Martínez Plumed, F.; Ferri Ramírez, C.; Hernández Orallo, J.; Ramírez Quintana, MJ. (2015). Knowledge acquisition with forgetting: an incremental and developmental setting. Adaptive Behavior. 23(5):283-299. doi:10.1177/1059712315608675.

Knowledge acquisition with forgetting: an incremental and developmental setting

Fernando Martínez-Plumed, Cèsar Ferri, José Hernández-Orallo, María José Ramírez-Quintana
DSIC, Universitat Politècnica de València, Camí de Vera s/n, 46022 València, Spain.
E-mails: {fmartinez,cferri,jorallo,mramirez}@dsic.upv.es

Abstract

Identifying the balance between remembering and forgetting is the key to abstraction in the human brain and, therefore, the creation of memories and knowledge. We present an incremental, lifelong view of knowledge acquisition which tries to improve task after task by determining what to keep, consolidate and forget, overcoming *the stability-plasticity* dilemma. Our framework can combine any rule-based inductive engine (which learns new rules) with a deductive engine (which derives a coverage graph for all rules) and integrates them into a lifelong learner. We rate rules by introducing several metrics through the first adaptation, to our knowledge, of the Minimum Message Length (MML) principle to a *coverage graph*, a hierarchical assessment structure which handles evidence and rules in a unified way. The metrics are used to forget some of the worst rules and also to consolidate those selected rules that are promoted to the knowledge base. This mechanism is also mirrored by a demotion system. We evaluate the framework with a series of tasks in a chess rule learning domain.

Keywords: Memory, forgetting, consolidation, knowledge acquisition, declarative learning, MML, lifelong machine learning.

1 Introduction

Memory in artificial systems is usually understood as an *encode-store-recall* structure where learned knowledge is placed, remaining static until recall. However, memory (as storage) should be understood as an active cognitive component (Wood et al., 2011), where the process of knowledge acquisition (automated process of abstracting knowledge from facts and other knowledge) cannot be understood as a naive accumulation of what is being learned. New knowledge must be checked to see whether it is redundant, irrelevant or inconsistent with old one, and whether it may be built upon previously learned knowledge. We argue that artificial intelligent systems should be developed for this purpose. This leads us to *the stability-plasticity* dilemma (Carpenter and Grossberg, 1988). The basic idea is that an adaptive cognitive system must be capable of learning new things (plasticity) without losing previously learned concepts (stability). This has been a designing principle within the perspective of neural computation over the last thirty years. Some of the proposed solutions

include: (a) dual-memory systems simulating the presence of short and long-term memory (French, 1997; Ans and Rousset, 1997), and (b) cognitive architectures such as the *Adaptive Resonance Theory* (ART, Grossberg 2013) emulating how the brain processes information. In both cases, those approaches can only incorporate new knowledge, without the ability of re-organisation or forgetting.

From our point of view, "truly" cognitive systems that are able to acquire new knowledge have to move towards more intelligent behaviours, thus being able to (a) support incrementally knowledge acquisition without the need to have (one-shot) models discarded and retrained repeatedly (which is not cost-effective), (b) integrate inductive and deductive reasoning algorithms for such a goal and guided by knowledge evaluation metrics (thus having the knowledge integrated in the system rather than being an adjunct storage system), and, finally, (c) focus on relevant knowledge (or discard what is not) by the use of cognitive mechanisms that simplify the learning of new knowledge. These principles, being the starting point for our knowledge acquisition approach, aim at generating adaptive behaviour in intelligent learning systems based on previously acquired knowledge. Following these requirements, below we overview some prior work in the area of knowledge acquisition.

Over the last decades, there has been an extensive work on growing knowledge bases from learned patterns and rules, in areas such as expert systems, machine learning, cognitive science, nonmonotonic logics, information systems and inductive (logic) programming. For instance, *Lifelong Machine Learning* (LML) (Thrun, 1996) is concerned with the persistent and cumulative nature of learning, namely, to be capable of (a) retaining and using prior knowledge, and (b) acquiring new knowledge over a series of prediction tasks. ELLA (Efficient Lifelong Learning Algorithm, Eaton and Ruvolo 2013) and NELL (Never-Ending Language Learner, Carlson et al. 2010) are two more recent approaches to LML, which are able to integrate many capabilities. However, from these works it is not easy to export or derive general principles to analyse a knowledge base and help in a general incremental knowledge acquisition process. A more profound knowledge management takes place with *concept drift* and *theory revision* (Rendell, 1995; Gama et al., 2014; Bragaglia and Ray, 2014), where some rules are replaced by new rules that are consistent with new experience. The areas of inductive logic programming (ILP) (Muggleton and De Raedt, 1994) or general inductive programming (IP) (Kitzelmann, 2010) have seen several approaches for incremental (Ferri-Ramírez et al., 2001) or cumulative systems (Henderson, 2014).

A crucial aspect relies on *theory and knowledge evaluation*. When the theory or hypothesis is considered as a whole and separated from the evidence, we have many well-founded proposals, such as the MML principle (Wallace and Boulton, 1968; Wallace, 2005) or the similar (but posterior) MDL principle (Rissanen, 1999). However, for knowledge integration and consolidation it is necessary to assess each part of the theory independently since different parts of the theory can have different degrees of validity, probability or reinforcement (Hernández-Orallo, 2000; Hernández-Orallo and García-Varea, 2000). But even in this case, there is still a separation between knowledge and evidence. It would be meaningful to fully integrate knowledge and evidence into a hierarchical assessment structure (from specific facts to more abstract rules). The perspective of a network or hierarchy of nodes that get support from other nodes is more common in the area of link analysis in web graphs such as the HITS algorithm (Kleinberg, 1999), PageRank (Brin and Page, 1998) or SALSA (Lempel and Moran, 2000), or in infometrics.

1.1 Forgetting

Knowledge acquisition and machine learning have much to learn from the study of human cognition and behaviour (Raducanu and Vitri, 2008; Erbas et al., 2014; Hourdakis and Trahanias, 2012; Leu et al., 2014). Cognitive factors, responsible for generating intelligent and adaptive behaviour, need to receive full consideration in order to improve current AI systems (not intelligent in human terms). In particular, there is a characteristic feature of intelligence that is essential for knowledge development: *forgetting*. Remembering absolutely everything prevents from having abstract thought (the process of generalisation). Forgetting can refer to a complete and irreversible elimination of significant old knowledge while learning new one; or it can denote that new learned knowledge is not always kept in the working memory but abstractly encoded by identifying their relation to abstract concepts already present in the knowledge base. This latter definition is the desired one: forgetting should exist in knowledge bases and learning systems to avoid possible information overflow and redundancy, and in order to preserve and strengthen important or frequently used rules and remove (or forget) useless ones.

The ability to focus on what to discard what is not relevant is becoming more relevant not only in cognitive science and neuroscience (Quiroga, 2012), but also in artificial intelligence (e.g., reasoning, planning, decision making). Usually considered in biology to be a combination of *decay* (to a lesser extent) and a proactive and retroactive *interference* (to a major extent) (Wixted, 2004), forgetting has been frequently used in AI systems because of performance and space constraints (Nuxoll et al., 2010; Alnajjar et al., 2009). Also known as *variable elimination*, forgetting has been widely investigated in the context of classical logic (Lin and Reiter, 1994; Lang and Liberatore, 2003; Lang and Marquis, 2010) and developed under the notion of logical equivalence. A similar approach but for reasoning from inconsistent propositional bases is proposed in (Lang and Marquis, 2010). Recently, the concept of forgetting has been widespread in other non-classical logic systems such as in logic programs (Zhang and Foo, 2006; Eiter and Wang, 2008; Wang et al., 2012) where a semantic forgetting is used developing a number of criteria for forgetting atoms; in modal logic (Zhang and Zhou, 2009; Su et al., 2009; Liu and Wen, 2011), in description logics (DLs) (Wang et al., 2010) and in planning (Erdem and Ferraris, 2007). Note that the forgetting mechanisms used in the previous approaches are based either on decay or relevance-related measures rather than interference. This is due to the symbolic nature for representing the information and the encapsulation property of symbols, which makes a scalar comparison (i.e. determination of the degree of overlap between discrete components) difficult (Wood et al., 2011). We argue that this could be overcome by means of hierarchical assessment structures of knowledge (as commented before) where relations between different individuals (pieces of knowledge) may be better understood and measured.

Closely related with the above concept we found *memory consolidation*, namely, the neurological process of converting information from short-term memory into long-term memory. Some studies about episodic memory in humans (Shastri, 2001; Dere et al., 2008) claim that memory traces in the hippocampus are not permanent and are occasionally transferred to neocortical areas in the brain through a consolidation processes. Recent cognitive models of memory have given great importance to consolidation procedures (Della Sala, 2010).

1.2 Research objectives

The development of a new learning system for knowledge acquisition that is meant to be cumulative is not an easy task. In fact, this research started when developing our system gErl (Martínez-Plumed et al., 2013, 2014). We were looking on a proper foundation for detailed knowledge assessment metrics and criteria for forgetting. The need of making general principles that were available for our system but also for any other system motivated the current work. In this work we take a most general approach by considering that we start with an off-the-shelf *inductive engine* (e.g., a rule learning algorithm, an ILP system or an IP system) and an off-the-shelf *deductive engine* (e.g., a coverage checker, an automated deduction system or a declarative programming language) and, over them, we construct a lifelong knowledge acquisition system where the taxonomy of memory follows the division imposed on human memory which separates processes for short and long-term recall (Hebb, 1949). For this purpose, several issues have to be addressed:

1. The inductive engine can generate many possible hypotheses and patterns. Once brought to the working memory (short-term memory system) we require metrics to evaluate how these hypotheses behave and how they are related to previous knowledge. Also, at any time new evidence can be added to the working space.

