Apprentissage par Analogies grâce à des outils de la Théorie des Catégories

L. Cordesses[1] O. Bentahar[1] K. Poulet[1] A. Laurent[1] S. Amar[1] T. Ehrmann[1] J. Page[2]

[1] Renault Innovation Silicon Valley, 1215 Bordeaux Drive, Sunnyvale, 94089 CA, USA
[2] CNRS-Université Paris Diderot, Laboratoire SPHERE, 5 rue Thomas Mann, 75205 Paris cedex 13

{lionel.cordesses, kevin.poulet, thomas.ehrmann}@renault.com,
{aude.laurent, sarah.amar, omar.bentahar}@nissan-usa.com,
ju.page@hotmail.fr

Résumé

Les algorithmes d'apprentissage automatique utilisés pour contrôler des systèmes physiques doivent pouvoir apprendre rapidement, avec peu d'exemples, notamment lorsque le coût ou la durée des expérimentations sont trop importants. Un tel objectif peut être atteint en utilisant des concepts provenant des sciences cognitives et des sciences sociales, et formalisés grâce à des outils mathématiques de la théorie des catégories et de la théorie du contrôle.

Ce formalisme aboutit à un système d'apprentissage automatique qui accumule les connaissances, puis les transpose à de nouvelles configurations. Cette approche est illustrée sur un système cyber-physique (un circuit de voitures), et sur les jeux Atari 2600.

Mots Clef

Apprentissage Automatique, Théorie des Catégories, Atari 2600, Circuit de voitures miniatures.

Abstract

Machine learning algorithms for controlling devices need to learn quickly, with few trials, when limited by the duration and cost of experiments. Such a goal can be attained with concepts borrowed from Cognitive Science and Social Science, and formalized mathematically using tools from Category Theory and Control Theory.

This leads to a machine learning system that accumulates knowledge, then transposes it to new configurations. Illustrations of this approach are presented on a cyber-physical system – the slot car game – and also on Atari 2600 games.

Keywords

Machine Learning, Category Theory, Atari 2600, Slot Car.

1 Introduction

Les algorithmes qui contrôlent des systèmes cyber-physiques doivent apprendre comment opérer rapidement dans un environnement partiellement connu, le tout avec de moins en moins de données. Les solutions à base d'apprentissage par renforcement, avec des réseaux de neurones par exemple, ont fait leurs preuves. Cependant, ces méthodes à la pointe de la recherche ont besoin de beaucoup de données d'entraînement, données que l'on risque de ne pas pouvoir obtenir si l'on respecte des contraintes budgétaires et temporelles.

Nous proposons ici une approche complémentaire à l'apprentissage par renforcement et aux réseaux de neurones pour permettre aux machines d'apprendre rapidement avec des puissances de calcul réduites. L'objectif est d'être aussi performant que les solutions existantes avec environ un pour cent des données et du temps d'entraînement usuels.

L'aspect novateur de notre travail repose sur l'utilisation d'outils élémentaires de la *théorie des catégories* — une branche des mathématiques datant des années 1940, assez peu utilisée en apprentissage automatique (en anglais : Machine Learning) — pour formaliser notamment la notion d'*analogie*. La théorie des catégories a été conçue en lien avec la topologie algébrique, pour permettre de transférer des théorèmes et des concepts d'une branche des mathématiques à une autre [28]. Nous utilisons ces outils pour avoir un cadre générique pour notre approche d'apprentissage automatique, ce qui permet, par exemple, de formaliser rigoureusement le transfert de connaissances.

Par ailleurs, nous pensons qu'il est fondamental de rechercher des structures dynamiques pour construire une intelligence artificielle (IA) adaptative. Les sciences sociales et les sciences cognitives montrent que les structures dynamiques cadrent la cognition (cf. [22]), les langues (cf. [8]) et les interactions sociales (cf. [26]). En effet, ces structures permettent aux humains d'agir et de s'adapter en fonction des expériences passées.

Dans cet article, nous commençons par étudier les méthodes d'apprentissage automatique pour les processus décisionnels markoviens (MDP) appliqués à des jeux. Ensuite, nous expliquons les motivations qui supportent notre approche inspirée des sciences sociales. Grâce au concept d'*équivalence de catégories*, nous décrivons des classes de problèmes non bijectifs, sur lesquels nous centrons notre

approche. Il en résulte un comportement innovant de l'algorithme de contrôle. Enfin, nous illustrons cette approche avec des résultats sur un circuit de voitures miniatures, et sur des jeux vidéo Atari 2600. Notre méthode utilise également le savoir accumulé tout au long des expériences passées, et peut donc servir à valider des concepts modernes tels que le « lifelong machine learning » [5].

