A knowledge growth and consolidation framework for lifelong machine learning systems

Fernando Martínez-Plumed, César 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—A more effective vision of machine learning systems entails tools that are able to improve task after task and to reuse the patterns and knowledge that are acquired previously for future tasks. This incremental, long-life view of machine learning goes beyond most of state-of-the-art machine learning techniques that learn throw-away models. In this paper we present a long-life knowledge acquisition, evaluation and consolidation framework that is designed to work with any rule-based machine learning or inductive inference engine and integrate it into a long-life learner. In order to do that we work over the graph of working memory rules and introduce several topological metrics over it from which we derive an oblivion criterion to drop useless rules from working memory and a consolidation process to promote the rules to the knowledge base. We evaluate the framework on a series of tasks in a chess rule learning domain.

Index Terms—Lifelong machine learning, oblivion criterion, knowledge topology and acquisition, declarative learning.

I. INTRODUCTION

The traditional view of machine learning (ML) applications usually portrays a training dataset from which a model is learnt and evaluated and a deployment dataset where the model is applied. Whenever a new application task appears, the old model is discarded and a new one is learnt from a new training set. Recently, different areas, such as transfer learning, multitask learning, lifelong ML, never-ending learners or even deep learning have tried to convert ML systems in more incremental tools able to learn knowledge and partially reuse it for other tasks.

The ingredients of a learning system that is able to acquire knowledge incrementally and arrange it properly in order to improve its performance are the following: (1) an inductive inference engine (generating patterns), (2) a deductive coverage engine (calculating knowledge interdependence), (3) a knowledge quality analyser (evaluating knowledge utility and consistency), (4) an oblivion procedure (discarding those patterns that seem useless) and (5) a consolidation process (promoting those patterns that are reliable and useful). The two first components are common in artificial intelligence (ML and automated deduction), but the latter three have usually been investigated in knowledge-based systems, cognitive science and informetrics (including bibliometrics and webometrics).

In this paper we develop a general framework that, given rule-based inductive and deductive subsystems, is able to constantly evaluate and arrange the rules in such a way that long-life ML becomes possible. The framework is based on a graph of deductive dependencies between examples and rules, from which several topological metrics are derived, such as support, certainty and significance. From these metrics we derive an oblivion mechanism to remove rules from working memory and a consolidation mechanism to promote rules to the long-term knowledge base. We evaluate the framework with a problem domain where several patterns about chess rules have to be learnt. We see how the system is able to keep the useful rules and perform differently when new tasks can take advantage of the knowledge that has been consolidated from previous tasks. To our knowledge, this is the first general and comprehensive framework for life-long ML that can be applied to a diversity of rule-based learning systems.

The rest of the paper is organised as follows. Section II presents the architecture of our framework and defines the metrics used to work with rules and knowledge. The experimental evaluation of our system is presented in Section III. Section IV reviews other related approaches in the literature. Finally, section V concludes the paper and outlines some future work.

II. APPROACH

The acquisition of learnt patterns after several tasks cannot be understood as a naïve accumulation of what has been learnt. New rules can be redundant or inconsistent with old ones or may build upon previously acquired knowledge. Despite some notable efforts [1]–[3], the integration of ML with knowledge acquisition is still mostly unresolved. One of the key issues is that learning systems must be capable of learning new things (plasticity) without losing previously learned concepts (stability). This is called the The Stability-Plasticity dilemma [4].

In this work we take a different approach by considering that we start with an inductive engine (e.g., a rule learning algorithm, an inductive logic programming (ILP) system or an inductive programming (IP) system) and a deductive engine (e.g., a coverage checker, an automated deduction system or a declarative programming language) and, over them, we construct a long-life ML system (see Figure 1).

![Fig. 1: Architecture of the long-life learning framework.](image-url)
For this purpose, several views have to be addressed: (1) The inductive engine can generate many possible hypotheses and patterns. Once brought to working memory we require metrics to evaluate how these hypotheses behave and how they are related. (2) As working memory and computational time are limited, we need an oblivion criterion to discard some rules. (3) The deductive engine checks the coverage of each hypothesis independently, using the consolidated knowledge as auxiliary rules, but not other working rules. As a result, only when new knowledge is consolidated we can use it for new problems or for more difficult examples of the same problem. (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. This means that rules must promoted and demoted.

One of our design considerations is that the approach should work with any inductive engine that is able to generate rules, such as association rule algorithms, IP systems such as gErI [5], or ILP systems such as Progol [6]. Actually, we must also provide a deductive engine for the representation language.

Let us now describe some notation that will be necessary to describe how the framework works in more detail.