2. As working memory and computational time are limited, we need a forgetting criterion to discard some rules which are considered irrelevant in terms of informativeness. Forgetting should, therefore, follow a *proactive interference* principle (rather than decay or relevance) where new information may be more likely to be forgotten because of already existing information covering or overlapping the former.

3. The deductive engine checks the coverage of each hypothesis independently, using the background or consolidated knowledge as auxiliary rules, but not other working rules. As a result, only when new knowledge is consolidated (long-term memory system) we can use it for new problems or for more difficult examples of the same problem. This means that deduction is "modulo the background knowledge".

4. The promotion of rules into consolidated knowledge must avoid unnecessarily large knowledge bases and the consolidation of rules that are useless, too preliminary or inconsistent. That is, rules must be promoted and demoted to keep a powerful, but still manageable knowledge base.

The idea of *coverage graph* is used as the basis for structuring knowledge (and, thus, their relations) and is delegated to the deductive engine. The generation of new rules is delegated to the inductive engine. The crucial part is the definition of appropriate metrics to guide the way knowledge develops. For this purpose, the MML principle is used as a sound theoretical ground for the metrics. Note that the use of both linking structures with complexity and compression metrics helps determine the degree of overlap between individuals. Therefore, the ability to remember (consolidation) can be disrupted by what has been previously learned (existing information).

The following section introduces the notion of coverage graph, our setting for a knowledge base, over which we introduce an adaptation of the MML principle and related metrics in

section 3. Section 4 deals with knowledge structuring, how rules are forgotten, promoted and demoted. Section 5 includes several experiments that illustrate how knowledge consolidation and forgetting works. Finally, section 6 closes the paper with the contributions and some future work.

2 Coverage graph

We consider that 'rules' (expressions that define relations or functions in a declarative way) are used for representing examples, hypotheses and background knowledge. Rules are denoted by lower case Greek letters where $class(\rho) = c$, $c \in C$ and C is the set of classes, such as $\{\text{false}, \text{true}\}$. The set of all possible rules is denoted by $\mathcal{R}$, where $W \subset \mathcal{R}$ is the working space or memory, and $K \subset \mathcal{R}$ is the background or consolidated knowledge base.

Rules are presented as vertexes or nodes V (as determined by the deductive engine) in a directed acyclic graph $G(V, A)$ we call *coverage graph* (which is the DAG representation of a specific working space) because the directed edges A represent the coverage relation between the different rules. A rule ρ_a is covered by another rule ρ_b if $(K \cup \rho_b) \models \rho_a$. The precise understanding of the semantic consequence operator $\models$ will depend on the rule representation language used and the deductive engine. Hence, given an edge $a = (\mu, \nu)$ (or $\mu \to \nu$), we say that ν is directly covered by μ using K[1]. The set of ancestors and successors of a node ν are defined as $anc(\nu) = \{\mu | \mu \to \nu\}$ and $suc(\mu) = \{\nu | \mu \to \nu\}$ (respectively). Also, we distinguish two subsets of nodes: *leaves*, nodes without successors ($|suc(\nu)| = 0$), where $leaves_c$ denotes the set of *leaves* of class c; and *roots*, nodes without ancestors ($|anc(\nu)| = 0$).

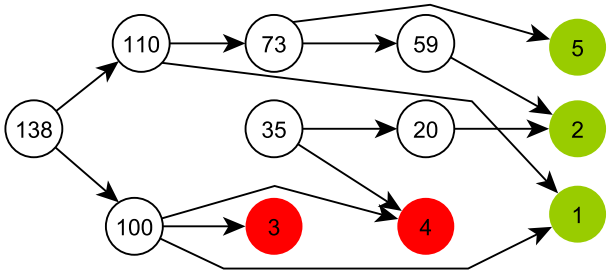


Figure 1: *Coverage Graph* of the *family relations* problem. Green and red nodes refer to positive and negative examples respectively. The graph shows rule IDs according to Table 1.

Figure 1 shows an example of *Coverage Graph* of a well-known ILP problem (Muggleton and De Raedt, 1994): the family relationship (rules in Table 1). In this problem, the task is to define the target relation $daughter(X, Y)$, which states that person X is daughter of person Y. W consists of three positive examples (rules 1, 2 and 5), two negative ones (rules 3 and 4), and seven selected rules that try to generalise and solve the problem (Table 1 right), whereas K is composed of the relations *female* and *parent* (Table 1 left). Note that the rules in K have not been included in the graph for clarity, although they belong to the initial "consolidated knowledge".

[1]For simplicity, the *coverage graphs* do not include the edges for the transitive closure of the covering relation, i.e., if a node μ covers nodes ν and γ, but ν also covers γ, only the edges $\mu \to \nu$ and $\nu \to \gamma$ are included in the graph.

Table 1: Left: Background Knowledge for the *family relations* problem. Right: Rules of this problem in Prolog notation.

Background Knowledge		**Rules**	
ID	Rule	ID	Rule
k1	parent(ann, mary).	1	daughter(mary,ann).
k2	parent(ann, tom).	2	daughter(eve,tom).
k3	parent(tom, eve).	3	daughter(tom,ann).
k4	parent(tom, ian).	4	daughter(eve,ann).
k5	female(ann).	5	daughter(cris,tom).
k6	female(mary).	100	daughter(X,Y):- female(Y),parent(Y,mary).
k7	female(eve).	59	daughter(eve,tom):- female(eve),parent(tom,eve).
		20	daughter(eve,tom):- female(eve).
		35	daughter(eve,Y):- female(eve).
		73	daughter(X,tom):- female(X),parent(tom,X).
		110	daughter(X,Y):- female(X),parent(Y,X).
		138	daughter(V,W):- female(X),parent(Y,Z).

3 Basic metrics for acquired knowledge assessment

In order to select and arrange the set of rules in the working space, various measures of usefulness, relevance and consistency have to be derived from the *coverage graph*. Based on the idea that the relevance or usefulness of a rule can be stated by the relationship between its own complexity and the complexity of the rules it covers, a general criterion such as the *Minimum Message Length* (Wallace and Boulton, 1968) (MML) can be used as a starting criterion from which to derive new metrics.

3.1 Minimum Message Length

The relationship between complexity, inference and compression underlies the Minimum Message Length (MML) principle. MML is one of the most popular selection criterion in inductive inference and provides an interpretation of the Occam's Razor principle: the model generating the shortest overall message is most likely. For a formal justification and its relation to Kolmogorov complexity and the related MDL principle, see Li and Vitányi (2008); Wallace and Dowe (1999); Wallace (2005). Using Shannon's information theory, MML restates the length of the message as a probability in a Bayesian interpretation (Wallace and Boulton (1968)) asserting that the best conclusion to draw from data is the theory with the highest posterior probability. We quantify this immediately below.

Given a hypothesis H and an observed learning problem E (evidence), we can express the posterior probability $P(H|E)$ by the application of Bayes's Theorem as:

$$P(H|E) = \frac{P(H) \cdot P(E|H)}{P(E)} \tag{1}$$

where $P(H)$ is the prior probability of model H, $P(E|H)$ is the likelihood, and $P(E)$ is the probability of evidence E. The information-theoretic interpretation of MML is that a given evidence E of probability $P(E)$ can be coded by a message of length $L(E) = -log_2 P(E)$ (Shannon, 1948). Therefore, taking the negative logarithm of the expression 1 and according to the MML philosophy, the length of a hypothesis H given a fixed evidence E, $L(H|E)$, is defined as the sum of three simple heuristics: a complexity-based heuristic which measures the complexity of H $(L(H))$, a coverage heuristic which measures how much

extra information is necessary to express the evidence given the hypothesis H ($L(E|H)$) and the length of the evidence ($L(E)$) which equal for all competing hypotheses:

$$L(H|E) = L(H) + L(E|H) - L(E) \tag{2}$$

By minimising equation 2 we maximise the posterior probability, which means searching for the model with the shortest message.

Apart from its connection with Kolmogorov complexity and Solomonoff induction (Li and Vitányi, 2008), which gives additional support for its use, the MML principle (and the similar MDL principle) has been successfully applied in many areas of machine learning, AI and cognitive science. However, to our knowledge, the MML principle has always been applied to select between hypotheses w.r.t. some given evidence. In our case, we have a coverage graph where rules cover other rules, so they become H and E at the same time. With this in mind , the MML principle can be adapted to be hierarchically applied: instead of measuring the length of a hypothesis H given fixed evidence E, we want to measure the length of each rule ρ in W with respect to the rest of rules in W (which includes examples and hypotheses) because ρ can model not only examples, but also other rules. Therefore, $L(\rho|W)$ is defined as the sum of the length of ρ ($L(\rho)$), and the length necessary to express the rules in $\{W - \rho\}$ not modelled by ρ ($L(W|\rho)$), minus the length of the total rules in W ($L(W)$). Formally:

$$L(\rho|W) = L(\rho) + L(W|\rho) - L(W) \tag{3}$$

Apparently, it just seems a notational change wrt. equation 2. This is only true for the first term, which is estimated in the same way as the original MML principle.

