This function is given by: *adjust* : *Neuronn* ´ S_p^n ´ *Sp* ® *Neuronn*, where *adjust* = *l*<*cn,l*>.*len.la.*<<*adjustGene*(c_1^n, e_1^n ,*a*), ..., *adjustGene*(c_n^n, e_n^n ,*a*)>,*l*>.

4.4 Schemata

A *schema* consists of a tuple <<c_E^n, *lE*>, <c_O^n, *lO*>, *am, a, n*>, where <c_E^n, *lE* > is a sensorial entry neuron, <c_O^n, *lO* > is a goal neuron, *am* Î *Am* is an action chromosome, *a* Î *Sp* is a learning value, and n Î *Sp* is an evaluation value. The schemata domain will be denoted by *Scheman,m = Neuronn ´ Neuronn ´ Am ´ Sp ´ Sp.*

The decomposition operations of this domain are: *entry* : *Scheman.m* ® *Neuronn,* where *entry*(<<c_E^n, *lE*>, <c_O^n, *lO*>, *am, a, n*>) = <c_E^n, *l*>, *goal* : *Scheman,m* ® *Neuronn*, where *goal* (<<c_E^n, *lE* >, <c_O^n, *lO*>, *am, a, n*>) = <c_O^n, *lO*>, *action* : *Scheman,m* ® *Am* where *action*(<<c_E^n, *lE* >, <c_O^n, *lO* >, *am, a, n*>)= *am, learning* : *Scheman,m* ® *Sp*, where *learning*(<<c_E^n, *lE* >, <c_O^n, *lO* >, *am, a, n*>)=*a,* and *evaluation* : *Scheman,m* ® *Sp*, where *evaluation*(<<c_E^n, *lE* >, <c_O^n, *lO* >, *am, a, n*>)=*n*.

The *training* of a schema <<c_E^n, *lE*>, <c_O^n, *lO*>, *am, a, n*>, relatively to a pair entry/goal <s_E^n, s_O^n>, where s_E^n ∈ S_p^n and s_O^n ∈ S_p^n corresponds to a new schema in which the neurons <c_E^n, *lE*> and <c_O^n, *lO*> were adjusted to the entries s_E^n and s_O^n respectively, by a rate defined by *a*. Formally, the function *training* : *Scheman,m* ´ S_p^n ´ S_p^n ® *Scheman,m,* is given by *training* = *lesqn,m.ls*$_E^n$*.ls*$_O^n$*.* <*adjust*(*entry*(*esqn,m*), s_E^n, *learning*(*esqn,m*)), *adjust*(*goal*(*esqn,m*), s_O^n, *learning*(*esqn,m*)), *action*(*esqn,m*), *learning*(*esqn,m*), *evaluation*(*esqn,m*)>.

The *attenuation of the training* of a schema consists on reducing the value *a* by a rate *e* Î *Sp*. The attenuation rate *e* for *a* is reduced at each cycle of iteration. The function *attenuation* : *Scheman,m ´ Sp* ® *Scheman,m* is defined by: *attenuation = lesqn,m. le.* <*entry*(*esqn,m*), *goal*(*esqn,m*), *action*(*esqn,m*), *e ´ learning*(*esqn,m*), *evaluation*(*esqn,m*)>.

The evaluation of a schema is given by the value of *n*. A *re-evaluation* of a schema relative to a pair entry/goal <s_E^n, s_O^n>Î S_p^n ´ S_p^n consists on the weighted average of *n* with the similarity of the entry

s_O^n and the chromosome of the goal neuron of the schema c_O^n. This average is weighted relatively to the similarity of the entry s_E^n and the chromosome of the entry neuron c_E^n, that is, as great is the value of $similarity(s_E^n, c_E^n)$ greater is the participation of the value $similarity(s_O^n, c_O^n)$ in the calculation of the average. The function *re-evaluation* : *Scheman,m* ´ S_p^n ´ S_p^n ® *Scheman,m* is given by *re-evaluation* = *lesqn,m*. ls_E^n. ls_O^n. <*entry*(*esqn,m*), *goal*(*esqn,m*), *action*(*esqn,m*), *learning*(*esqn,m*), (*evaluation*(*esqn,m*) ´ (1 - *similarity*(s_E^n, *entry*(*esqn,m*)) + *similarity*(s_O^n, *goal*(*esqn,m*)) * *similarity*(s_E^n, *entry*(*esqn,m*)))>.

