Lifelong Machine Learning with Adaptive Multi-Agent Systems

Nicolas Verstaevel, Jérémy Boes, Julien Nigon, Dorian d'Amico, Marie-Pierre Gleizes

Lifelong Machine Learning with Adaptive Multi-Agent Systems

Nicolas Verstaevel, Jérémy Boes, Julien Nigon,
Dorian d'Amico and Marie-Pierre Gleizes
IRIT, Université de Toulouse, Toulouse, France
name.surname@irit.fr

Keywords: ambient systems : multi-agent systems : lifelong learning : self-adaptive systems : self-organization

Abstract: Sensors and actuators are progressively invading our everyday life, as well as industrial processes. They form complex and pervasive systems usually called ”ambient systems” or ”cyber-physical systems”. These systems are supposed to efficiently perform various and dynamic tasks in an ever-changing environment. They need to be able to learn and to self-adapt throughout their life, because designers cannot specify *a priori* all the interactions and situations they will face. These are strong requirements that push the need for lifelong machine learning, where devices can learn models and behaviors and transfer them to perform other tasks. This article presents a multi-agent approach for lifelong machine learning.

Contents

1 INTRODUCTION

The rapid increasing of the computational capabilities of electronic components and the drastic reduction of their costs has allowed to populate our environments with thousands of *smart devices*, inducing a revolution in the way we interact with a more and more numerical world. The emergence of a socio-technical world where technology is a key component of our environment impacts many aspects of our existence, changing the way we live, work and interact. Applications of these *Ambient Systems* are wide and various, covering a spectrum from industrial applications to end-user satisfaction in a domotic context (Nigon et al., 2016a). The will to improve our manufacturing processes through the *Industry 4.0* topic (Jazdi, 2014), or the emergence of a *Green Economy* (Rifkin, 2016) are good illustrations of the usage of these cyber-physical systems.

A key feature of these systems is their pervasivity: they are composed of a huge variety of electronic components, with sensing and actuating capacities, that interact with each other, generating more and more complex data all in order to control dynamic and inter-disciplinary processes. They achieve their task throughout their life without any interruption. The complexity of these systems, increased by the heterogeneity of their components and the difficulty to specify *a priori* all the interactions that may occur, induces the need for lifelong learning features.

In this paper, we address the lifelong learning problem in ambient systems as a problem of self-organization of the different components. Giving each component the ability to self-adapt to both the task and the environment is the key toward a lifelong learning process. First, we argue that work on lifelong machine learning and self-organization shares similar objectives. Then, we propose a generic framework, based on our previous and current works on *Adaptive Multi-Agent Systems*, that enable lifelong learning through interactions in a set of distributed agents. We then study the relevance of our approach in a simulation of a robotic toy. Finally, we conclude with the challenges and perspectives opened by this work.

2 AMBIENT SYSTEMS, LIFELONG MACHINE LEARNING AND SELF-ORGANIZATION

In this section, we introduce the basics of lifelong machine learning and self-organization in order to highlight the similar problematic addressed by these two domains.

2.1 Machine Learning and Ambient Systems

Machine learning is an increasingly trending topic in the industry, becoming an indispensable tool to design artificial systems. Designers use machine learning to face the inability to specify a priori all the interactions that could occur or to handle the complexity of a particular task. The idea of designing learning machines lies at the very heart of artificial intelligence and is studied since the very beginning of computer science. (Mitchell, 2006) defines the field of machine learning as the science seeking to answer the questions ”how can we build computer systems that automatically improve with experience?” and ”what are the fundamental laws that governs all learning process?”. According to (Mitchell, 2006), a machine is said to learn with respect to a particular task T, performance metric P, and type of experience E, if the system reliability improves its performance P at task T, following experience E. The field of Machine Learning is traditionally classified in three categories (Russell et al., 2003), depending on how the experience E is performed:

- **Supervised Learning** which infers a function from labelled data provided by an *oracle*.
- **Unsupervised Learning** which looks for hidden structures in unlabelled data provided by an external entity.
- **Reinforcement Learning** which learns through the interaction between a learner and its environment to maximize a utility function.

While the difference is on the way the experience is gathered, all these learning algorithms intend to learn a particular task. The learning system is then specialised through its experience to increase its performance. The notion of task in the AI community is highly discussed, and in the absence of a common definition or test framework, it is difficult to truly compare those different approaches (Thorisson et al., 2016).

Learning in an ambient system differs in the way that the task to learn is a priori unknown and dynamic. These systems must handle unanticipated tasks whose specifics cannot be known beforehand. Indeed, these systems are characterized by: their non-linearity (the behaviour of such systems is not defined by simple mathematical rules), the high number of entities that compose it, the high number of interactions between these entities, their unpredictability. In addition, ambient systems are generally used in conjunction with the presence of human users. Those human users have varied and multiple needs, which implies to deal at runtime with various tasks. Moreover, the devices of ambient systems are made to last several years, during which their usage may change. We have no way to know to what end they will be used five or ten years from now. Thus, ambient systems require that machine learning algorithms handle the dynamics of their environments and never stop learning. On the next sections, we introduce lifelong machine learning and draw a parallel with self-organization.

2.2 Lifelong Learning Machine: Definitions and Objectives

Definition: *Lifelong Machine Learning, or LML, considers systems that can learn many tasks over a lifetime from one or more domains. They efficiently and effectively retain the knowledge they have learned and use that knowledge to more efficiently and effectively learn new tasks.* (Silver et al., 2013)

The idea of design artificial systems with Lifelong Learning capacities is not new (Thrun and Mitchell, 1995), but their is a upsurge work probably helped by the recent advances in neural networks (Pentina and Lampert, 2015).

Artificial systems have to face more and more dynamic and various environments. The impact of those quick evolutions impose severe limitations on designers capacities to a priori determine all the events that can occur in the environment. While traditional machine learning focuses on optimising a system performance to achieve a particular task, LML proposes to deal with various tasks in a variety of environment.