The term $L(\rho)$ can be defined in different ways depending on the rule representation language. For instance, if we are using logical or functional rules (as in the *family* example), we could use the following approximation. Given Σ a set of m_Σ functor symbols of arity ≥ 0 (functions or constants), and $\mathcal{X}$ a set of $m_\mathcal{X}$ variable symbols, we could define the length of a rule ρ containing n_Σ functors and $n_\mathcal{X}$ variables as (note that we promote variables over functors):

$$L(\rho) \triangleq m_\Sigma \log_2(n_\Sigma + 1) + \frac{m_\mathcal{X}}{2} \log_2(n_\mathcal{X} + 1) \tag{4}$$

Table 2 (columns $L(\rho)$ and *class*) shows the length in bits and the class for the rules in the graph in Figure 1.

3.2 MML goes hierarchical: Support

Following with equation 3, we are going to reunderstand the terms $L(W|\rho)$ and $L(W)$ to be adapted to *coverage graphs* and multiclass settings. Roughly speaking, these terms capture the "net profit" of the rules both in terms of support or coverage. More formally, we define the support of a rule $\rho \in W$ as:

$$S(\rho, W) \triangleq L(\rho) - L(\rho|W) = L(W) - L(W|\rho) \tag{5}$$

where $L(W) - L(W|\rho)$ represents the coverage of a rule ρ expressed in bits, that is, the length of all the rules in W minus the length of the rules not covered by ρ. Therefore, the support of a rule ρ represents the length of the rules it covers:

$$S(\rho, W) = \sum_{\nu : \rho \models \nu} L(\nu) \tag{6}$$

leading to a formulation of equation 3 in terms of support:

$$L(\rho|W) = -S(\rho, W) + L(\rho) \tag{7}$$

which establishes that maximising $S(\rho, W)$ and minimising $L(\rho)$ reduces $L(\rho|W)$, which means searching for the rule ρ that covers the maximum number of rules and has the lowest length.

The following step adapts equation 7 to be used in *coverage graphs* taking into account that the edges for the transitive closure of the coverage relation are not explicitly included.

In order to consider the upwards propagation, only the *leaves* will have an initial support value which is equal to its length in bits, and the rest of nodes will distribute it recursively by propagating this support. Thus, the new support ($S'(\rho, W)$) adapted to work on *coverage graphs* is defined as:

$$S'(\rho, W) \triangleq \begin{cases} L(\rho) & \text{if } \rho \in \mathit{leaves} \\ \sum\limits_{\nu \in suc(\rho)} S'(\nu, W) & \text{otherwise} \end{cases} \tag{8}$$

In order to avoid the scenario where the less grounded (upper) nodes get higher and higher support values, the support measure is required to satisfy a *conservative* condition. This property is somehow related to the law of conservation of energy, implying that at any node in a coverage graph, the sum of the total support flowing into that node is equal to the sum of the total support flowing out of that node.

Now, to make S' (equation 8) conservative we need to divide the support coming from the outcoming of a specific node ν by $|anc(\nu)|$ in order to equally distribute the support of ν between all of its ancestors.

Therefore, the new formula used to calculate the support of a rule ($\dot{S}(\rho, W)$) is defined as:

$$\dot{S}(\rho, W) \triangleq \begin{cases} L(\rho) & \text{if } \rho \in \mathit{leaves} \\ \sum\limits_{\nu \in suc(\rho)} \frac{\dot{S}(\nu, W)}{|anc(\nu)|} & \text{otherwise} \end{cases} \tag{9}$$

and re-formulating equation 7 in terms of this conservative support:

$$\dot{L}(\rho|W) = -\dot{S}(\rho, W) + L(\rho) \tag{10}$$

Equation 9 now accomplishes the mandatory conservative condition that the support of a node (which depends on its successors) has to be always entirely allocated in its ancestors together with the support inherited from other covered nodes.

This implies (but not vice versa) that the total sum of the support in the *leaves* in the *coverage graph* is equal to the total sum of the support at the *roots*:

$$\sum_{\mu \in leaves} \dot{S}(\mu, W) = \sum_{\nu \in roots} \dot{S}(\nu, W) \tag{11}$$

For each *leaf* in the *coverage graph* we have n different paths whereby the support flows upwards to *root* nodes. Whenever a path is forked (an ancestor is found), the support is always divided by the number of the outcoming paths, having the ancestors an equally part of the support and thus having the roots a proportion of the original support of the leaves transitively covered by them. Therefore, if we assume that the total support at the roots is different from the total support at the *leaves*, it means that an external transfer of support (which comes from or goes to other sources) has happened. However, accordingly to equation 9, this is not possible and, therefore, the total sum of the support at the *roots* always remains constant and equal to the total support at the *leaf* nodes (see Figure 2).

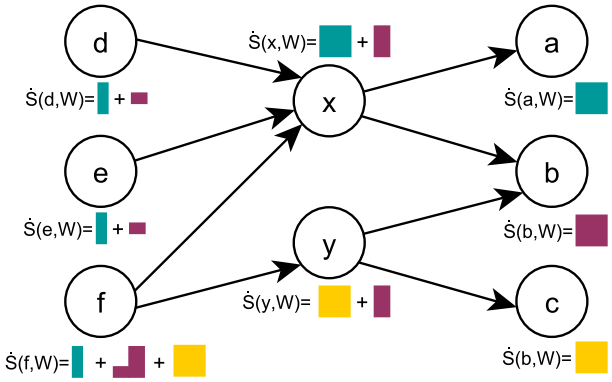


Figure 2: Graphical representation of the flow of the conservative support (by using equation 9) in a *coverage graph*: the support of each *leaf* node is always allocated in the *roots*.

Finally, we need to take into account that, since the working space W can accommodate examples of different classes which have to be handled equally (e.g., positive and negative classes as in the *family* problem in Table 1), we need our metric to distinguish between them. Hence, there are as many support values for each node as many different classes there are in the working space, each one holding the *conservative* property and formally defined as:

$$\dot{S}_c(\rho, W) \triangleq \begin{cases} L(\rho), & \text{if } \rho \in leaves_c \\ \sum\limits_{\nu \in suc(\rho)} \frac{\dot{S}_c(\nu, W)}{|anc(\nu)|} & \text{otherwise} \end{cases} \tag{12}$$

with equation 10 being defined for classes as follow:

$$-\dot{L}_c(\rho|W) = \dot{S}_c(\rho, W) - L(\rho) \tag{13}$$

The value of $\dot{L}_c$ is interpreted as the hierarchical version of the MML principle, with the sign change now $-\dot{L}_c$ the higher the better. Note that, as a leaf (ground example) does not cover any other rule, for any $\rho \in leaves_c$ it holds $-\dot{L}_c(\rho|W) = 0$ (because $\dot{S}_c(\rho, W) = L(\rho)$) and $-\dot{L}_{c'}(\rho|W) = -L(\rho), \forall c' \in C, c' \neq c$ (because $\dot{S}_{c'}(\rho, W) = 0$ since ρ has no successors).

Following with the *family* example, Table 2 (columns $\dot{S}_+$, $\dot{S}_-$, $-\dot{L}_+$ and $-\dot{L}_-$) shows, respectively, the support and the negative form of $L(\rho|W)$ per class (positive and negative) of the rules in the graph in Figure 1.

3.3 Optimality

By using the support (equation 12) as the sole criterion to rank the rules in W is useful provided there are only rules belonging to one class. However, when there are more than one class in W, we need to consider the purity or confidence of the rules. In the same spirit of the MML principle, we define the optimality as the difference between the cost of coding a rule following equation 13 for a specific class and the cost of coding the exceptions, i.e., the support of the rules covered that belong to the other classes. We use a factor β indicating the relevance of rules being as pure as possible. Formally:

$$opt_c(\rho, W) \triangleq -\beta \cdot \dot{L}_c(\rho|W) - (1-\beta) \cdot \sum_{\substack{c' \in C \\ c' \neq c}} \dot{S}_{c'}(\rho, W) \tag{14}$$

leading to a *generic* optimality of a rule as:

$$opt(\rho, W) \triangleq \max_{c \in C}(opt_c(\rho, W)) \tag{15}$$

Following with the *family* example, Table 2 (columns opt_+ and opt_-) shows the optimality values per class (the generic optimality in bold) for the rules in the graph of Figure 1 using $\beta = 0.5$. According to these values, rule 110 is the most significant rule, as it can be easily viewed in the *coverage graph* because it covers all the positive examples and no negative one. Notice that the examples (rule IDs from 1 to 5) have zero values for the optimality for the class they belong to because their support is equal to their length (equation 12) and no exceptions are covered. The intuitive idea behind is that the optimality of the examples should be the baseline level to ensure the usefulness of the rest of rules, i.e., those rules with a *generic* optimality value less than zero mean that they are not worth keeping in the working space (are less useful than the examples), while those rules with an optimality value greater than 0 mean that, at least, they are more significant than a single example.

4 Structuring knowledge: forgetting, promotion and demotion

In our setting, rules are repeatedly generated by the inductive engine and added to the working space W. In order to prevent the possible never-ending growth of W, we need some mechanisms to discard those rules that are not useful, are inconsistent or do not get enough support.

4.1 Forgetting mechanism

The optimality of a rule ρ is a core metric to determine its usefulness, but it is also important to see whether ρ could be considered superfluous because it is covered (transitive or directly)

Table 2: Metrics derived from the *coverage graph* (Figure 1) of the rules on the right side of Table 1. The columns represent (from left to right) for each rule: its length, class, support, $-\dot{L}(\rho|W)$ (equation 13), Optimality (where bold values indicate the *generic* optimality as for equation 15) and permanence. By considering the support values, we can only establish a ranking per classes. However, ranking the rules by optimality, we can decide that the best rule is 110. Furthermore, given the permanence values, we see that rule number 59 is redundant (Figure 1) because it is covered by a more significant rule (with ID 110), and it has the lowest value of permanence.