### A. Notation

We consider that ‘rules’ are used for expressing examples (denoted as \( e \) or, alternatively, as \( e^+ \) and \( e^- \) when we refer to positive and negative examples), patterns and background knowledge. The set of all possible rules is denoted by \( \mathcal{R} \), where \( R \subset \mathcal{R} \) is the working space or memory, and \( B \subset \mathcal{R} \) is the background or consolidated knowledge base.

We represent the rules as vertexes or nodes \( v \) in an acyclic directed graph \( G(V,A) \) that we call coverage graph, because the directed edges \( A \) represent the coverage relations between the different rules\(^1\) as determined by the deductive engine. Hence, if there is an edge \( a = (x,y) \) (or \( x \to y \)), then \( y \) is said to be directly covered by \( x \) using \( B \). The set of ancestors and successors of a node \( v \) are defined as \( \text{anc}(v) = \{ x | x \to v \} \) and \( \text{suc}(v) = \{ y | v \to y \} \) (respectively). Figure 2 shows an example of Coverage Graph of a well-known ILP problem: family relationships. In this problem, the task is to define the target relation \( \text{daughter}(X,Y) \), which states that person \( X \) is daughter of person \( Y \). The problem consists of three positive examples, two negative ones, and seven selected rules that try to generalise and solve the problem (Table I right), whereas \( B \) is composed of the relations \( \text{female} \) and \( \text{parent} \) (Table I left). Note that the rules in \( B \) have not been included in the graph for clarity, although they belong to the initial consolidated knowledge.

### B. Metrics

From the coverage graph, we derive several indicators to determine which rules are more relevant, useful and consistent.

\(^1\)We say that a rule \( \rho_a \) is covered by another rule \( \rho_b \) if \( \rho_b \theta \)-subsumes \( \rho_a \), namely, \( \rho_a \prec \rho_b \) iff exists a substitution \( \theta \) such that \( \rho_b \theta \subseteq \rho_a \).

\(^2\)For simplicity, the coverage graphs do not include the edges for the transitive closure of the covering relation, i.e., if a node \( x \) covers nodes \( y \) and \( z \), but \( y \) also covers \( z \), only the edges \( x \to y \) and \( y \to z \) are included.

---

**Fig. 2:** 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 I.

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

**TABLE II:** Metrics for the rules on the right side of Table I.

1) **Support:** The support of a rule aims at representing how important a rule is in terms of coverage: the higher number of examples covered, the higher the support value is. All the rules have two kinds of support, positive and negative. The examples have a fixed support value and patterns. Once brought to working memory we require metrics to evaluate how these hypotheses behave and how they are related. (2) As working memory and computational time are limited, we need an oblivion criterion to discard some rules. (3) The deductive engine checks the coverage of each hypothesis independently, using the consolidated knowledge as auxiliary rules, but not other working rules. As a result, only when new knowledge is consolidated we can use it for new problems or for more difficult examples of the same problem. (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. This means that rules must promoted and demoted.

---

**TABLE II:** Metrics for the rules on the right side of Table I.

1) **Support:** The support of a rule aims at representing how important a rule is in terms of coverage: the higher number of examples covered, the higher the support value is. All the rules have two kinds of support, positive and negative. The examples have a fixed support value \( \text{Sup}_f \) that is given initially as: \( \text{Sup}_f(e^+) = 1 \), \( \text{Sup}_f(e^-) = 0 \), \( \text{Sup}_f(e^{-}) = 0 \) and \( \text{Sup}_f(e^{--}) = -1 \). For the rest of nodes, \( \text{Sup}_f(v) \) and \( \text{Sup}_f(v) \) are 0. From here, we propagate the support through the acyclic graph backwards:

\[
\text{Sup}^+(\rho) = \text{Sup}_f(\rho) + \sum_{v \in \text{anc}(\rho)} \frac{\text{Sup}^+(v)}{|\text{anc}(v)|}
\]

\[
\text{Sup}^-(\rho) = \text{Sup}_f(\rho) + \sum_{v \in \text{anc}(\rho)} \frac{\text{Sup}^-(v)}{|\text{anc}(v)|}
\]

Note that as we divide the support of the outcoming nodes by \( |\text{anc}(v)| \), the support is conservative. This means that the positive and negative supports of rule \( \rho \) are always an underestimate of how many positive and negative rules, respectively, \( \rho \) covers. For instance, the positive example with ID= 1 in Figure 2 is covered by the rules with IDs 100 and 110, so both of them receive half of its support (a positive support equal to