Researches on LML are then multidisciplinary, associating a wide range of researchers from computer sciences to cognitive sciences. Learning is considered as an ontogenesis process from which more and more complex structures are built through experience and LML addresses the problematic of reusing previously learned knowledge (Zhang, 2014).

(Thrun and Mitchell, 1995) categorise LML approaches in two categories, depending on the hypothesis put on the environment. The first one considers learners that have to learn various tasks in similar environments. Those algorithms build an action model which learns the impact of the learner’s action on the environment. As the environment is the same, this model is independent of the task to perform and can be used in various tasks. The second one considers that the same learner as to face multiple environments. This approach intends to model invariant features that are observed about the learner on its environment. In both categories, the objective is to build through experience a model from the different experiences and to transfer or reuse previously learned knowledge in order to speed up the learning process when the learner has to learn a new task.

2.3 Machine Learning and Self-Organization

The behavior and the interactions of the components of a system define the function of the system. When the components change their behavior or their interactions, the function of the system is necessarily modified. If these changes are only controlled by the components themselves, the system is said to be self-organizing. The concept of self-organization is studied in various domains, from biology to computer science. The following definition is given by (Serugendo et al., 2011).

Definition: *Self-organization is the process with which a system changes its structure without any external control to respond to changes in its operating conditions and its environment.*

In other words, self-organization enables a system to change its internal behavior in order to reach and maintain efficient interactions with its environment. This is very close to machine learning, where a software system tries to improve its function by adjusting itself regarding past experiences. Self-organization is actually a fine way to achieve machine learning. It is particularly efficient when a system has many heterogeneous components with various degree of autonomy, such as an ambient system and its many connected devices. In this case, self-organization implies the distribution of control and helps to tackle complexity by focusing on the local problems of the components rather than the global task of the system. This is why designing self-organizing systems is a key challenge to tackle complexity in ambient systems.

Multi-agent systems are particularly suitable paradigm to design and implement self-organizing systems. The next section presents our self-organizing multi-agent pattern for machine learning.

3 LIFELONG MACHINE LEARNING WITH ADAPTIVE MULTI-AGENT SYSTEMS

3.1 Adaptive Multi-Agent Systems

The Adaptive Multi-Agent Systems (AMAS) approach aims at solving problems in dynamic non-linear environments by a bottom-up design of cooperative agents, where cooperation is the engine of the self-organisation process (Georgé et al., 2011). Over the years, the approach has been applied to designed and developed various self-adaptive multi-agent systems to tackle real-world complex problems, such as robot control (Verstaevel et al., 2016) or heat engine optimization (Boes et al., 2014). A recurrent key feature of these systems is their ability to continuously learn how to handle the context they are plunged in, in other words to map the current state of their perceptions to actions and effects. In these applications, a set of distributed agents locally learns and exploits a model of an agent behavior from the interactions between the agent and its environment. On the next section, we present the Self-Adaptive Context Pattern (Boes et al., 2015), our proposal to design agents with lifelong learning capacities.

3.2 The Self-Adaptive Context Learning Pattern

The Self-Adaptive Context Learning Pattern is a recurrent pattern, designed with the AMAS approach, that we have been using to solve complex problems based on the autonomous observation by an agent of the impacts of its own activity on the environment (Nigon et al., 2016b). The pattern is composed of two coupled entities:

- An *Exploitation Mechanism*, which function is to perform actions over the environment. It is the acting entity which has to decide and apply the action to performed by a controlled system in the current context. Its decision is based on its own knowledge, including constraints from the application domain, and additional information provided by the Adaptation Mechanism.
- An *Adaptation Mechanism*, which builds and maintains a model describing the current context of the environment and its possible evolutions. This model is dynamically built by correlating the activity of the Exploitation Mechanism to the observation of the environment.

The *Adaptation Mechanism* and the *Exploitation Mechanism* are coupled entities and form a control system (Figure 1). The *Adaptation Mechanism* perceives information from the environment (Arrow 3) and uses this information to provide information about the current system context to the *Exploitation Mechanism*. Thanks to those information and its own internal knowledge, the *Exploitation Mechanism* decides of the action to perform and applies it (Arrow 2). The Exploitation Mechanism provides a feedback to the *Adaptation Mechanism* about the adequacy of the information performed (Arrow 4). The adequacy is evaluated by the observation of the environment by the *Exploitation Mechanism*. This pattern

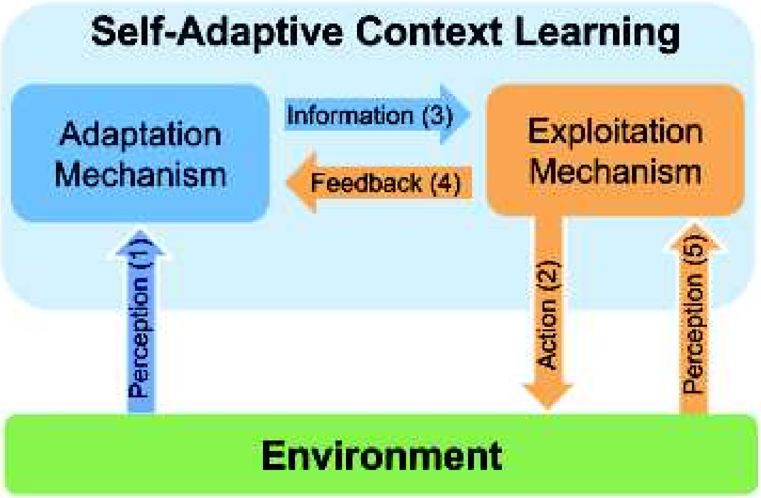

Figure 1: The Self-Adaptive Context Learning Pattern: view of one agent and its interaction with the environment (Boes et al., 2015)

is highly generic and task-independent. The *Adaptation Mechanism* models information that are useful for the *Exploitation Mechanism*, which exploits this information to decide of the action to performed. By its actions, the *Exploitation Mechanism* modifies the environment, enabling the *Adaptation Mechanism* to update its knowledge about the environment and to provide new information about the current context. This loop enables the system to continuously acquire knowledge about its environment through its own experience. Through this loop, the Adaptation Mechanism models information about the interaction between the Exploitation Mechanism and its environment. This information is autonomously provided to the Exploitation Mechanism which uses it to improve its behavior.