ID	$L(\rho)$	*class*	$\dot{S}_+$	$\dot{S}_-$	$-\dot{L}_+$	$-\dot{L}_-$	opt_+	opt_-	*perm*
1	17.844	+	17.844	0.0	0.0	-17.844	**0.0**	-17.844	-12.863
2	17.844	+	17.844	0.0	0.0	-17.844	**0.0**	-17.844	-12.863
3	17.844	−	0.0	17.844	-17.844	0.0	-17.844	**0.0**	-2.933
4	17.844	−	0.0	17.844	-17.844	0.0	-17.844	**0.0**	-2.933
5	17.844	+	17.844	0.0	0.0		0.0	-17.844	-12.863
100	11.977		8.922	26.766	-3.0549	14.788	-14.91	**2.933**	2.933
59	18.791		8.922	0.0	-11.114	-20.036	**-5.557**	-14.479	-14.479
20	11.591		8.922	0.0	-2.668	-11.591	**-1.334**	-10.256	-1.334
35	9.284		8.922	8.922	-0.362	-0.362	**-4.642**	**-4.642**	-4.642
73	13.114		26.766	0.0	13.651	-13.114	**6.825**	-19.939	-6.037
110	9.962		35.688	0.0	25.726	-9.962	**12.863**	-22.825	10.173
138	12.462		44.61	26.766	32.147	14.303	**2.69**	-15.153	2.69

by another rule of higher optimality. If it is the case, ρ is mostly redundant and it could be discarded safely. This idea leads to the following definitions for the permanence of a rule:

$$perm_c(\rho, W) \triangleq opt_c(\rho) - \max(0, \max_{\nu:\nu\models\rho} opt_c(\nu)) \tag{16}$$

and its *generic* permanence:

$$perm(\rho, W) \triangleq \max_{c\in C}(perm_c(\rho, W)) \tag{17}$$

The lower the value of permanence a rule has, the higher the odds it has to be forgotten. Following with the *family* example, Table 2 (column *perm*) shows the permanence values per each rule. We see that rule number 59 is the rule with lower permanence and, thus, candidate to be forgotten: it is redundant (see Figure 1) because is covered by a more significant rule (id 110).

When we perform a forgetting step, the coverage graph is affected and coverages are also affected. In order to keep as much information about the past support, each rule is provided with a trace of its old support. In cognitive systems this is associated to notions such as the preservation of belief and trust even if we forget the particular cases that gave support to a given statement. Thus, the forgetting mechanism works as follows:

1. If a non-*leaf* node is selected to be forgotten, the support of its successors has to be re-distributed among their ancestors and the ancestors of the forgotten node.

2. In case there is a forgetting step that removes a leaf node, its support has to be equally distributed among the rules that cover it which inherit it as their "residual" support value associated to each class c (res_c).

Hence, the equation 12 is modified to include the residual:

$$\mathring{S}_c(\rho|W) \triangleq \begin{cases} L(\rho) & \text{if } \rho \in leaves_c \\ res_c + \sum_{\nu\in suc(\rho)} \frac{\mathring{S}_c(\nu,W)}{|anc(\nu)|} & \text{otherwise} \end{cases} \tag{18}$$

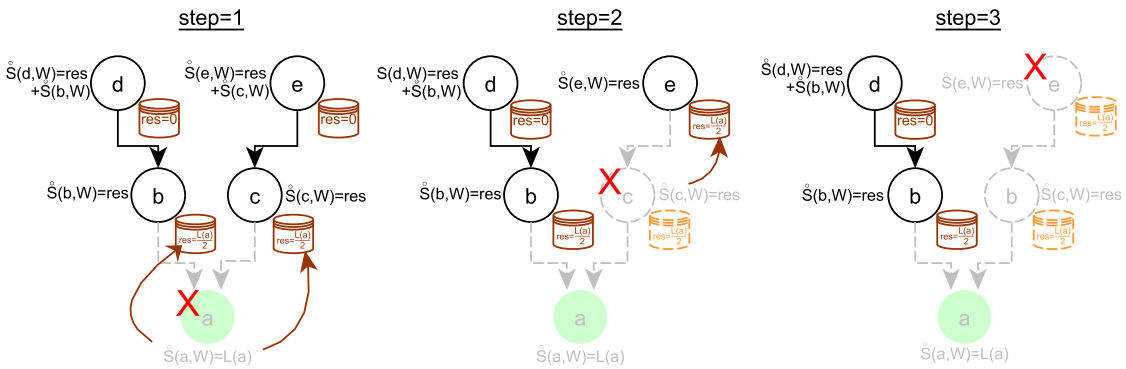


Figure 3: Forgetting mechanism performed over a complete branch. Steps 1 and 2 show how the support from the forgotten nodes (those which blurred colours and marked with a red cross) is distributed (pointing-up arrows) among the outcoming nodes increasing their res_c value. Step 3 shows that when the last forgetting step removes a node without ancestor nor successors and a non-zero res_c, this value cannot be distributed and, therefore, is lost. Therefore, the conservative property over the support measure occurs in all steps, but the initial amount of support at $step = 0$ ($\sum_{\mu \in leaves} \mathring{S}(\mu, W) = L(a)$) has been reduced by half at the $step = 3$ ($\sum_{\mu \in leaves} \mathring{S}(\mu, W) = \frac{L(a)}{2}$).

where res_c is initially set as 0. For each forgetting step, the support of forgotten nodes is distributed among their outcoming nodes increasing their res_c value, but if the last forgetting step removes a node without ancestor nor successors and a non-zero res_c, this value cannot be further distributed and, then, it is lost. This decrements the total support of the graph; although the support will remain conservative, the total amount will be lower than the total support of the *coverage graph* before the forgetting steps. Consequently, in the end some rules may have an under-estimated support value in terms of how many rules (of different classes) they cover (see Figure 3)[2].

4.2 Consolidated knowledge: promotion and demotion

Finally, some of the rules with good indicators in the working space have to be eventually promoted to consolidated knowledge (or *belief*). This has to be a careful process, as the consolidated knowledge will be used by the deductive engine to calculate coverage. This means that an inconsistent rule that is promoted to the consolidated knowledge may have important consequences on the behaviour of the system.

The promotion function can be tuned for the application, but a general choice is to use a threshold θ_p on the optimality.

When a rule is promoted to consolidated knowledge K, it cannot be target of the forgetting mechanism and, hence, be forgotten. It may happen that this rule can be eventually removed from the consolidated knowledge. Therefore, the promotion system is mirrored by a demotion system, with the use of another threshold θ_d. The original background knowledge (K_0) cannot be demoted (and forgotten).

In the *family* example (Table 1) in case we would establish θ_p equal to the average optimality of all the rules in the working space, all the rules that exceed this average value will be consolidated to the background knowledge base (rules 110 and 73).

[2]More details can be found in (Martínez-Plumed et al., 2015)[Figure 5 and Table 3] through nine consecutive forgetting steps with the *family* example.

5 Experiments

We claim that our approach is able to address the Stability-Plasticity dilemma (mentioned in section 1) in a lifelong or incremental learning process. For this purpose, we have conducted an experimental evaluation to explore the following questions: (a) is it possible to gradually generate a large repository of consolidated knowledge assessing the usefulness of the rules? (b) is our approach able to forget or revise the existing knowledge in order to generate a rich and reusable knowledge base? and (c) how are the process and the resulting knowledge structure understood in terms of cognitive systems that must acquire and develop knowledge incrementally? We want to illustrate these features in one single domain. The ultimate goal of these experiments is to see whether the framework is general enough to work with off-the-shelf inductive and deductive engines, to better understand how the metrics and procedures work, and finding whether they may require some tuning or improvement to the framework before addressing other problems.

5.1 Methodology

We will focus on learning a model of legal moves of different pieces of chess from a set of legal and illegal move examples (extracted from (Muggleton et al., 1989)). In our framework, the legal moves are the positive examples and the illegal moves the negative ones (so we have two classes). Each example representing a move is denoted by a triple from the domain $Piece \times Pos \times Pos$, where the second and third components represent, respectively, the piece's initial position and its destination on a chessboard. Positions are represented by a tuple from the domain $File \times Rank$ where files (a-h) stand for columns and ranks (1-8) stand for rows. For instance, Figure 4 illustrates all the possible moves of a knight from a specific initial position (k) to several other positions (k'). We will use a Prolog notation (as in the example in the previous section). The only background predicate used is the absolute difference, *diff(X,Y)*, that calculates the distance between X and Y, where both X and Y can be ranks or files (see Table 3).

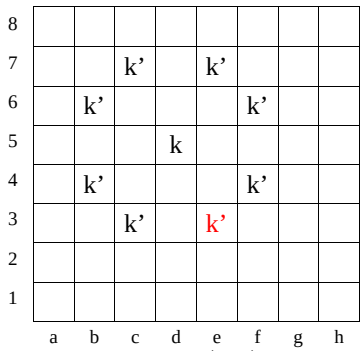


Figure 4: Possible moves of the knight from position (d,5). The particular legal move from k to k' will be represented as `move(knight,pos(d,5),pos(e,3))`.

The challenge we would like to face is knowledge acquisition in a progressive way from examples provided incrementally. A random set of chess moves from all chess pieces in the game except the pawn is given. This includes 28 positive and 12 negative examples. We also consider that an inductive engine (in our case the ILP system Progol (Muggleton, 1995))

Table 3: Background knowledge for the chess problem.