---

**TABLE II:** Metrics for the rules on the right side of Table I.

**TABLE II:** Metrics for the rules on the right side of Table I.

1) **Support:** The support of a rule aims at representing how important a rule is in terms of coverage: the higher number of examples covered, the higher the support value is. All the rules have two kinds of support, positive and negative. The examples have a fixed support value \( \text{Sup}_f \) that is given initially as: \( \text{Sup}_f(e^+) = 1 \), \( \text{Sup}_f(e^-) = 0 \), \( \text{Sup}_f(e^{-}) = 0 \) and \( \text{Sup}_f(e^{--}) = -1 \). For the rest of nodes, \( \text{Sup}_f(v) \) and \( \text{Sup}_f(v) \) are 0. From here, we propagate the support through the acyclic graph backwards:

\[
\text{Sup}^+(\rho) = \text{Sup}_f(\rho) + \sum_{v \in \text{anc}(\rho)} \frac{\text{Sup}^+(v)}{|\text{anc}(v)|}
\]

\[
\text{Sup}^-(\rho) = \text{Sup}_f(\rho) + \sum_{v \in \text{anc}(\rho)} \frac{\text{Sup}^-(v)}{|\text{anc}(v)|}
\]

Note that as we divide the support of the outcoming nodes by \( |\text{anc}(v)| \), the support is conservative. This means that the positive and negative supports of rule \( \rho \) are always an underestimate of how many positive and negative rules, respectively, \( \rho \) covers. For instance, the positive example with ID= 1 in Figure 2 is covered by the rules with IDs 100 and 110, so both of them receive half of its support (a positive support equal to
0.5 and a negative support equal to 0). The reason for this is that if too many rules are covering the same examples, their support is shared and reduced, thus avoiding many rules for a few examples.

2) Certainty or groundedness: The certainty or groundedness of a rule means how much we can trust in it, and its value decreases as the length of the paths that connect the rule to the example increases. This is natural since a concept that is too far from the actual examples is more general than them and, thus, it is less grounded. All the rules have two kinds of groundedness, the positive and the negative one. The examples have, in general, the highest possible certainty (they are the only ones in which we can trust completely, assuming a learning scenario without noise). Cert is a fixed value assigned to the examples with the initial values Cert\(^+\)\((e^+)\) = 1, Cert\(^-\)\((e^-)\) = 0 and Cert\(^-\)\((e^-)\) = -1. As we want positive certainty to go between 0 and 1 and negative certainty between -1 and 0, we use a logistic function \(L(x) = \frac{1}{1+e^{-x}}\):

\[
\begin{align*}
\text{Cert}^+(\rho) &= 2L(\text{Cert}^+(\rho)) + \sum_{\nu \in \text{sup}^{\rho}(\nu)} \text{Cert}^+(\nu) - 1 \\
\text{Cert}^-(\rho) &= 2L(\text{Cert}^-(\rho)) + \sum_{\nu \in \text{sup}^{\rho}(\nu)} \text{Cert}^-(\nu) - 1
\end{align*}
\]

3) Optimality or Significance: Support and certainty are related metrics but are focused on different things, in a similar way. The support is shared and reduced, thus avoiding many rules for a fixed set of examples. This idea leads to the following definition for the permanence of a rule:

\[
\text{Perm}(\rho) = \text{Opt}^{\rho} - \arg \max_{\nu \in \text{anc}(\rho)} \text{Opt}(\nu) + \frac{1}{\text{size}(\rho)}
\]

where size refers to the size (in bits) of the rule (definition available at [7]). Inspired by the MML philosophy [8], [9], the last term allows us to give more preference to those rules that are shorter.

The lower the value of permanence a rule has, the higher possibilities it has to be forgotten, depending on an oblivion threshold. Figure 3 shows the evolution of the coverage graph in Figure 2 through four oblivion steps.\(^3\) Due to space limitation, we do not include in the paper the measures for the rules, which are available at [7]. Following with the example, in step 1 we see that rule number 59 is redundant because it is covered by a more significant rule (with ID 110, Opt(110) = 1.34, Opt(59) = 0.231), and it has the lowest value of permanence (Perm(59) = -0.649 and Perm(110) = 1.55). Thus, rule 59 is forgotten, the coverage graph is redrawn and the metrics are recalculated.

In case there is an oblivion step that deletes any leaf node, its support is distributed equally among the rules that cover it, and added to the “fixed” or intrinsic support of these rules. Similarly, the certainty and its fixed values are also ‘inherited’ upwards (but not distributed).

D. Consolidated knowledge: promotion and demotion

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.