The pattern results of an abstraction of various application of the AMAS approach to different problems. Then, it is independent of the implementation of the Adaptation Mechanism and the Exploitation Mechanism. It only focuses on how those two entities has to interact in order to learn over time, throughout their activity. In this pattern, the learning is made by the Adaptation Mechanism, whereas the acting is made by the Exploitation Mechanism. However, to be truly functional, the SACL pattern imposes that the Adaptation Mechanism continuously learns from the interactions in a Lifelong way.

3.3 Solving problems with SACL

The pattern has been applied to various problems where learning is a way toward self-organisation (Boes et al., 2015). In each of those problems, the recurrent key feature is to learn a model of the interactions between the agent and its environment by extracting the context in which those actions are performed and associating information about this context in order to lately use this information to behave well. To illustrate this recurrent key feature, we show how the pattern is used to design Supervised Learning agents and Reinforcement Learning agents, and we point out similarities of design and differences.

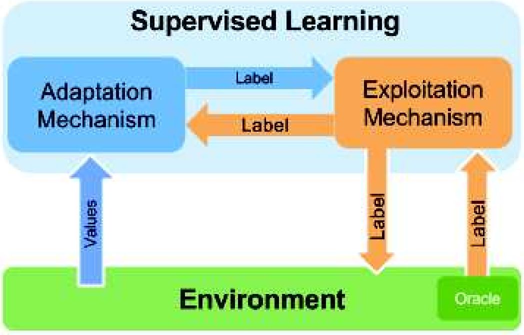

Figure 2: The Self-Adaptive Context Learning Pattern applied to Supervised Learning problems.

3.3.1 Supervised Learning Agent

A supervised learning algorithm infers a function from labelled data provided by an *oracle*. Supervised learning algorithms perform either a *classification* or a *regression*, depending on either the output is discrete or continuous. Traditionally, Supervised Learning algorithms differentiate the learning phase, where labelled situation are provided by the oracle, with the exploitation phase, where new situations have to be labelled by the agent itself.

With the SACL pattern, those two phases are not separated. The supervised learning process is performed on-line. In order to do so, the *oracle* is considered as an autonomous entity which takes part of the environment which may provide information to the Exploitation Mechanism. The function of the SACL pattern is then to mimic the behavior of this external entity. Thus, the Exploitation Mechanism observes the behavior of the oracle, and when the oracle provides a label describing a situation, it compares this label given by the *oracle* to the label proposed by the Adaptation Mechanism. Then, it generates a feedback toward the Adaptation Mechanism to inform if either or not the current situation is correctly mapped by the Adaptation Mechanism. The Adaptation Mechanism uses this feedback and the observation of the data that describes the current states of the environment in order to adjust its model. Whenever an oracle provides information about the current situation, the output of the SACL agent is the label of the oracle. On the other hand, when the oracle does not give information about the current situation, the Exploitation Mechanism exploits the information provided by the Adaptation Mechanism to label the current situation. The figure 2 summarizes the usage of the SACL pattern for the design of Supervised Learning agents.

Thus, when performing supervised learning with SACL, the Adaptation Mechanism has to associate the adequate label to the current state of the environment, in other words, to extract the context of each label in order to be able to correctly propose the adequate label to its Exploitation Mechanism when the oracle does not provide this label.

This form of learning with the SACL pattern has been applied to the problematic of learning user preferences in ambient systems (Guivarch et al., 2012) and learning from a set of demonstrations performed by a human operator in ambient robotic applications (Verstaevel, 2016). In those applications, examples are provided by the oracle throughout the activity of the system, without restarting or clearing the learning process.

3.3.2 Reinforcement Learning Agent

Reinforcement Learning algorithms learn through their own experience to maximize a utility function. Basically, each action is associated with a numerical reward provided by the environment which evaluates the utility of the performance of this particular action in the current context. Learning a model of the consequence of the performance of an action to the utility value allows the agent to select actions in order to maximize its reward.

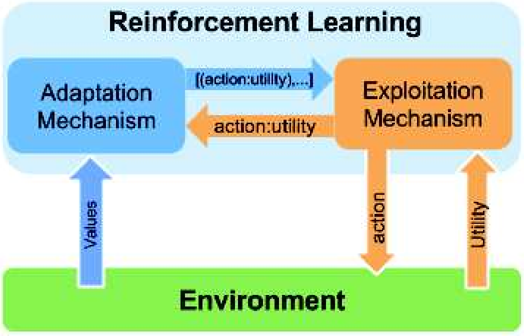

Figure 3: The Self-Adaptive Context Learning Pattern applied to Reinforcement Learning problems.

The figure 3 illustrates how we used the SACL pattern to design a Reinforcement Learning agent. Here, the Exploitation Mechanism performs actions on the environment and receives a feedback about the utility of this action. The action and utility value are provided to the Adaptation Mechanism which associates the current state of the environment and the performance of the action the utility value. The Adaptation Mechanism is used to build a model of the consequences of performing a particular action under a certain context. This model is then used to provide to the Exploitation Mechanism information about the consequences of performing a set of actiosn on the current context. Using this knowledge, the Exploitation Mechanism can select the action which maximizes its reward.