ID	ρ	ID	ρ
k1	project(a,1).		
k2	project(b,2).		rdiff(Rank1,Rank2,Diff):-
k3	project(c,3).	k11	rank(Rank1), rank(Rank2),
k4	project(d,4).		Diff1 is Rank1-Rank2, abs(Diff1,Diff).
k5	project(e,5).		
k6	project(f,6).		fdiff(File1,File2,Diff) :-
k7	project(g,7).		file(File1),file(File2),
k8	project(h,8).	k12	project(File1,Rank1),
k9	abs(X,X) :- X>=0.		project(File2,Rank2),
k10	abs(X,Y) :-,X<0, Y is -X.		Diff1 is Rank1-Rank2, abs(Diff1,Diff).

is generating rules during the whole process (60 in total), which arrive to the system in a random order. A geometric distribution determines how many examples and rules are given for each step of the system: the probability that n examples (and similarly for rules) are given is $P(X = n) = (1-p)^{n-1} \cdot p$ where n is $1, 2, 3, \ldots$ and p is the probability of success (we set it to 0.5). In order to better mimic a situation where the inductive engine can produce rules it has already generated, we use this distribution with replacement. Similarly, we have considered replacement for the examples. We have set the consolidation criterion with a threshold of optimality greater than the average of the optimality value of the rules in W (provided that it is above the average optimality of the evidence). Furthermore, since we want the consolidated knowledge to represent legal chess moves, we have set the β parameter equal to 0.1 (equation 14) for penalising those rules that are not pure.

5.2 Consolidation without forgetting

In a first experiment we try to show what would happen without applying the forgetting mechanism and check whether the MML-based measures work successfully for knowledge acquisition. Figure 5 shows the evolution of the learning process during 500 steps. As no rules are forgotten, the rule population (dashed brown line) reaches its maximum value (40 examples and 60 rules) and it stagnates ignoring any new evidence which arrives to the system (because they are already placed in W) from step 180 onwards. In this case we have assumed that all the evidence of the chess problem can be allocated in W, however it could be the case that all knowledge of a problem does not fit into W (due memory restrictions) thus collapsing with no improvement. The same applies to both the average optimality of all rules (dashed blue line) and the consolidated ones (dashed green line), since no more new rules are allocated into W, no further learning or knowledge improvement can take place.

Table 4 shows the consolidated rules at step 500[3] that almost represent all the legal chess moves (only two movements of the knight are missing) and there is only one rule ($x20$) which, despite representing a legal move, does not completely generalise the movement of the king piece. The conclusion is that the metrics provide a guarantee of promoting those rules that, having the maximum compression, best describe the problem.

[3]See (Martínez-Plumed et al., 2015)[Table 12] for all the rules in W at step 500.

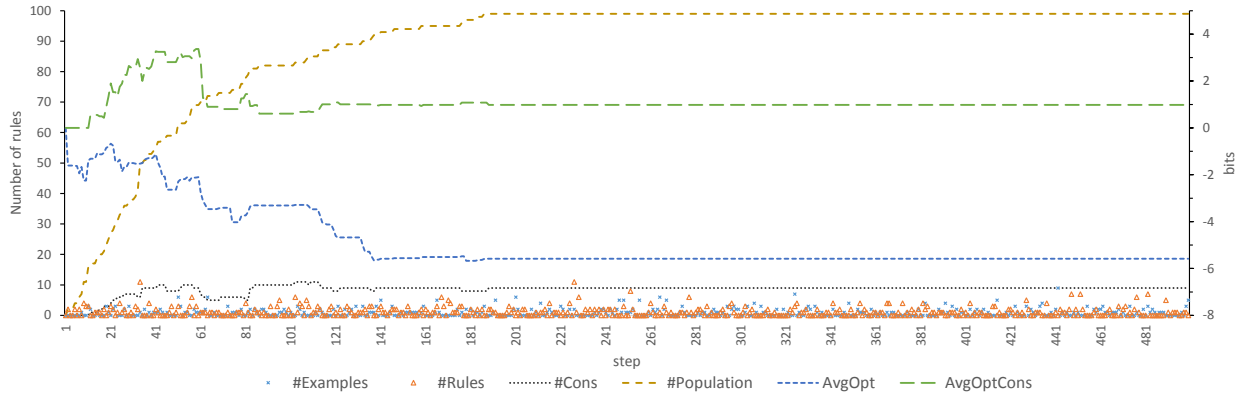


Figure 5: Evolution of some indicators for the chess problem *without* the forgetting mechanism: *#Examples* and *#Rules* show the examples that arrive and the rules that are generated by the inductive engine, #Cons is the number of consolidated rules (initially the background knowledge) and *#Population* is the total number of rules (magnitudes shown on the left y-axis). *AvgOpt* and *AvgOptCons* show, respectively, the average optimality for all rules and for all consolidated rules (magnitudes shown on the right y-axis). After the working space is filled with all the evidence and rules, the metrics become stable.

Table 4: Consolidated rules and metrics for the chess problem without the forgetting mechanism at step 500. IDs in bold represent those rules that perfectly generalise the legal moves of the chess pieces.

ID	ρ	$L(\rho)$	$\dot{S}_+$	$\dot{S}_-$	$-\dot{L}_+$	$-\dot{L}_-$	opt_+	opt_-	$Perm$
r15	move(rook,pos(A,B),pos(A,C)).	22.133	49.594	0	27.461	-22.133	2.746	-46.847	2.746
q19	move(queen,pos(A,B),pos(C,B)).	13.214	27.052	0	13.838	-13.214	1.383	-25.668	1.383
q12	move(queen,pos(A,C),pos(A,D)).	13.214	27.052	0	13.838	-13.214	1.383	-25.668	1.383
r16	move(rook,pos(A,B),pos(C,B)).	20.455	33.815	0	13.360	-20.455	1.336	-32.479	1.336
x18	move(king,pos(A,B),pos(C,D)) :- rdiff(B,D,1), fdiff(A,C,1).	34.275	40.578	0	6.303	-34.275	0.630	-39.947	0.630
x20	move(king,pos(A,B),pos(A,C)) :- rdiff(B,D,1).	22.918	27.052	0	4.134	-22.918	0.413	-26.638	0.413
x13	move(king,pos(A,B),pos(C,B)) :- fdiff(A,C,1).	22.918	27.052	0	4.134	-22.918	0.413	-26.638	0.413
q23	move(queen,pos(A,B),pos(C,D)) :- rdiff(B,D,E),fdiff(A,C,E).	28.993	32.462	0	3.469	-28.993	0.346	-32.115	0.346
b10	move(bishop,pos(A,B),pos(C,D)) :- rdiff(B,D,E), fdiff(A,C,E).	24.534	26.300	0	1.766	-24.534	0.176	-26.123	0.176

5.3 Consolidation with forgetting

Now, we repeat the same experiment, but using forgetting. This represents a situation where we have a more limited working space, so it is necessary to forget rules in order to allocate new ones. We want to show that if our approach is able to find a solution without forgetting, a suitable (and possibly better) solution should exist by using forgetting. In order to do that, we have executed several configurations varying the size of the working space ($|W| \in \{20, 30, 40, 50, 60, 70, 80, 90\}$). Also, every time the limit is exceeded the forgetting process is launched, forgetting up to 25%, 50% or 75% of the most meaningless rules. Each different configuration has been launched 10 times giving 240 executions in total.

Table 5 is a *heat map*, showing, for each possible configuration ($|W| \times forgetting(\%)$) how many times a specific rule appears in the consolidated knowledge in 10 repetitions, from white (0 times), light yellow (1 time) to dark green (10 times). Rules that are not represented in the heat map is because they have not been consolidated at any time. As a reference, the bottom row ($|W| = 100$) represents the consolidated rules by the first experiment (Table 4). We see that not only the set of consolidated rules almost always includes the reference

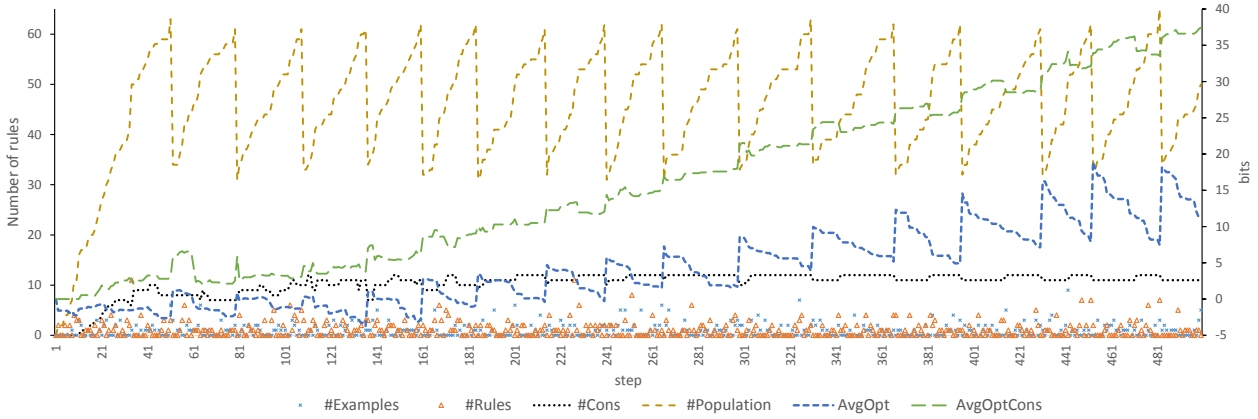


Figure 6: Evolution of the same indicators as in Figure 5 for the chess problem *with* the forgetting mechanism (for a configuration with $|W| = 60$ and up to 50% of rules forgotten). Now we see a bumpier picture, where the forgetting mechanism takes place every 30 steps approximately.

solution (even with very limited resources), but also the forgetting criterion allows the system to include those rules that perfectly generalise the moves of the king (rules in bold). The rest of rules included in the consolidated set in each experiment also generalise different movements of the pieces.