We will set a threshold \(\theta_p\) on the optimality to consolidate or promote a rule to a belief status. B is therefore continuously grown with new rules. The promotion system is mirrored by a demotion system, with the use of another threshold \(\theta_d\) such that if any rule in B has a lower value of optimality, then the rule is demoted to the working space. The original background knowledge (\(B_0\)) cannot be demoted (and forgotten).

In the example in Figure 3, we have established \(\theta_p\) equal to the average optimality (using the absolute value of \(\text{Opt}(\rho)\) before averaging) of all the rules in the working space. Then, in step 1, all the rules that exceed this average value (0.695) will be consolidated to the background base (rules 1, 2, 5, 73 and 110). Any rule that is promoted to the consolidation step cannot be target of the oblivion mechanism until it is demoted to the working space again (in the example, we have considered a demoting threshold \(\theta_d\) equal to \(\theta_p\)). Thus, in step 2, rule 1 has the lowest permanence value (Perm(1) = -0.357) but 35 (Perm(35) = -0.113) is forgotten instead, because the former is a consolidated rule.

III. EXPERIMENTS

As mentioned in section II, one of the issues in other learning systems is the Stability-Plasticity dilemma. We claim that our approach is able to address this issue in a lifelong learning process. For this purpose, we have conducted an experimental evaluation to explore the following questions: (a) is it possible to generate a large repository of consolidated knowledge assessing the usefulness of the rules? and (b) is our approach able to forget or revise the existing knowledge in order to generate a reusable knowledge base? We 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 any inductive and deductive engine. Additional information not included in the paper about the rules and their measures for each experiment is available at [7].
The inductive engine can produce rules it has already generated. Each example corresponds to a move on an empty board, and is represented by a triple from the domain $\text{Piece} \times \text{Pos} \times \text{Pos}$, where the second and third components are, respectively, the piece's initial position and its destination on the chessboard. Positions are expressed as tuples from the domain $\text{File} \times \text{Rank}$ where files (a-h) stand for columns and ranks (1-8) stand for rows. For instance, using a Prolog notation (as in the example in the previous section), the fact \text{move(knight,pos(d,5),pos(e,3))} represents a legal move of the knight from position (d,5) to (e,3). The only background predicate used is the absolute difference, \text{diff(X,Y)}, that calculates the distance between $X$ and $Y$, where both $X$ and $Y$ can be ranks or files (see Table III).

A random set of 50 chess moves (positive and negative examples) from all chess pieces except the pawn (rook, bishop, knight, queen, and king) is given. We also consider that an illegal (negative) move examples (extracted from [10]). Each example is assigned to the system in a random order. In our case, the inductive engine is generating rules during the whole process and they are arriving to the system in a random order. In our experiments, we assumed that the inductive engine can produce rules it has already generated. The inductive engine can take advantage of. Table V shows the set of consolidated rules which consists of the previously learnt general moves from the rook and the bishop pieces (see Table IV). In the second phase, we provided the system with the rules and examples of moves of the rook and bishop (15 and 30 rules respectively). The consolidation criterion has not been changed, but the maximum number of rules in the working space has been reduced to 15 and also the percentage of meaningless rules that are forgotten for each oblivion process (30%). In the first 100 steps of Figure 5 we can see that, due to the lower maximum number of rules allowed in the working space and the lower percentage of rules forgotten, the oblivion mechanism runs here every few steps, showing non-constant sawtooth-like wave ramps for the number of rules in the working space (dotted yellow line). However, the number of consolidated rules remains constant almost from the beginning. The set of consolidated rules after 100 steps perfectly generalises all the legal moves of the rook and bishop pieces (see Table IV). In the second phase, we provided the system with a new set of rules and examples (10 and 20 rules respectively) only representing moves of the queen. Apart from using the background knowledge that is provided initially, it should also be possible to use the previously learnt general moves from the rook and the bishop in order to express the moves of the queen. This is what the inductive engine can take advantage of. Table V shows the set of consolidated rules which consists of the previously learnt general moves from the rook and the bishop pieces.
rules that generalise the legal moves from the rook and bishop, and a new set of rules that represents the legal moves of the queen. This latter set includes a pair of rules \(q_{29}\) and \(q_{25}\) that use the rook and bishop rules and represent all the possible moves of the queen: \(q_{25}\), which covers both the horizontal and vertical moves of the queen; and \(q_{29}\), which covers the same move as the rule \(q_{23}\) (diagonal movement) but it is more likely to be forgotten in case of demotion due to its larger size.