Similarly to the Supervised Learning agent, the Adaptation Mechanism observes both the current state of the environment and the activity of the Exploitation Mechanism. It dynamically builds a model of the interaction between the Exploitation Mechanism and its environment. This information is then used in order to improve the Exploitation Mechanism behavior.

Reinforcement Learning agents with SACL have been used for the optimization of heat-engines control (Boes et al., 2014).

3.4 AMOEBA: Agnostic MOdEl Builder by self-Adaptation

Building a model of the interactions that occurs between the Exploitation Mechanism and the environment is the core of the SACL pattern and its learning process. The Adaptation Mechanism has to dynamically model these interactions. In order to do so, we designed a generic Multi-Agent system based on the AMAS approach, AMOEBA (for Agnostic MOdEl Builder by self-Adaptation), for building model of complex Systems. In AMOEBA, a set of Context agents is dynamically created to discretize the environmental state into events. To design this system, we made some hypotheses about the environment:

- The observed world is deterministic: the same causes produce the same effects.
- The environment perceived by AMOEBA is not the world as a whole. AMOEBA therefore has only an incomplete view of the world.
- Perceived information may be inaccurate.
- The collected data are orderable values (real, integer ...).
- The dynamics of evolution of the observed world variables are potentially very different from one to another.

In this section, we present the structure and behaviour of Context agents and the different mechanisms that enable Context agents to collectively build a model of the interactions. In a didactic way, we consider that the environmental state is observed through a set of sensors described by a vector $E \in R^n$. However, the model is functional with any orderable value. Basically, each component $e_i \in E$ describes the value of a particular data.

3.4.1 Context Agents: Definitions

Context Agents are the core of learning in AMOEBA *Context Agent* may be assimilated to a unit of knowledge which locally describes the evolution of a particular variable of the environment. At start, the system is empty of *Context agents* as it does not possess an a priori knowledge about the environment. *Context agents* are created at runtime to model new interactions. They are composed by a set of *validity ranges* and a *local model*.

Definition: A Context Agent $c \in C$ is composed of a set of validity range V and a local model L.
Each *validity range* v is an interval associated to a particular value $e_i \in E$ of the environmental state E. Each e_i is modelled by a *validity range* within each *Context Agent*. The intervals are defined by a minimal and a maximal float value.

Definition: A validity range $v_i \in V$ is associated to any value $e_i \in E$. A particular v is an interval $[v_{min}, v_{max}] \subset [e_{min}, e_{max}]$.
A validity range enables to discretize the environmental state into two types of events: *valid* or *invalid*. Those events are determined by comparing the current value of the variable e_i and the range of v.

Definition: A validity range $v_i \in V$ is said *valid* if and only if $ei \in v_i$ and *invalid* otherwise.
In the same way, a *Context Agent* is a discretization of the environmental state E into the two same type of events.

Definition: A Context Agent $c \in C$ is said *valid* if and only if $\forall v_i \in V, e_i \in [v_{min}, v_{max}]$ and *invalid* otherwise.
The *local model L* is a linear regression of the environmental state $L : E \rightarrow R$. Each *Context Agent* disposes of its own *local model* and thus, of its own linear regression.
The function of a *Context Agent* is dual:

- To send the value computed by its *local model* to the Exploitation Mechanism whenever the *Context Agent* is *valid* and,
- Thanks to the feedbacks from the *Exploitation Mechanism*, to adjust both its *local model* and its *validity ranges* so that the information sent to the *Exploitation Mechanism* is useful for its activity.

3.4.2 Context Agents: Creation and Self-Organisation

At start, AMOEBA does not contain any Context Agent. Indeed, all the Context Agents are going to be created and will build themselves using self-organization mechanisms. These changes / creations of agents take place when the agents are trouble to perform their function efficiently. The main adaptations performed by a Context Agent are :

- *Changing its validity range.* Indeed, when the results obtained by the local model are not satisfactory for the *Exploitation Mechanism*, it may be preferable for the agent to no longer be active in the current situation. To do this, it reduces its validity ranges. Conversely, if it finds that its proposal would have been relevant in a close situation, the agent may broaden its ranges of validity to include the situation in question (figure 5).
- *Changing its local model.* When the results obtained by the local model are not satisfactory for the *Exploitation Mechanism*, an other way to improve the results of the Context Agent is to change the local model. This consists in adding to the linear regression of the local model the point represented by the current situation. This change is smoother than the previous one, and will therefore be preferred by the Context Agent when the error is small.
- *Destroy itself.* When the validity ranges of the Context Agent are too small, the agent could con-

sider that it has no chances to be useful again. So, it destroys itself in order to save resources.

Another mechanism is implemented to complement those mentioned earlier. When no proposal is made by the Context Agents, or when all their propositions are false, *a new Context Agent is created* to carry the information when a similar situation occurs again. This is how new agents are added to the system.

3.4.3 Synthesis

AMOEBA is a generic model builder to model the evolution of a particular variable. It based on the Adaptive Multi-Agent approach. It's key feature is to be dynamically populated by a set of Context Agents which evolves dynamically to maintain an up-to-date model of the variables it observes.
The reader may see that Context Agents are similar to neurons in Artificial Neural Networks. Artificial Neural Networks are composed of interconnected neurons, where each neuron is a small computational unit with inputs, outputs, an internal state and parameters. One could see Context Agents as ”advanced” neurons, and the architecture of AMOEBA, where messages navigate from the environment to the different Context Agents, as a form of multilayer perceptron. However, the information within AMOEBA is not feed-forward (contrary to neural networks). Thus, the main difference between AMOEBA and Artificial Networks is the organization of the interaction between the entities. With artificial Neural Network, the topology of the network has to be fixed a priori in regards with the task to learn whereas within AMOEBA, agents self-organize and the topology (the number of agents and the way they interact) evolves dynamically through experience.