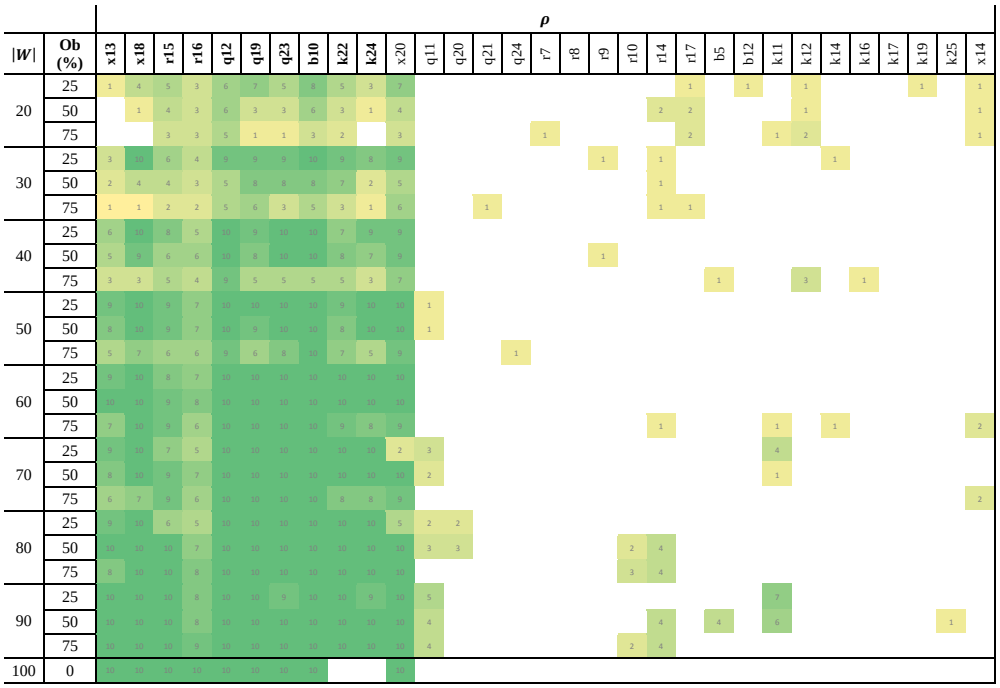
In order to compare both experiments, Figure 6 shows the evolution of the system during 500 steps for one of the 24 configurations ($|W| = 60$ and forgetting up to 50%). Now, the variations in the amount of consolidated rules (dotted black line) and rules in W (dashed brown line) allow us to observe how the forgetting mechanism works (every 30 steps approximately). Table 6 presents the consolidated rules at the final step (500)[4]. In this case, this set perfectly generalises all the legal moves of all the chess pieces. The system has reached a stable situation in which the number of consolidated rules (dotted black line) remains almost constant from step 250. The average optimality of both the consolidated rules (dashed green line) and all the rules (dashed blue line) have an increasing trend due to the distribution with replacement used to populate the working space. The appearance of new rules in the system or the execution of the forgetting mechanism mainly affect the average optimality of W (dashed blue line): every time it runs, the working space is cleaned of useless rules which strongly affects the metrics of the rules in W (and to a lesser extent to the consolidated set of rules (dashed green line)) that have to be recalculated. Compared with the former experiment, the number of rules in W has been reduced (with one order of magnitude (10x) speedup in execution) obtaining a better set of consolidated knowledge: it comprises all the rules that solve the chess problem, including the two moves of the knight, rules $k22$ and $k24$, missing in the first experiment.

5.4 Incremental knowledge acquisition

Finally, the last experiment shows the capability of our approach for the incremental learning of new knowledge from previously consolidated concepts. This experiment has two phases. In the first one we have only taken rules and examples of the rook and bishop moves (15 and

[4]See (Martínez-Plumed et al., 2015)[Table 13] for all the rules in W at step 500.

Table 5: Heat map showing the percentage of times a rule has been consolidated for each configuration ($|W|$=20..100, forgotten(%)=25%..75%) and 10 repetitions. Larger values are represented by dark green squares and smaller values by lighter yellow squares (those blank cells represent absence for this rule in all repetitions for a specific configuration). The last row shows the results without forgetting (as a reference). Rules in bold (ρ) belong to the problem solution.



30 rules respectively) providing the system with them in the same way as in the previous experiment. The consolidation criterion has not been changed, but the maximum number of rules in W has been established to 15 (in order to allow the forgetting mechanism to work) and the percentage of rules that are forgotten up to 25%, due to the smaller size of W. Table 7 shows the set of consolidated rules after 100 steps, which perfectly generalises all the legal moves of the rook and the bishop. In the first 100 steps of Figure 7 we see how the forgetting and consolidation mechanisms runs every few steps, due to the lower maximum number of rules in W, the lower percentage of forgotten rules and the geometric distribution used to provide the rules. We can also see a non-constant sawtooth-like wave ramps for the number of rules in W (dashed brown), whereas the number of consolidated rules remains almost constant from step 45 to 100.

In the second phase, we provided the system with a new set of rules and examples (10 and 20 respectively) only representing moves of the queen chess piece. At this point it should be possible to use the previously learned rook and bishop moves apart from using the background knowledge provided initially. Table 8 shows the set of consolidated rules which contains the previously consolidated rules that generalise the legal moves of the rook and bishop, and a new set of rules that represents the legal moves of the queen. This latter set includes rules $q29$ and $q25$ that use the rook and bishop rules and represent all the possible moves of the queen. The second half of Figure 7 (from step 100) shows how the forgetting mechanism runs even more frequently than previously (dashed brown) due to the increment of consolidated rules (that cannot be targeted by forgetting). Again, the number

Table 6: Consolidated rules and metrics (as in Table 4) for the chess problem *with* forgetting at step 500 (with $|W| = 60$ and forgetting up to 50%). IDs in bold represent rules that perfectly generalise chess legal moves.

ID	ρ	$L(\rho)$	$\dot{S}_+$	$\dot{S}_-$	$-\dot{L}_+$	$-\dot{L}_-$	opt_+	opt_-	$Perm$
x18	move(king,pos(A,B),pos(C,D)) :- rdiff(B,D,1), fdiff(A,C,1).	34.275	662.774	0	628.499	-34.275	62.849	-599.924	-22.681
x13	move(king,pos(A,B),pos(C,B)) :- fdiff(A,C,1).	22.918	459.884	0	436.966	-22.918	43.696	-416.187	-41.834
k22	move(knight,pos(A,B),pos(C,D)) :- rdiff(B,D,2), fdiff(A,C,1).	34.275	446.358	0	412.083	-34.275	41.208	-405.149	29.394
q23	move(queen,pos(A,B),pos(C,D)) :- rdiff(B,D,E),fdiff(A,C,E).	28.993	437.340	0	408.347	-28.993	40.834	-396.505	-1.513
x20	move(king,pos(A,B),pos(A,C)) :- rdiff(B,D,1).	22.918	392.254	0	369.336	-22.918	36.933	-355.320	8.116
r15	move(rook,pos(A,B),pos(A,C)).	22.133	389.999	0	367.866	-22.133	36.786	-353.212	6.602
q19	move(queen,pos(A,B),pos(C,B)).	13.214	365.202	0	351.988	-13.214	35.198	-330.003	-7.149
k24	move(knight,pos(A,B),pos(C,D)) :- rdiff(B,D,1), fdiff(A,C,2).	34.275	369.710	0	335.435	-34.275	33.543	-336.166	25.561
q12	move(queen,pos(A,C),pos(A,D)).	13.214	311.098	0	297.884	-13.214	29.788	-281.309	-12.559
r16	move(rook,pos(A,B),pos(C,B)).	20.455	284.046	0	263.591	-20.455	26.359	-257.686	-3.824
b10	move(bishop,pos(A,B),pos(C,D)) :- rdiff(B,D,E), fdiff(A,C,E).	24.534	275.028	0	250.494	-24.534	25.049	-249.978	12.731

of consolidated rules (dotted black line) remains constant most of the time (from steps 140 to 200).

Table 7: Consolidated rules and metrics (as in Table 4) for the chess problem (rook + bishop moves) at step 100. All rook and bishop legal moves are covered by these rules and no better rules can be obtained. IDs in bold represent rules that perfectly generalise the rook and bishop moves.