The *reproduction of a pair of schemata* consists in the reproduction of the components of the schemata. Thus, to reproduce two schemata *e*1*n,m* and *e*2*n,m* it is necessary to combine the entry and goal neurons, the action chromosome, and the other values of each schema.

Let *e*1*n,m* = <<$c1_E^n$,$l1_E$, >, <$c1_O^n$,$l1_O$>, *a*1*m*, *a*1, *n*1> and *e*2*n,m*= <<$c2_E^n$,$l2_E$>, <$c2_O^n$,$l2_O$>, *a*2 *m*, *a*2, *n*2> be schemata. The reproduction of the schemata *e*1*n,m* and *e*2*n,m* must originate a new pair of schemata *e*3*n,m* = <<$c3_E^n$,$l3_E$>, <$c3_O^n$,$l3_O$>, *a*3*m*, *a*3, *n*3> and *e*4*n,m* = <<$c4_E^n$,$l4_E$>, <$c4_O^n$,$l4_O$>, *a*4 *m*, *a*4, *n*4> on the following form:

a) ($c3_E^n$,$c4_E^n$) = *reproduction*($c1_E^n$, $c2_E^n$);

b) ($c3_O^n$,$c4_O^n$) = *reproduction*($c1_O^n$,$c2_O^n$);

c) (*a*3*m*,*a*4 *m*) = *reproduction*(*a*1 *m*,*a*2 *m*);

d) The pairs of activation threshold ($l3_E$,$l4_E$) and ($l3_O$, $l4_O$) can be generated from ($l1_E$,$l2_E$) and ($l1_O$,$l2_O$) by a genetic process. It's possible to represent the rational values as chromosomes in the binary alphabet {0,1}, that correspond to the more basic model of genetic algorithms. The reproduction of these values can be done using crossover and mutation.

e) The values *a*3 and *a*4 must be fixed near to their upper bound (it may be used a value near to 0.7), because the schemata *e*3*n,m* and *e*4*n,m* were not tested in the learning process.

f) The values of *n*3 and *n*4 must be fixed at an arbitrary value lower than 1/3. Therefore, a recently created schema will have an evaluation value a bit smaller than the average obtained randomly for this value. If the destruction process is guided by this value, it will destroy the neurons that have been tested before the ones that haven't.

4.5 Populations

A *population* consists of a set of schemata. Some operations will be defined on populations. The domain *Populationn,m* will be defined as a powerset of the domain *Scheman,m*, that is, $Population^{n,m} = 2^{Schemd^{n,m}}$.

The *selection* of a schema from a population *Qn,m* Î *Populationn,m*, given a sensorial entry *en* Î

S_p^n, is done relatively to the value obtained by the application of each neuron $<ci_E^n, li_E>$ to the entry. The probability of a schema *ein* to be selected is given by the rate of the application value $<ci_E^n, li_E>$ relatively to the summation of the values of $<cj_E^n, lj_E>$(*where* 1 £ *j* £ |*Qn,m*|). The function *selection* : *Populationn,m* ´ S_p^n ® *Scheman,m*, is given by *selection* = *lQn,m.lsn.*(*en,m such that en,m is selected among the schemata of Qn,m with probability given by* $\dfrac{aplication\ (e^{n,m}, s^n)}{\sum_{j=1}^{n} aplication\ (ej^{n,m}, s^n)}$.

5 Non-Supervised Sensory-Motor Agent

A *non-supervised sensory-motor agent* may be seen as a set of signal sensors, a set of signal-based effectors and a population of schemata *Qn,m*.

A sensory-motor agent *An,m* lives in an environment *En,m*, that provides *n* signals to the agent's sensors, and it suffers actions composed by *m* signals coming from agent's effectors.

An agent may be denoted by *An,m* = <*Q n,m,e m* >, where *Qn,m* are schemata, and *em* are signals present in the agent's effectors, that are passed to the environment. The agent's environment may be denoted by *Em,n* = <*Rm,n,sn*>, where *sn* is a set of entry signals for the agent, and *Rm,n* is a set of environment schemata.

Such schemata are not necessarily defined as the agent's schemata. If the environment has a physical nature, those schemata are physical relations between objects, that is, the observed cause/effect relations.

Let $s_{|t|}^n \in S_p^n$ and $e_{|t|}^m \in S_p^m$ be representations of signals coming from the agent's sensors and signals going to the physical effectors, respectively, in an instant of time *t*. Then, $<Q_{|t|}^{n,m}, e_{|t|}^m>$, is an *instant description of a cognitive agent* in the instant of time *t*, and $<R_{|t|}^{m,n}, s_{|t|}^n>$ is an *instant description of an environment* which the agent interacts in the instant *t*.