4 COMBINING SUPERVISED LEARNING AND REINFORCEMENT LEARNING: AN EXPERIMENT WITH A CHILDREN'S ROBOTIC TOY

We propose to study the usage of SACL in a Lifelong reinforcement learning problem. We put ourselves in the context of child entertainment with a robotic toy. Firstly, we discuss of the motivations for this experiment. Then, we present the experimental process and the result we obtained. At last, we conclude on some perspectives highlighted by this experiment.

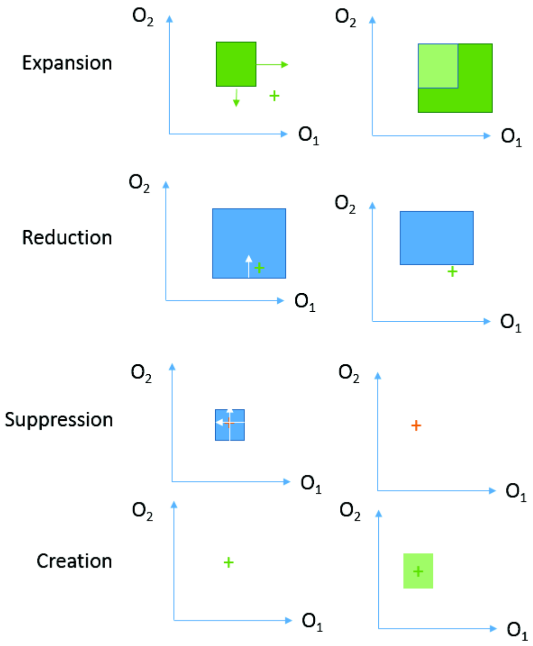

Figure 4: Representation of the different adaptation mechanisms of the Context Agents in two dimensions. Each dimension represents the values that can be taken by a toy sensor, and the size of the validity range determines the size of the rectangle along this dimension

4.1 Motivations

Robotic toys tend to be more democratised and the scientific community is working on their usage as an educational and therapeutic tool (Billard, 2003). Robins and Dautenhahn (Robins and Dautenhahn, 2007) have especially shown that their use encourages the interactions among autistic children. But as each human is different, the social skills needed to maintain the attention in the human/toy interaction requires the toys to be able to self-adapt to anybody (Huang, 2010).
To this extent, an interesting approach is the *Imitation Learning* (Billard, 2003) (Robins and Dautenhahn, 2007). *Imitation Learning* is a form of *Learning from Demonstration*, a paradigm mainly studied in robotics allowing a system to acquire new behaviors from the observation of human activity (Argall et al., 2009). It is inspired by the natural tendency in some animal species and human to learn from the observation of their congeners. Imitation has a major advantage: it is a natural step for the child development and a medium for social interactions (Piaget, 1962). To imitate the child then seems to be a good way to initiate and maintain the interactions. However the imitation poses a correspondence problem between what is ob-

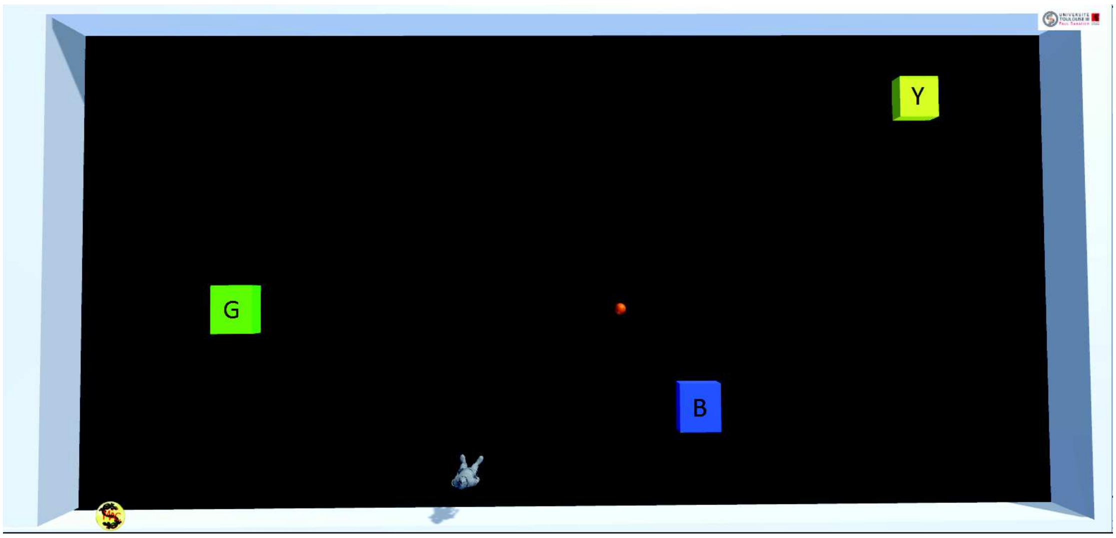

Figure 5: A bird's eye view of the simulation. A robotic ball is immersed in an arena composed of 3 coloured blocks. The contact between the toy and a block induces a laugh which is interpreted as a reward function. The long term objective is to maximize laugh production.

served (the object to imitate) and what can be done by the imitating entity (Nehaniv and Dautenhahn, 2002). Imitation also imposes the existence of a function allowing to associate the observed state of the world to the corresponding state of the toy. Such a function is complex and limits the interaction modalities.