ID	ρ	$L(\rho)$	$\dot{S}_+$	$\dot{S}_-$	$-\dot{L}_+$	$-\dot{L}_-$	opt_+	opt_-	$Perm$
b10	move(bishop,pos(A,B),pos(C,D)) :- rdiff(B,D,E), diff(A,C,E)	24.534	590.635	0	566.101	-24.534	56.610	-534.024	24.904
r15	move(rook,pos(A,B),pos(A,C)).	22.133	277.282	0	255.149	-22.133	25.514	-251.767	25.514
r16	move(rook,pos(A,B),pos(C,B)).	20.455	223.178	0	202.723	-20.455	20.272	-202.905	20.272
r7	move(rook,pos(A,2),pos(B,2)).	19.718	175.838	0	156.120	-19.718	15.612	-160.226	-4.659
r14	move(rook,pos(A,2),pos(C,D)).	21.133	162.311	0	141.178	-21.133	14.117	-148.193	14.117
r9	move(rook,pos(a,B),pos(h,B)).	19.133	121.733	0	102.600	-19.133	10.260	-111.473	-10.011

5.5 Discussion

The chess domain has been an appropriate starting point for illustrating and evaluating our approach. Firstly, it is a well-known and well-structured domain where we can test our metrics and procedures. Furthermore, it involves a lifelong or incremental learning process and thus it can be used as a test bed for addressing the stability-plasticity dilemma. Finally, due to its intrinsic hierarchical nature (in terms of coverage between rules), the coverage function is well defined.

Our ultimate goal with the experiments was to provide some insight into the generality, efficiency and usefulness of the forgetting and consolidation cognitive procedures: we show that by using a proactive interference-based approach we allow knowledge to be forgotten because of already existing information covering or overlapping the former. This is differen to decay-based forgetting principles in which elements fade due to the mere passage of time and hence are very dependent on the order in which examples appear and rules are created. From the above experiments, we see that the repository of rules can be well structured and

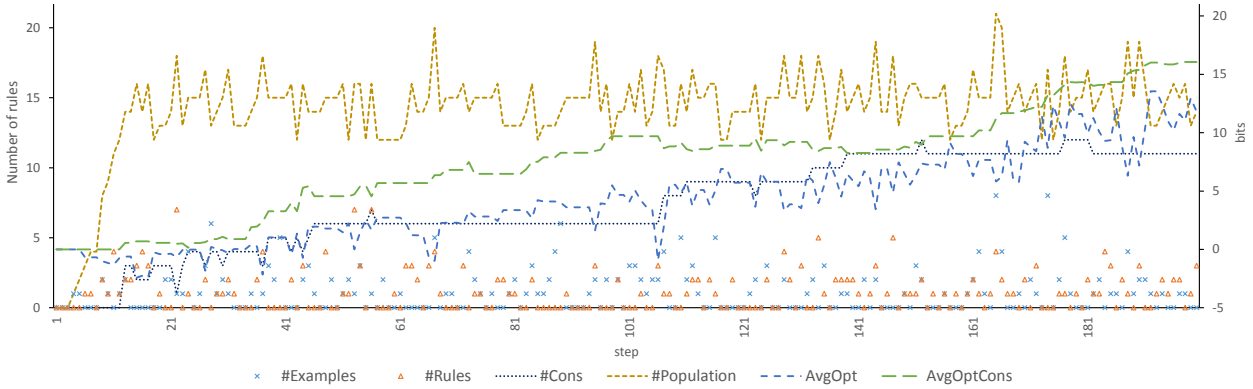


Figure 7: Evolution of the same indicators as in Figure 5 for the incremental chess problem (rook and bishop moves in the first 100 steps, and queen moves in the following 100 steps) *with* the forgetting mechanism (for a configuration with $|W| = 15$ and up to 25% of rules forgotten). We see a non-constant sawtooth-like picture for the number of rules. The forgetting mechanism takes place every few steps due to the small amount of rules allowed and the low percentage of rules discarded in every forgetting step. Nonetheless, the consolidated rules became constant in each different learning process.

Table 8: Consolidated rules and metrics (as in Table 4) for the chess problem at step 200 (the 100 first steps for learning the rook and bishop moves, and the 100 following steps for learning the queen moves). All legal queen moves are covered by using the previously learned rook and bishop moves. IDs in bold represent rules that perfectly generalise the rook, bishop and queen moves.

ID	ρ	$L(\rho)$	$\dot{S}_+$	$\dot{S}_-$	$-\dot{L}_+$	$-\dot{L}_-$	opt_+	opt_-	$Perm$
b10	move(bishop,pos(A,B),pos(C,D)) :- rdiff(B,D,E), fdiff(A,C,E).	34.275	590.635	0	566.101	-24.534	56.610	-534.024	56.610
q29	move(queen,pos(A,B),pos(C,D)) :- move(bishop,pos(A,B),pos(C,D)).	34.275	432.832	0	405.116	-27.716	40.511	-392.320	40.511
q25	move(queen,pos(A,B),pos(C,D)) :- move(rook,pos(A,B),pos(C,D)).	22.133	417.051	0	389.335	-27.716	38.933	-378.117	38.933
r15	move(rook,pos(A,B),pos(A,C)).	22.918	277.282	0	255.149	-22.133	25.514	-251.767	25.514
r16	move(rook,pos(A,B),pos(C,B)).	34.275	223.178	0	202.723	-20.455	20.272	-202.905	20.272
q12	move(queen,pos(A,C),pos(A,D))	13.214	173.583	0	160.000	-13.214	16.036	-157.546	16.036
r7	move(rook,pos(A,2),pos(B,2)).	28.993	175.838	0	156.120	-19.718	15.612	-160.226	-4.659
r14	move(rook,pos(A,2),pos(C,D)).	22.918	162.311	0	141.000	-21.133	14.117	-148.193	14.117
r9	move(rook,pos(a,B),pos(h,B)).	24.534	121.733	0	102.600	-19.133	10.260	-111.473	-10.011

ranked by the metrics and the system consolidates those rules that are appropriate, therefore responding affirmatively to the question (a) at the beginning of this section. Regarding question (b), we also see that a moderate limitation of working space with forgetting is even capable to improve the identification of the rules to be consolidated, and, what is better, prevents the system from stagnating or collapsing in situations where we have bounded resources. Finally, in connection with question (c), we see the behaviour in an incremental setting, where the knowledge previously acquired can be used in new tasks.

Consequently, the proposed approach for knowledge acquisition is a favourable compromise to the stability-plasticity dilemma: (plasticity) the promotion and demotion mechanisms together with the evaluation metrics rank and structure the knowledge allocated in the working space avoiding useful knowledge losses; (stability) the forgetting mechanism together with the evaluation metrics are in charge of removing those meaningless and redundant pieces of knowledge.

6 Conclusions

Learning a set of rules from data is nowadays a well-known problem for which many approaches exist, from data science to robotics. However, the use of background knowledge and the consolidation of new knowledge is one of the conspicuous problems in the understanding and creation of cognitive systems, and the management of more lifelong learning and knowledge acquisition systems. The organisation of complex knowledge structures in terms of coverage graphs allows for a straightforward and principled approach to knowledge acquisition, consolidation (promotion), revision (demotion) and forgetting. All this can be analysed at a meta-level, with the use of off-the-shelf deductive and inductive engines in charge of, respectively, establishing the relations between the different rules and the generation of new rules. This modularity, and the ability of dealing with declarative knowledge bases (logical, functional, algebraic, equational, grammatical, etc.) opens up a range of applications in learning, knowledge acquisition, developmental cognition, expert systems and other intelligent systems that are meant to have a non-ephemeral life.

The main contributions of this work are: (1) The first extension of the MML principle to a knowledge network (in the form of coverage graph). While the MML principle has a Bayesian inspiration, the metrics are more flexible than actual probabilities, stauncher when pieces of the working space are removed, and can be combined into metrics for different processes. (2) We show that the development of a formal epistemological setting to realise how knowledge can be acquired, supports a constructive and developmental knowledge acquisition processes. In particular, we have seen how the forgetting criterion is not only necessary when the working space is finite but it can even be beneficial. (3) Our approach is parametrisable to other cognitive or intelligent systems, as it works at a meta-level and is independent of the actual deductive and inductive mechanisms that are used underneath. (4) The nonmonoticity problem of knowledge acquisition and revision is approached in a more lightweight and robust way, and the system can cope with redundancy, inconsistency or even uncertainty produced by conflictg resolutions or complex semantic artefacts. (5) *The stability-plasticity dilemma* has been addressed efficiently.

We plan to apply the setting to some other applications, by using the same or other deductive and inductive engines, including rules dealing with numerical or continuous features, such as time or space, and other notions of non-crisp coverage and example certainty, such as probabilities, degrees of truth, fuzzy rules, etc. We also keep on with the integration into our declarative learning system gErl[5] (Martínez-Plumed et al., 2013). Furthermore, other cognitive or AI systems, such as decision support systems can benefit from the application of the metrics introduced here. Finally, two further desirable characteristics for our approach could be explored: (a) interactiveness, namely, the ability to find an additional (human or not) input source if a problem statement is ambiguous or incomplete; (b) contextually, that is, to identify, understand and extract contextual or even cognitive elements such as syntax, semantics, domain, time, location, environment, goal, ..., which may be useful to move beyond the current knowledge acquisition systems.

[5]For a first approach using gErl as a deductive and inductive engine, see (Martínez-Plumed et al., 2015, Appendix).

Acknowledgements

This work has been partially supported by the EU (FEDER) and the Spanish MINECO under grant TIN 2013-45732-C4-1-P and FPI-ME grant BES-2011-045099, Generalitat Valenciana PROMETEO2011/052 and the REFRAME project, granted by the European Coordinated Research on Long-term Challenges in Information and Communication Sciences & Technologies ERA-Net (CHIST-ERA), and funded by the Ministerio de Economía y Competitividad in Spain (PCIN-2013-037).

References

Alnajjar, F., Zin, I.B.M., Murase, K., 2009. A hierarchical autonomous robot controller for learning and memory: adaptation in a dynamic environment. Adaptive Behavior 17, 179–196.

Ans, B., Rousset, S., 1997. Avoiding catastrophic forgetting by coupling two reverberating neural networks. Comptes Rendus de l'Acadmie des Sciences - Series {III} - Sciences de la Vie 320, 989 – 997.