5.1 Sensory-Motor Agent's Working

The working of a non-supervised sensory-motor agent *An,m* = <*Qn,m,em*> in an environment *Em,n* = <*Rm,n,sn*> is given by the following algorithm:

Step 1: Selection. The agent observes the sensorial entry $s_{|t|}^n$ provided by the environment. Then the agent selects a winner schema for activation by $winner_{|t|}^{n,m} := selection\ (Q_{|t|}^{n,m}, s_{|t|}^n)$.

Step 2: Activation. The agent activates $winner_{|t|}^{n,m}$ by putting on the correspondent positions in

$e^{m}_{|t+1|}$ the proper signals (those different of #) of the chromosome *action*($winner^{n,m}_{|t|}$).

Step 3: Adjust. The agent adjusts the weights of the entry and goal neurons accordingly to the data provided by the environment: *entry*($winner^{n,m}_{|t|}$) := *adjust*($winner^{n,m}_{|t|}$, $s^{n}_{|t|}$, *learning*($winner^{n,m}_{|t|}$); and *goal*($winner^{n,m}_{|t|}$) := *adjust*($winner^{n,m}_{|t|}$, $s^{n}_{|t+1|}$, *learning*($winner^{n,m}_{|t|}$).

Step 4: Attenuation. The agent decrements the learning value to attenuating the adjusting rate of the schema: $winner^{n,m}_{|t|}$:= *attenuation*($winner^{n,m}_{|t|}$, 0.95). (Here, it was used an attenuation of 5%, but other values may be used).

Step 5: New evaluation. The agent observes again the sensorial entry after the effectors activation. Let this observation be represented by $s^{n}_{|t+1|}$. The agent calculates the new evaluation of $winner^{n,m}_{|t|}$ by $winner^{n,m}_{|t|}$:= *re-evaluation*($winner^{n,m}_{|t|}$, $s^{n}_{|t|}$, $s^{n}_{|t+1|}$).

Step 6: Reproduction. If the re-evaluation of $winner^{n,m}_{|t|}$ reduced the value of *evaluation*($winner^{n,m}_{|t|}$), then the agent makes the reproduction of $winner^{n,m}_{|t|}$ with another schema given by *selection*($Q^{n,m}_{|t+1|}$, $s^{n}_{|t|}$). The agent adds the new pair of schemata to $Q^{n,m}_{|t+1|}$, and eliminates from $Q^{n,m}_{|t+1|}$ the two schemata with the smallest values for n, and then returns to the step 1.

This working indicates a behavior where concepts are identified with activity schemata. Initially, each schema is randomly generated, so that the actions executed by it make no sense. As the entry and goal neurons are adjusted, the schema become able to identify the proper results of its actions, but it can not create new actions.

The constant adjustment of goal neurons increments gradually the evaluation of many schemata. In some moment, however, the action *am* applied to a certain entry may not product the desired effect, lowering the schema evaluation, and performing the role of a *perturbation*, as defined by Piaget [PIA 76]. Those perturbations begin schemata reproduction (Piaget has called this process *differentiation*). The reproduction doesn't destroy the schema that has suffered the perturbation, but the schemata with the lowest evaluation. Thus, the perturbed schema is maintained, and a pair of new schemata is created by reproduction. These new schemata will try to compensate the perturbation.

In the case of reproduction with the perturbed schema, the weighted selection of schemata of highest value is in accord with the genetic algorithm process. This makes the agent to search some good schemata in order to try to accommodate the perturbation.

5.2 Effects of the Sensory-Motor Agent's Working

The effect of a sensory-motor agent $A^{n,m} = < Q^{n,m}, e^{m} >$ in an instant t, when it's ready to act in the instant t+1 on an environment $< R^{m,n}, s^{n} >$, follows the laws of environmental operations in the

instant t, given by $R^{m,n}_{[t]}$, which determine how the actions $e^{m}_{[t+1]}$ produced by the agent in the instant t+1 affect the sensorial entries $s^{n}_{[t+1]}$ that the environment provides to the agent in the instant t+1.