2 Travaux antérieurs

Nous avons décidé de tester notre approche sur un circuit de voitures miniatures et sur des jeux vidéo Atari, car ces deux expériences impliquent de prendre des décisions, tâche plus complexe que la classification pure. Dans ce contexte, les approches d'apprentissage automatique appliquées à la prise de décision reposent souvent sur de nombreux essais, ce qui implique de grandes quantités de données d'entraînement et requiert une grande puissance de calcul.

D'autre part, les jeux sont devenus un banc d'essai classique pour les algorithmes d'IA. Les circuits de voitures miniatures, par exemple, sont utilisés pour évaluer la performance des systèmes de prise de décision. Le traitement d'images basé sur de l'apprentissage par renforcement présenté dans [12] estime la position de la voiture sur le circuit grâce à un perceptron multicouche à convolution. L'entraînement prend $12\,\mathrm{h}$ et l'apprentissage de la stratégie de contrôle requiert $30\,\mathrm{min}$ supplémentaires. La solution plus rapide proposée par [24] pour le même système de circuit de voitures autonome dépend d'accéléromètres et d'un microcontrôleur rajoutés sur la voiture pour d'abord créer une cartographie du circuit, puis ensuite contrôler la vitesse de la voiture.

Les jeux vidéo sont aussi devenus de plus en plus utiles pour fournir une représentation cyber-physique de notre environnement. Même des systèmes anciens, tels que la console Atari 2600, fournissent une grande diversité de situations, allant des labyrinthes (jeux de style Pac-Man) et jeux d'action (tel que Space Invaders) aux jeux de balle et raquette (Breakout, Pong). Malgré le fait que ces différents problèmes réclament des stratégies variées pour être résolus par un joueur humain standard, ils impliquent tous des prises de décision et ont donc été modélisés par des MDPs [18]. Ce cadre permet l'implémentation de nombreuses méthodes différentes. Certains travaux, tel que celui présenté dans [18], utilisent une image de l'aire de jeu remise à l'échelle en entrée d'un « deep Q-network » (DQN), avec pour but de sélectionner la meilleure action à jouer. Une autre possibilité est l'utilisation de méthodes classiques de recherche et de planification, telles que le « Iterated Width algorithm » [16] ou les algorithmes de parcours d'arbre tels que l'algorithme de recherche arborescente Monte-Carlo [21] pour calculer la meilleure action possible. L'apprentissage par renforcement peu profond [15] repose sur une représentation linéaire simplifiée, mais obtient cependant des résultats similaires aux méthodes non-linéaires. Enfin, l'apprentissage maître-élève [3], l'apprentissage par renforcement inverse [13], et l'apprentissage par imitation [20] peuvent également être utilisés pour entraîner un agent au comportement plus humain, qui a une efficacité et une réussite proches de celles des méthodes susmentionnées. Les « Schema Networks » [11] font un pas supplémentaire en direction du transfert de connaissances et de la réduction des besoins en données et en puissance de calcul pour l'entraînement, en utilisant un simulateur physique et en travaillant à un niveau plus élevé avec des entités plutôt que des pixels. Cette méthode obtient le même score qu'un joueur humain sur Breakout (30 points) avec $100\,000$ images d'entraînement.

3 Sciences Sociales, Sciences Cognitives et Théorie du contrôle pour l'apprentissage automatique

Pour atteindre notre objectif de réduction des besoins en données d'entraînement et en puissance de calcul, nous avons basé notre IA sur des concepts de haut niveau déjà étudiés par différentes sciences.

Tout ce qui est produit par l'humain contient, en partie, une certaine structure humaine. Par exemple, lorsque des humains conçoivent un jeu, ils utilisent des règles implicites qui leur permettent de jouer et de s'amuser. Cela signifie que ces jeux, comme les autres créations humaines, contiennent un minimum de structure pour que le joueur humain puisse les comprendre, et ils contiennent également de la dissonance cognitive pour les rendre attractifs [6][25][27]. La plupart des problèmes réels possèdent cette ambivalence.