Another form of *Learning from Demonstration* proposes to use *Reinforcement Learning* algorithms combined with *Learning from Demonstration* (Knox and Stone, 2009). *Reinforcement Learning* algorithms brings an interesting solution to the correspondence problem by letting the learning system discovering through its own experience what are the interesting interactions. Then, the toy can explore its environment and learn from its interactions with the child. However, in its original form, *Reinforcement Learning* algorithms require the existence of a feedback function which associates at each state of the world a reward value. This function is often complex to specify because it requires to evaluate *a priori* the distance to an objective. However, in our context, such objective is *a priori* unknown and has to be discovered. On the other hand, the *Inverse Reinforcement Learning* approach (Argall et al., 2009) proposes to infer this reward function from a set of reward examples. The system is fed with a set of situations and the associated rewards and the system infers a function that models this reward. This function can then be used to optimise a policy. But this solution implies that an external entity is able to provide the required demonstrations to learn the reward function which is, in the case being considered, non trivial. To face this challenge, some approach proposes to use child's laugh as a reinforcement metric (Niewiadomski et al., 2013).

Robotic toys that learn to interact with child are good illustration of the need for Lifelong Learning agents as maintaining children attention requires a constant adaptation to the child's needs.

4.2 Description of the experiment

We study a simulation with the game physic engine Unity3D (figure 5) in which a spherical robotic toy is immersed in an arena. A child, also in the arena, observes the behavior of the robotic toy. The arena is composed of 3 blocks of different colours. The contact between the toy and a block produces a sound, which depends on the texture of the block (determined by its colour). We postulate that the sound produced by the toy collision with an obstacle causes the child's laugh. The noise caused by the collision and the laugh intensity depends on the nature of the collided obstacle. Each object causes a different combination of high or low noise and strong or weak laugh. The laugh, observed through the volume intensity of the sound it produces, is used as a utility function and the objective is to maximize laugh over time.

The aim of the experiment is more to study how the AMOEBA model is built and maintained up to date during the experiment, than proposing a new reinforcement learning method.

4.3 Description of the SACL architecture

The SACL pattern is instantiated to perform this experimentation (figure 6). The Adaptation Mechanism is implemented with an instance of AMOEBA (as described in part 3.4). The Exploitation Mechanism is a simple controller, whose objective is to maximize the volume intensity of the environment. In order to do so, it has to decide of the (x,y) coordinates it wants to reach. A simple control loop determines, in function of the coordinate to reach, and the current state of the toy, which is the action to apply. The toy is controlled in velocity, which is then determined by the position to reach. In order to behave well, the Exploitation Mechanism has to receive from the AMOEBA instance a set of reachable positions and their associated volume. The AMOEBA instance will use as input the current x and y positions of the sphere and the current volume of the simulation, in order to build a model of the actions to be taken in order to increase the volume intensity according to the context. At each time step, the Context Agents inside AMOEBA will propose a set of coordinates (x,y) and their associated volume intensity to the Exploitation Mechanism. In this experiment, all the Context Agents inside AMOEBA are sent to the Exploitation Mechanism.

The selection of the coordinates to reach by the Exploitation Mechanism is based on a combination of three factors:

- The number of control loops elapsed since the last selection of the Context Agent δ. Each time a Context Agent is *valid*, the value δ is reset to 0. The Exploitation Mechanism selects the Context Agents which δ value is higher than 95% of the maximum δ value.
- The volume intensity computed by the Context Agent ω. The second selection process then selects among the previous Context Agent set, the Context Agents with a value ω higher than 95% of the maximum D value.
- The reachability of the coordinates proposed by the Context Agent D, which is here computed with the Euclidean distance. At last, the Exploitation Mechanism selects the Context Agent which proposes to reach the closest position.

Once the coordinates are chosen, the control loop is applied in order to reach those coordinates. The selection is performed at each time step of the control loop, allowing the Exploitation Mechanism to continuously update its coordinate objectives. At the same time the AMOEBA updates its model by observing the evolution of the environment.

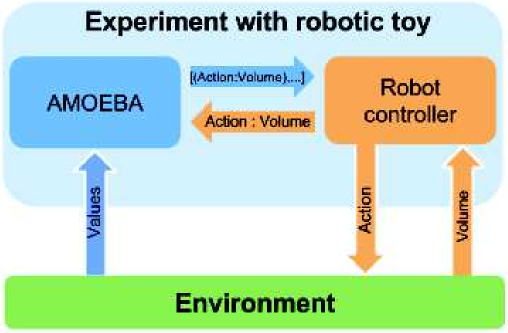

Figure 6: The Self-Adaptive Context Learning Pattern implemented in order to control a robotic toy.

4.4 Experimentation and Results

The experiment is carried out in two phases of three minutes. At first, an exploration phase allows the robotic toy to discover its environment. This phase is equivalent to a supervised learning approach where examples are provided to the learning algorithm by an oracle (which is the environment itself).

Two different exploration strategies are studied: a random exploration, in which their is no guaranty to visit all the blocks, or a scripted path, in which all the blocks are knocked at least one time. In both cases, the Adaptation Mechanism uses the data observed during this step to construct and update its model.

In a second step, an Exploitation phase allows the Exploitation mechanism to use the built model to maximize the intensity of the volume. This does not prevent the Adaptation Mechanism from continuing to enrich itself by observing the environment. This second phase is then similar to a reinforcement learning approach.

Thus, the experiment is a combination of supervised and reinforcement learning.

On the rest of this section, we present and discuss some results we obtained. More precisely, we put the focus on the evolution of the model built by AMOEBA, and the evolution of the global

4.4.1 Evolution of the model built by AMOEBA

The figures 7 and 8 show two-dimensional visualisations of the *Context Agents* paving on the dimensions space at the end of the exploration and exploitation phases for each experiment. This visualisation is a graphical representation of the different *Context Agents* and their associated utility level. Each rectangle represents a *Context Agent* and the sub-space in which this *Context Agent* is valid. Their colour depends on the utility level which is associated to them.