The interaction process between agent and environment may be though of as a process going on through the evolution of the system *Agent Environment*, represented by $<An,m,Em,n>$. The evolution of this pair is denoted by a sequence of cycles of the form $<<Q^{n,m}_{[t]}, e^{m}_{[t]}>,< R^{m,n}_{[t]}, s^{n}_{[t]}>> \rightarrow_A <<Q^{n,m}_{[t]}, e^{m}_{[t+1]}>,< R^{m,n}_{[t]}, s^{n}_{[t]}>> \rightarrow_E <<Q^{m}_{[t]}, e^{m}_{[t+1]}>,< R^{m,n}_{[t+1]}, s^{n}_{[t+1]}>> \rightarrow_A <<Q^{n,m}_{[t+1]}, e^{m}_{[t+1]}>,< R^{m,n}_{[t+1]}, s^{n}_{[t+1]}>>$ where $\rightarrow_A$ represent steps of the agent's working, and $\rightarrow_E$ represent steps of the environment's working.

The sequence of those steps describes the join working of the pair agent/environment. It indicates how agent's actions modify its environment, and how these modifications affects the agent's schemata, and consequently it's sensory-motor capacity. Thus, it represents the agent's cognitive development, in interaction with its environment, and captures, in a certain sense, some processes identified by genetic epistemology.

6 Conclusions

The main contribution of this model is the combination of neural networks and genetic algorithms resulting in a non-supervised learning mechanism able to self-adaptation to environmental modifications.

The schema mechanism approximates a certain kind of human classification method. Concepts are associated to schemata, and schemata are built from observables, actions and goals. If the action reaches the schema's goal, then the observable belongs to the concept associated to the schema; otherwise the schema must be differentiated. The differentiation occurs when a couple of schemata is genetically reproduced. In this case, the action, observables and goals are recombined for creating new schemata, which try to better represent the structure of activity related to the environmental objects. The genetic mechanism is responsible for finding the best configurations for the tuple *observable action goal*.

Schemata that repeatedly fail in the search for their goals will have their fitness reduced. On the other side, schemata that reach their goals will have their fitness incremented. Thus, schemata that are not representing correct concepts will be destroyed while others are preserved.

An important aspect in a human classification method is the capacity of concept generalization and specialization. Normally, an exemplar belongs to many classes simultaneously. For instance, a cat is at the same time identified as a mammal, a vertebrate, an animal, etc. The schema mechanism approximates this generalization effect because it simulates a many-winners competitive neural network [WAZ 93]. Certain neuron configurations may be considered specialization of another. Given a neuron (cn, l), then a neuron (dn, k) is a *specialization* of (cn, l) if every proper genes of dn are similar to the genes of cn, and if l is similar to k. Saying that "x is similar to y" means a diffuse equality, that would be defined, as $| x - y |$ 0.2. On this sense, a schema is a specialization of another when it incorporates a new characteristic that doesn't exists in the other schema.

Moreover, the value of *l* defines a certain kind of specialization, but in another sense. A schema (*cn, l*) is a specialization of (*dn, k*) when *l* is greater than *k*, and *cn* and *dn* are similar and have the same quantity of improper genes.

This paper advances the formal study of the relationship between interactive working of sensory-motor agents in environments whose characteristics are known and the process of cognitive development of those agents [COS 93]. In particular, it's our interest to study in a near future the role that the interaction process may play in the passage from the sensory-motor level (described here) to the cognitive level, where symbolic representations are handled. This study will verify whether the proposed schema mechanism suffices to handle that level changing, or whose new mechanisms are required.

7 References

[COS 93] COSTA, Antônio Carlos da Rocha *Machine Intelligence: Sketch of a Construtivist Approach.* Ph.D. Thesis, Porto Alegre: CPGCC-UFRGS, 1993.

[DAY 90] DAYHOFF, Judith E. *Neural Network Architectures - an introduction.* New York: Van Nostrand Reinhold, 1990.

[DEM 93] DEMAZEAU, Yves, MÜLLER, Jean Pierre *Decentralized Artificial Intelligence 1.* North-Holland, Elsevier Science Publishers, 1990.

[GOL 89] GOLDBERG, David *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading: Addison-Wesley, 1989.

[HOL 75] HOLLAND, John *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[PIA 76] PIAGET, Jean *The Equilibration of Cognitive Structures - The Central Problem of Cognitive Development.* Rio de Janeiro: Zahar, 1976.

[RAM 94] RAMOS, Edla M. F. *Personal Communication.* Florianópolis, 1994.

[WAZ 93] WAZLAWICK, Raul S. *An Operatory Model for Knowledge Construction.* Ph.D. Thesis, Florianópolis: PGEP-UFSC, 1993.