Bragaglia, S., Ray, O., 2014. Nonmonotonic learning in large biological networks, in: Proc. 24th Int. Conf. on Inductive Logic Programming.

Brin, S., Page, L., 1998. The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst. 30, 107–117.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr., E.R., Mitchell, T.M., 2010. Toward an architecture for never-ending language learning, in: Proc. of the 24th Conference on Artificial Intelligence, pp. 1306–1313.

Carpenter, G., Grossberg, S., 1988. The art of adaptive pattern recognition by a self-organizing neural network. Computer 21, 77–88.

Della Sala, S., 2010. Forgetting. Psychology Press.

Dere, E., Easton, A., Nadel, L., Huston, J. (Eds.), 2008. Handbook Of Behavioral Neuroscience. volume 18. Elsevier.

Eaton, E., Ruvolo, P.L., 2013. Ella: An efficient lifelong learning algorithm, in: ICML, pp. 507–515.

Eiter, T., Wang, K., 2008. Semantic forgetting in answer set programming. Artificial Intelligence 172, 1644 – 1672.

Erbas, M.D., Winfield, A.F.T., Bull, L., 2014. Embodied imitation-enhanced reinforcement learning in multi-agent systems. Adaptive Behaviour 22, 31–50.

Erdem, E., Ferraris, P., 2007. Forgetting actions in domain descriptions, in: Proc. of the 22nd AAAI Conference on Artificial Intelligence, pp. 409–414.

Ferri-Ramírez, C., Hernández-Orallo, J., Ramírez-Quintana, M.J., 2001. Incremental learning of functional logic programs, in: Functional and Logic Programming, pp. 233–247.

French, R.M., 1997. Pseudo-recurrent connectionist networks: An approach to the "sensitivity-stability" dilemma. Connection Science 9, 353–379.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A., 2014. A survey on concept drift adaptation. ACM Computing Surveys 46, 44.

Grossberg, S., 2013. Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world. N. Nets. 37, 1–47.

Hebb, D., 1949. The organization of behavior. Wiley, New York.

Henderson, R., 2014. Cumulative Learning in the Lambda Calculus. Ph.D. thesis. Imperial College London.

Hernández-Orallo, J., 2000. Constructive reinforcement learning. International Journal of Intelligent Systems 15, 241–264.

Hernández-Orallo, J., García-Varea, I., 2000. Explanatory and creative alternatives to the MDL priciple. Foundations of Science 5, 185–207.

Hourdakis, E., Trahanias, P., 2012. Computational modeling of observational learning inspired by the cortical underpinnings of human primates. Adaptive Behavior 20, 237–256.

Kitzelmann, E., 2010. Inductive programming: A survey of program synthesis techniques, in: 3rd Workshop AAIP.

Kleinberg, J.M., 1999. Authoritative sources in a hyperlinked environment. J. ACM 46, 604–632.

Lang, J., Liberatore, P., 2003. Propositional independence: Formula-variable independence and forgetting. Journal of Artif. Intel. Research 18, 2003.

Lang, J., Marquis, P., 2010. Reasoning under inconsistency: A forgetting-based approach. Artificial Intelligence 174, 799–823.

Lempel, R., Moran, S., 2000. The stochastic approach for link-structure analysis (salsa) and the tkc effect. Comput. Netw. 33, 387–401.

Leu, G., Curtis, N.J., Abbass, H.A., 2014. Society of mind cognitive agent architecture applied to drivers adapting in a traffic context. Adaptive Behaviour , 123–145.

Li, M., Vitányi, P.M., 2008. An Introduction to Kolmogorov Complexity and Its Applications. 3 ed., Springer Publishing Company.

Lin, F., Reiter, R., 1994. Forget it!, in: In Proc. of the AAAI Fall Symposium on Relevance, pp. 154–159.

Liu, Y., Wen, X., 2011. On the progression of knowledge in the situation calculus, in: Proc. of the Int. Joint Conference on Artificial Intelligence, p. 976.

Martínez-Plumed, F., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J., 2013. Learning with configurable operators and RL-based heuristics, in: New Frontiers in Mining Complex Patterns. LNCS, pp. 1–16.

Martínez-Plumed, F., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J., 2014. A knowledge growth and consolidation framework for lifelong machine learning systems, in: ICMLA, pp. 111–116.

Martínez-Plumed, F., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J., 2015. Forgetting and consolidation for incremental and cumulative knowledge acquisition systems. CoRR abs/1502.05615.

Muggleton, S.H., 1995. Inverse entailment and progol. New Generation Computing 13, 245–286.

Muggleton, S.H., Bain, M., Hayes-Michie, J., Michie, D., 1989. An experimental comparison of human and machine learning formalisms, in: In Proc. of 6th International Workshop on Machine Learning, pp. 113–118.

Muggleton, S.H., De Raedt, L., 1994. Inductive logic programming: Theory and methods. The Journal of Logic Programming 19, 629–679.

Nuxoll, A., Tecuci, D., Ho, W.C., Wang, N., 2010. Comparing forgetting algorithms for artificial episodic memory systems, in: Proc. of the Symposium on Human Memory for Artificial Agents. AISB, pp. 14–20.

Quiroga, R.Q., 2012. Concept cells: the building blocks of declarative memory functions. Nature Reviews Neuroscience 13, 587–597.

Raducanu, B., Vitri, J., 2008. Learning to learn: From smart machines to intelligent machines. Pattern Recognition Letters 29, 1024 – 1032.

Rendell, L.A., 1995. Lessons from theory revision applied to constructive induction, in: ICML, p. 185.

Rissanen, J., 1999. Hypothesis selection and testing by the mdl principle. The Computer Journal 42, 260–269.

Shannon, C.E., 1948. A mathematical theory of communication. Bell system technical journal 27.

Shastri, L., 2001. Biological grounding of recruitment learning and vicinal algorithms in long-term potentiation, in: Emergent Neural Computational Architectures Based on Neuroscience. volume 2036 of *LNCS*, pp. 348–367.

Su, K., Sattar, A., Lv, G., Zhang, Y., 2009. Variable forgetting in reasoning about knowledge. Journal of Artificial Intelligence Research 35, 677.

Thrun, S., 1996. Is learning the n-th thing any easier than learning the first, in: Advances in Neural Information Processing Systems, pp. 640–646.

Wallace, C., 2005. Statistical and Inductive Inference by Minimum Message Length (Information Science and Statistics). Springer.

Wallace, C.S., Boulton, D.M., 1968. An information measure for classification. The Computer Journal 11, 185–194.

Wallace, C.S., Dowe, D.L., 1999. Refinements of MDL and MML coding. Comput. J. 42, 330–337.

Wang, Y., Zhang, Y., Zhou, Y., Zhang, M., 2012. Forgetting in logic programs under strong equivalence., in: KR, pp. 643–647.

Wang, Z., Wang, K., Topor, R., Pan, J., 2010. Forgetting for knowledge bases in dl-lite. Annals of Mathematics and Artificial Intelligence 58, 117–151.

Wixted, J.T., 2004. The psychology and neuroscience of forgetting. Annu. Rev. Psychol. 55, 235–269.

Wood, R., Baxter, P., Belpaeme, T., 2011. A review of long-term memory in natural and synthetic systems. Adaptive Behavior , 1059712311421219.

Zhang, Y., Foo, N.Y., 2006. Solving logic program conflict through strong and weak forgettings. Artificial Intelligence 170, 739 – 778.

Zhang, Y., Zhou, Y., 2009. Knowledge forgetting: Properties and applications. Artificial Intelligence 173, 1525 – 1537.

About the authors

Fernando Martínez-Plumed is currently a PhD student in Machine Learning and Inductive Programming n the Department of Information Systems and Computation at the Technical University of Valencia (UPV, Spain). He received his B.Sc. degree (2009) in Computer Science and M.Sc. degree (2010) in Software Engineering, Formal Methods and Information Systems from the UPV. His main research interests are Machine Learning, Inductive Programming, Cognitive Systems and Data Mining.

Cèsar Ferri is currently an Associate Professor in the Department of Information Systems and Computation at the Technical University of Valencia (UPV, Spain). In 2003 he received his PhD in applying declarative languages for machine learning from the UPV. As a researcher, his work has focused mainly in Machine Learning and the following related topics: model evaluation, ROC analysis, multi-classifiers, Inductive Funtional-Logic Programming and Inductive Programming. He is co-author of several papers in these topics published in the top conferences and journals in the Machine Learning area.

José Hernández-Orallo is currently an Associate Professor in the Department of Information Systems and Computation at the Technical University of Valencia (UPV, Spain). He holds a B.Sc. and a M.Sc. in Computer Science from UPV, partly completed at the École Nationale Supérieure de l'Électronique et de ses Applications (France), and a Ph.D. in Logic with a doctoral extraordinary prize from the University of Valencia. His academic and research activities have spanned several areas of artificial intelligence, machine learning, data mining and intelligent system evaluation. He has published four books and more than a hundred journal articles and conference papers on these topics.

María José Ramírez-Quintana is currently an Associate Professor at the Technical University of Valencia (UPV). She recevied her PhD degree (1993) in Computer Science from the UPV. Her actual research work is oriented to multi-paradigm inductive programming, machine learning and data mining. In particular the main research activities of relevant interest are model evaluation for decision support, multi-class and hierarchical classification, distance-based methods and general learning models. She is co-author of several papers on international journals or presented in international meetings as the results of the research activity.