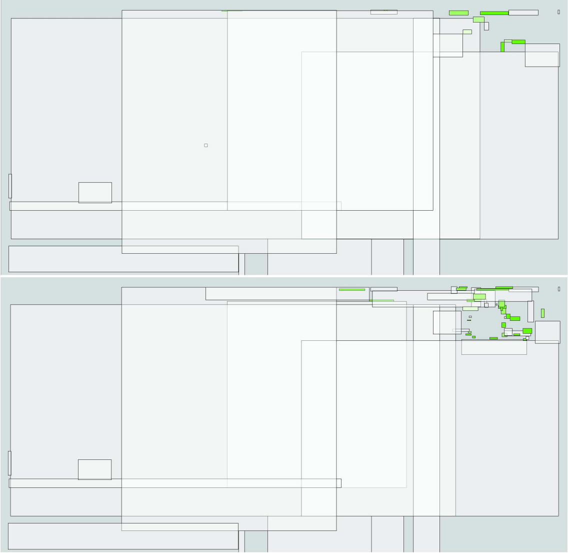

Figure 7: 2D visualisation of the *Context Agents* paving after a random exploration phase and after an exploitation phase.

The white colour means that the utility level is zero. The green shows a utility level superior to zero. The more the green is intense, the more the utility level is high. The grey areas without contours are areas that the toy has not yet explored, so the model has not any information about those areas, which explains the absence of *Context Agent*.

The upper figure 7 illustrates the model built by AMOEBA after a random exploration of the environment. By comparing this projection with the location of the obstacles in the arena, we observe that only the top right corner is populated with green squares. This implies that only the yellow block has been discovered during the exploration phase.

On contrary, the upper figure 8, which shows the model built at the end of a scripted exploration phase, shows three distinct areas populated by green squares, which corresponds to the blue and yellow blocks, and the upper wall of the arena. Those two figures illustrate how AMOEBA has populated its model with Context Agents. This shows that the model built by AMOEBA is dependant of the situation that it has experimented.

If we observe the evolution of the same model at the end of the exploitation phases (lower parts of figure 8 and 7), we can see a significant evolution of the distribution of context agents in the areas of contact with obstacles. In both cases, better details of the edges of the obstacles are obtained. In the case of random exploration, it is found that only one obstacle was actually used, but all of its edges were observed. Conversely, only a few edges of several obstacles were exploited following the random exploration.

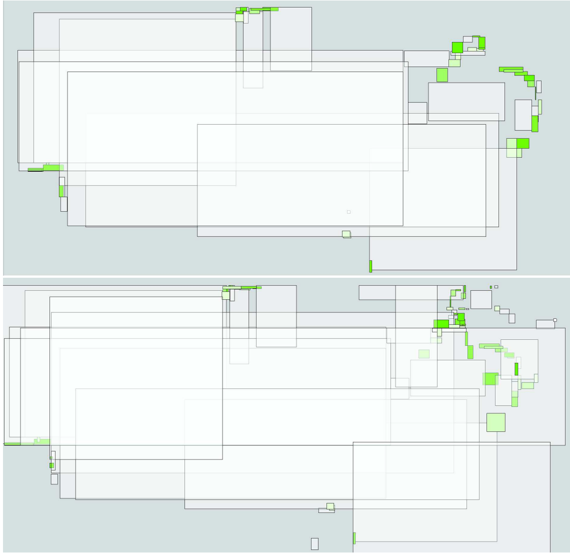

Figure 8: 2D visualisation of the *Context Agents* paving after a scripted exploration phase and after an exploitation phase.

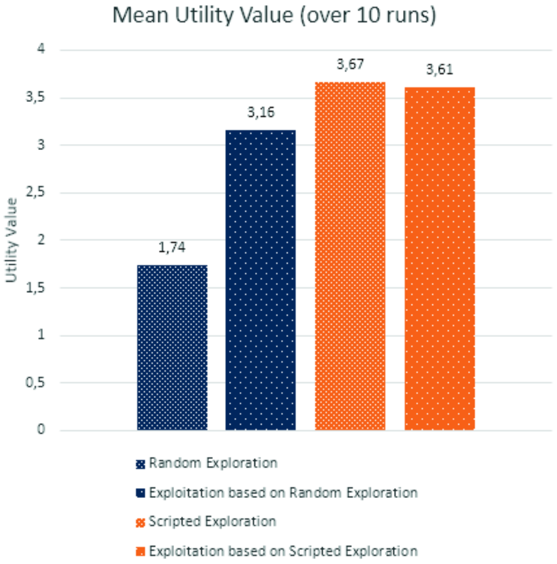

Figure 9: Comparison of the utility value between exploration and exploitation phases.

4.4.2 Evolution of Global Utility Value

In order to evaluate this SACL implementation, we propose to compare the global utility level generated during the phase of exploration with the utility level generated during the phase of exploitation. Each *exploration phase* and *exploitation phase* lasts three

minutes for the two experiments, and the experiment is computed ten time each.
The figure 9 shows the mean utility value obtained in each phase for the two different exploration strategies. The mean utility value during the random exploration is 1.74 and grows to 3.16 during the exploitation phase. During the scripted exploration phase, the mean utility value goes from 3.67 to 3.61 during the exploitation phase
In the case of random exploration, there is significant progress in average utility during the exploitation phase. Conversely, utility decreases (very slightly) during the exploitation phase linked to scripted exploration. This can be explained by the fact that the scenario is already relatively efficient. Nevertheless, this remains an interesting result because the system has learned to produce behavior that is almost as effective as an ad hoc scenario.