**IV. RELATED WORK**

The results of the experiments presented in Section III show that our approach is able of incremental and continual learning based on the reuse of knowledge. This is the context of *Lifelong Machine Learning* [11], or LML, and related to areas such as Multiple Task Learning, Transfer Learning, Reinforcement Learning or Incremental Learning, have similar objectives.

One distinctive trait of our approach is that we use a rule-based approach and knowledge is arranged by coverage. This is different to other lifelong learning approaches that use *explanation-based neural networks* (EBNN) [12], back-propagation neural networks [13], [14] and reinforcement learning approaches [15]. Another important difference is that here we do not work with a task-to-task mapping as in transfer learning.

Closer to our approach we find ELLA (Efficient Lifelong Learning Algorithm) [16] and NELL (Never-Ending Language Learner) [17], which are another approaches to LML able to consolidate knowledge within a long-term domain knowledge structure. However they do not introduce oblivion mechanisms that could eliminate obsolete knowledge and lack the layered view of knowledge of a covering graph. Henderson [18] presents an inductive inference system that automatically acquires knowledge employed for solving harder problems through experience of solving easier tasks. This work is centred on abstract learning and the generalisation of rules, but does not grow the existing knowledge base systematically.

From a very different point of view, our coverage graph is also highly influenced by those problems related to link analysis in web graphs such as the HITS algorithm [19], PageRank [20] or SALSA [21]. Other related ideas appear in areas of informetrics, but in our case it is not reputation but explanatory relevance and consistency what matter.

**V. CONCLUSIONS**

In this paper we have presented a new framework to deal with lifelong machine learning, where knowledge is promoted from working space to consolidated knowledge (and vice versa). We have also introduced an oblivion mechanism to discard those rules that are useless. The framework is modular and easily adaptable to many inductive engines that are able to output rules (or other units of knowledge) provided that we have a deductive engine to check coverage between the rules, in order to create the coverage graph. The use of oblivion criteria can be used to help tuning the system according to how much space there is available, and most especially, about how many rules can be processed in a reasonable time by the deductive engine, as the working space graph needs to be recalculated for every step.

The goal of this work was to evaluate the framework in general terms. From the experiments we have seen that the metrics and thresholds work reasonably well. Additional evaluation is needed with more and different problems and other inductive and deductive engines in order to get further information about the quality of the metrics we have defined and the use of thresholds for forgetting, promoting and demoting.

**REFERENCES**


Fig. 5: Evolution of the same indicators as in Figure 4 for the incremental chess problem (rook and bishop moves in the first 100 steps, and queen moves in the following 100 steps). We see a non-constant sawtooth-like picture for the number of rules in the working space whereas the consolidated rules became constant in each different learning process, and the global optimality becomes stable between 1 and 1.5.

TABLE IV: Consolidated rules and metrics (from left to right: size, positive and negative certainty, positive and negative support, permanence, is consolidated?, and optimality) for the chess problem (rook and bishop moves) at step 100. All the legal moves for these pieces are covered by the rules and no better rules can be obtained.

<table>
<thead>
<tr>
<th>ID</th>
<th>Rule</th>
<th>Size</th>
<th>Cert⁺</th>
<th>Cert⁻</th>
<th>Sup⁺</th>
<th>Sup⁻</th>
<th>Perm</th>
<th>Cons</th>
<th>Opt</th>
</tr>
</thead>
<tbody>
<tr>
<td>b10</td>
<td>move(bishop,pos(A,B),pos(C,D)) :- rdiff(B,D,E), fdiff(A,C,E).</td>
<td>7.924</td>
<td>0.905</td>
<td>0.0</td>
<td>1.833</td>
<td>0.0</td>
<td>1.784</td>
<td>true</td>
<td>1.658</td>
</tr>
<tr>
<td>r15</td>
<td>move(rook,pos(A,B),pos(A,C)).</td>
<td>18.133</td>
<td>0.761</td>
<td>0.0</td>
<td>2.0</td>
<td>0.0</td>
<td>1.419</td>
<td>true</td>
<td>1.522</td>
</tr>
<tr>
<td>r16</td>
<td>move(rook,pos(A,B),pos(A,C)).</td>
<td>18.133</td>
<td>0.761</td>
<td>0.0</td>
<td>1.5</td>
<td>0.0</td>
<td>1.038</td>
<td>true</td>
<td>1.141</td>
</tr>
</tbody>
</table>

TABLE V: Consolidated rules and metrics (same as in Table IV) for the chess problem at step 200 (the 100 firsts steps for learning the rook and bishop moves, and the 100 following steps for learning the queen moves). All queen legal moves are covered by taking advantage of previously learned moves of rook and bishop pieces, whose legal moves are also covered.