Parmi les structures clés, on retrouve les schémas cognitifs universels auxquels les humains s'attendent quant aux objets et à l'environnement : état intérieur (intention et affect), propriétés matérielles (gravité, conservation de la forme, continuité de la trajectoire) ou propriétés naturelles non-humaines (mouvement et croissance). Ces schémas cognitifs sont acquis durant l'enfance quand l'enfant découvre le monde au fil de ses mouvements et en utilisant ses sens [22].

Nous supposons qu'une IA ayant la connaissance de ces schémas cognitifs serait capable de prendre les bonnes décisions pour jouer, et de transférer les connaissances acquises à une grande variété de problèmes similaires.

3.1 Entités dans le temps et dans l'espace

Nous avons conçu notre IA pour qu'elle raisonne au niveau syntaxique et non au niveau de l'échantillon ou du pixel. D'une certaine manière, cela correspond à raisonner au niveau du morphème en linguistique, défini dans [8] comme le plus petit élément significatif d'un langage. Nous appelons entités les éléments de base de ce niveau syntaxique.

Ces entités sont détectées par des outils classiques de traitement du signal et d'apprentissage automatique. Elles sont comme les objets de notre vie de tous les jours : des tronçons de circuit (lignes droites et virages), des voitures, des

balles, des raquettes, des murs. Elles sont organisées géométriquement dans un espace et peuvent être décrites, par exemple, par les coordonnées cartésiennes de leur boîte englobante.

Les données caractérisant ces entités sont collectées à chaque échantillon de temps pour construire des modèles dynamiques décrits par les équations d'état (1) à temps discret pour un système d'ordre N_d, où x est le vecteur d'état, y la mesure, u la commande, A la matrice d'état, B la matrice de commande, C la matrice d'observation, D la matrice d'action directe, et k l'indice de l'échantillon.

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \tag{1}$$

3.2 Le Moi au sein du monde

Une étape clé pour se forger une identité propre lors du développement de l'enfant est appelée le *stade du miroir*. Nous avons conçu notre IA pour qu'elle commence à ce stade en identifiant le « Moi » parmis toutes les entités.

Cette identification du « Moi » est réalisée en utilisant des résultats de la théorie du contrôle, à savoir le critère de contrôlabilité d'un système dynamique, comme présenté, par exemple, dans [10]. Le système est complètement contrôlable si et seulement si (ssi) la matrice de contrôlabilité $C_{N_d} \triangleq [B|AB|\dots|A^{N_d-1}B]$ est de rang N_d. Nous définissons le « Moi » comme étant l'entité qui vérifie le critère de contrôlabilité.

Comme le « Moi » et son environnement peuvent tous les deux changer, les modèles établis peuvent ne plus être valides. En s'inspirant d'une idée provenant de l'approche scientifique décrite dans [23], selon laquelle il doit être possible d'invalider un système scientifique empirique par l'expérience, nous avons conçu notre algorithme de telle sorte à ce qu'il puisse réfuter ses propres modèles pour les remplacer par de meilleurs. Le critère pour cette décision est basé sur l'erreur entre les prédictions (basées sur le modèle) et les mesures. Cette remise en cause ne s'effectue pas seulement pendant la phase d'apprentissage mais également pendant la phase de test.

Une fois que le « Moi » est identifié, notre IA classifie les entités restantes entre amis et ennemis, d'une manière similaire à l'apprentissage par renforcement. Elle suit ensuite une simple stratégie de survie : essayer d'aller vers ses amis, à part si un ennemi est proche du « Moi », auquel cas la première chose à faire est de s'enfuir.

En résumé, notre IA prend en compte les récompenses de son environnement pour classifier les entités, comme en apprentissage par renforcement. Comme dans des approches scientifiques empiriques, elle met à jour les classifications et les modèles quand les mesures ne sont pas compatibles avec les prédictions. Enfin, elle décrit au moins une entité comme étant le « Moi », avec toutes les conséquences que cela implique pour sa survie.