5 CONCLUSIONS AND PERSPECTIVES

The evolution of technologies now enables to consider that artificial systems will face more and more complex and dynamic environments where they will have to perform more and more various tasks. As the variety of environments and tasks is increasing, these systems needs to constantly adapt their behavior in order to maintain the usefulness of their interactions with their environment.
This requirement for constantly learning from the interaction with the environment will be a key component of Ambient Systems, notably because of their socio-technological aspect. Indeed, the notion of task in Ambient Systems is ambiguous and depends of its users, which makes them interact with human users. The incapacity to specify a priori all the interactions that can occur in these systems, combined with the high dynamics of those types of environment impeach an *ad hoc* design. On contrary, those artificial systems must constantly learn, through their own experiences, to interact with their environment.
In this paper, we present our use of the SACL pattern to design artificial systems with Lifelong Learning capacities. It proposes to design artificial systems in which a model is dynamically built by experience. This model is both exploited and enriched by the mechanism that uses it to behave.
This experiment illustrates how a model can be both built and exploited in real-time. The simulation we performed also shows that our approach is suitable for both supervised and reinforcement learning approaches.
The work introduced in this paper is currently being deployed in the neOCampus initiative which intends to transform the University of Toulouse into a smart lab. This deployment will allow real use-cases applications and comparative analysis in order to evaluate the benefits of our approach.

ACKNOWLEDGEMENTS

This work is partially funded by the Midi-Pyrenees region, in the neOCampus initiative (www.irit.fr/neocampus/) and supported by the University of Toulouse.

REFERENCES

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483.

Billard, A. (2003). Robota: Clever toy and educational tool. *Robotics and Autonomous Systems*, 42(3):259–269.

Boes, J., Migeon, F., Glize, P., and Salvy, E. (2014). Model-free Optimization of an Engine Control Unit thanks to Self-Adaptive Multi-Agent Systems (regular paper). In *International Conference on Embedded Real Time Software and Systems (ERTS2), Toulouse, 05/02/2014-07/02/2014*, pages 350–359. SIA/3AF/SEE.

Boes, J., Nigon, J., Verstaevel, N., Gleizes, M.-P., and Migeon, F. (2015). The self-adaptive context learning pattern: Overview and proposal. In *Modeling and Using Context*, pages 91–104. Springer.

Georgé, J.-P., Gleizes, M.-P., and Camps, V. (2011). Cooperation. In Di Marzo Serugendo, G., Gleizes, M.-P., and Karageogos, A., editors, *Self-organising Software*, Natural Computing Series, pages 7–32. Springer Berlin Heidelberg.

Guivarch, V., Camps, V., and Pninou, A. (2012). AMADEUS: an adaptive multi-agent system to learn a user's recurring actions in ambient systems. *Advances in Distributed Computing and Artificial Intelligence Journal, Special Issue n3*, Special Issue n3(ISSN: 2255-2863):(electronic medium).

Huang, C.-M. (2010). Joint attention in human-robot interaction. *Association for the Advancement of Artificial Intelligence*.

Jazdi, N. (2014). Cyber physical systems in the context of industry 4.0. In *Automation, Quality and Testing, Robotics, 2014 IEEE International Conference on*, pages 1–4. IEEE.

Knox, W. B. and Stone, P. (2009). Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the Fifth International Conference on Knowledge Capture*, pages 9–16. ACM.

Mitchell, T. M. (2006). *The discipline of machine learning*, volume 9. Carnegie Mellon University, School of Computer Science, Machine Learning Department.

Nehaniv, C. L. and Dautenhahn, K. (2002). The correspondence problem. *Imitation in animals and artifacts*, 41.

Niewiadomski, R., Hofmann, J., Urbain, J., Platt, T., Wagner, J., Piot, B., Cakmak, H., Pammi, S., Baur, T., Dupont, S., et al. (2013). Laugh-aware virtual agent and its impact on user amusement. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 619–626. International Foundation for Autonomous Agents and Multiagent Systems.

Nigon, J., Gleizes, M.-P., and Migeon, F. (2016a). Self-adaptive model generation for ambient systems. *Procedia Computer Science*, 83:675–679.

Nigon, J., Glize, E., Dupas, D., Crasnier, F., and Boes, J. (2016b). Use cases of pervasive artificial intelligence for smart cities challenges. In *IEEE Workshop on Smart and Sustainable City, Toulouse, juillet*.

Pentina, A. and Lampert, C. H. (2015). Lifelong learning with non-iid tasks. In *Advances in Neural Information Processing Systems*, pages 1540–1548.

Piaget, J. (1962). *Play, dreams and imitation in childhood*. New York : Norton.

Rifkin, J. (2016). How the third industrial revolution will create a green economy. *New Perspectives Quarterly*, 33(1):6–10.

Robins, B. and Dautenhahn, K. (2007). Encouraging social interaction skills in children with autism playing with robots. *Enfance*, 59(1):72–81.

Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., and Edwards, D. D. (2003). *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River.

Serugendo, G. D. M., Gleizes, M.-P., and Karageorgos, A. (2011). Self-organising systems. In *Self-organising Software*, pages 7–32. Springer.

Silver, D. L., Yang, Q., and Li, L. (2013). Lifelong machine learning systems: Beyond learning algorithms. In *AAAI Spring Symposium: Lifelong Machine Learning*, pages 49–55. Citeseer.

Thorisson, K. R., Bieger, J., and Thorarensen, T. (2016). Why artificial intelligence needs a task theory. In *Artificial General Intelligence: 9th International Conference, AGI 2016, New York, NY, USA, July 16-19, 2016, Proceedings*, volume 9782, page 118. Springer.

Thrun, S. and Mitchell, T. M. (1995). Lifelong robot learning. In *The biology and technology of intelligent autonomous agents*, pages 165–196. Springer.

Verstaevel, N. (2016). *Self-Organization of Robotic Devices Through Demonstrations*. Doctoral thesis, Universit de Toulouse, Toulouse, France.

Verstaevel, N., Régis, C., Gleizes, M.-P., and Robert, F. (2016). Principles and experimentations of self-organizing embedded agents allowing learning from demonstration in ambient robotics. *Future Generation Computer Systems*.

Zhang, B.-T. (2014). Ontogenesis of agency in machines: A multidisciplinary review. In *AAAI 2014 Fall Symposium on The Nature of Humans and Machines: A Multidisciplinary Discourse*.