4 La Théorie des Catégories comme cadre pour des analogies non bijectives

L'un des outils les plus efficaces dont dispose l'humain pour évoluer dans une situation inconnue est sa capacité à faire des *analogies* entre cette nouvelle situation et ses expériences passées. Nous pressentons qu'une IA capable d'établir des analogies, et sachant, par exemple, comment jouer au jeu Breakout sera capable de transposer ses capacités au jeu Pong, même si les aires de jeu et les règles ne sont pas identiques, à l'instar d'un joueur de tennis qui saurait au moins partiellement comment manier une raquette de badminton même s'il n'a jamais pratiqué ce sport.
Les situations où deux problèmes ont exactement le même nombre d'états et des structures isomorphes sont rares. La notion d'analogie semble alors adaptée au rapprochement de structures conceptuellement proches mais mathématiquement différentes. Or, la plupart des tentatives de formalisation de cette notion sont fondées sur une perspective structuraliste qui s'articule autour de la *théorie des ensembles via* le concept d'isomorphisme qui préserve les structures des ensembles. Il existe des outils mathématiques pour identifier des structures non isomorphes tels que l'équivalence de catégories en théorie des catégories [17]. En effet, celle-ci apparaît comme un cadre de travail plus naturel pour aborder la notion structuraliste d'analogie qui considère que les relations entre éléments sont plus essentielles que les éléments eux-même pour définir des analogies [9]. [1].
Dans les problèmes qui reposent sur la théorie des ensembles, l'identification se réduit aux relations d'identité et aux cas bijectifs, tandis que la théorie des catégories apporte des descriptions plus riches des objets, ce qui permet de nouvelles sortes d'identifications. Dans le cadre de cet article, nous nous limitons à des concepts très élémentaires de la théorie des catégories, qui pourraient être décrits en termes ensemblistes, tout comme les situations que l'on expose [2]. L'apport de ces travaux est le changement de perspective conceptuelle que nous croyons prometteur pour le domaine de l'apprentissage automatique [3].

Théorie des catégories. Une catégorie $\mathcal{C}$ est une collection d'objets et de morphismes (ou flèches) entre certains de ces objets, munie d'une composition de morphismes, et peut ainsi être assimilée à un graphe orienté. Si A et B sont

1. Une autre approche pourrait être l'utilisation de la théorie des modèles, plus spécifiquement du concept d'équivalence élémentaire [4] dans la perspective de doter l'IA de raisonnements logiques. Pour un panorama philosophique des concepts d'analogie et de raisonnement analogique, voir [1]

2. Ce qui n'est pas étonnant puisque la théorie des ensembles et la théorie des catégories, en tant que propositions mathématiques fondationnelles de toutes les mathématiques, se fondent mutuellement l'une l'autre. Chaque concept catégorique doit pouvoir être décrit en termes ensemblistes et réciproquement.

3. Dans une perspective bayésienne, [7] utilise des catégories de probabilités conditionnelles : les objets sont des espaces mesurables et les flèches des noyaux de Markov.

des objets de $\mathcal{C}$, la flèche $a : A \to B$ est un isomorphisme si elle est inversible, *i.e* s'il existe une flèche $b : B \to A$, telle que $ba = Id_A$ et $ab = Id_B$. Dans ce cas, les objets A et B sont isomorphes. Les isomorphismes définissent une relation d'équivalence sur la classe des objets de $\mathcal{C}$, pour laquelle on note $\mathcal{C}/\simeq$ son quotient. Aussi, si $F : \mathcal{C} \to \mathcal{C}'$ est ce qu'on appelle une équivalence de catégories, celle-ci induit une bijection $F' : (\mathcal{C}/\simeq) \to (\mathcal{C}'/\simeq)$ entre les classes d'objets isomorphes même si F n'est pas bijective. Dès lors, on n'identifie plus les objets (ou états) un à un entre deux situations, mais les types (ou classes d'isomorphismes) de ces états.

Ce procédé peut être utilisé dans le cas de problèmes observables. En effet, considérons deux ensembles d'états non vides $\mathcal{C}$ et $\mathcal{C}'$ sans autre hypothèse sur leur cardinalité. Supposons aussi l'existence de deux fonctions d'observations $f : \mathcal{C} \to O$ et $f' : \mathcal{C}' \to O'$. Quitte à restreindre O et O', supposons que ces deux fonctions sont surjectives. Pour tout $o \in O$, on dit que les états $x \in \mathcal{C}$ dont l'observation associée est o (*i.e* $f(x) = o$) sont de type T_o. Cela définit une relation d'équivalence R_f sur l'ensemble $\mathcal{C} : \forall x, y \in \mathcal{C}, x\ R_f\ y$ ssi $f(x) = f(y)$. Transposé dans le champ lexical des catégories, cela revient à placer une flèche inversible entre deux objets x et y de $\mathcal{C}$ ssi $x\ R_f\ y$. $\mathcal{C}$ devient alors une catégorie, où toutes les flèches sont inversibles et telle que $\mathcal{C}/\simeq$ est exactement le quotient $\mathcal{C}/R_f$, quotient qui définit aussi l'ensemble des types d'états de $\mathcal{C}$. Puisque ces types ont été définis via les observations, la surjection $f : \mathcal{C} \to O$ induit une bijection $\tilde{f} : (\mathcal{C}/\simeq) \to O$ entre l'ensemble de types et l'ensemble d'observations. $\tilde{f}$ est donc l'inverse de la fonction $T : O \to (\mathcal{C}/\simeq), o \mapsto T_o$ qui définit les types T_o. Le même raisonnement peut être reproduit à partir de $\mathcal{C}'$ et de f'.

Enfin, supposons qu'il existe une bijection $G : O \to O'$ entre les ensembles d'observation et que O et O' sont donc de même cardinalité. Ainsi, on peut définir une autre bijection $F' = \tilde{f'}^{-1} \circ G \circ \tilde{f} : (\mathcal{C}/\simeq) \to (\mathcal{C}'/\simeq)$ entre les ensemble de types d'objets, bijection en réalité induite par l'équivalence de catégories $F : \mathcal{C} \to \mathcal{C}'$ définie comme suit : pour tout $x \in \mathcal{C}$, soit $o = f(x)$ et soit $x' \in f'^{-1}(G(o))$, définissons F telle que $F(x) = x'$. Si $\mathcal{C}$ et $\mathcal{C}'$ ont des cardinaux différents, F ne peut pas être bijective, mais F' l'est. Par construction, F envoie chaque état x vers un état x' du même type (modulo G). Cela permet de relier les catégories $\mathcal{C}$ et $\mathcal{C}'$, de telle sorte que si l'on dispose d'une stratégie applicable dans $\mathcal{C}'$, elle peut être transposée dans $\mathcal{C}$ par la fonction F.

Si l'on remplace $\mathcal{C}$ et $\mathcal{C}'$ par des ensembles d'entités (option 1) ou de paires d'entités (option 2) dans deux (situations de) jeux différent(e)s, nous pouvons les comparer grâce à des fonctions d'observation du type f et f' comme décrit ci-dessus. Cela permet de transférer des connaissances relatives à des entités ou des paires d'entités. Nous avons ainsi expérimenté les cas particuliers où dans l'option 1, $f : \mathcal{C} \to \{0, 1\}$ et $f' : \mathcal{C}' \to \{0, 1\}$ valent 1 si l'entité est un « Moi » et 0 sinon ; et dans l'option 2, $f : \mathcal{C} \to \{0, 1\}$ et $f' : \mathcal{C}' \to \{0, 1\}$ valent 1 si les trajectoires des deux entités d'une paire s'intersectent, et 0 sinon. Ce qui est observé du Moi d'une situation peut alors être supposé du moi d'une autre situation par transfert de connaissances. Des observations ultérieures permettront alors de confirmer ou infirmer ces suppositions.

Une explication plus détaillée et approfondie de la théorie des catégories peut être trouvée dans [17]. Son utilisation nous permet de formaliser une grande diversité de jeux et de situations.

Analogies entre circuits de voitures miniatures. Nous introduisons les notations suivantes : soient N et N' les nombres de segments par configuration du circuit, et $\mathcal{C} = \{s, s \in [1, N]\}$, $\mathcal{C}' = \{s', s' \in [1, N']\}$ les ensembles de positions possibles de la voiture sur le circuit, qui ne sont utiles que pour le raisonnement théorique et ne sont pas utilisés lors des expériences. Soient $(u, i)_s$ (resp. $(u', i')_{s'}$) la tension et le courant mesurés lorsque la voiture passe sur la section s (resp. s'). Soit $1 \leq s_0 \leq N$ (resp. $1 \leq s'_0 \leq N'$) la position initiale de la voiture dans la configuration $\mathcal{C}$ (resp. $\mathcal{C}'$). Nous notons k une ligne droite de $\mathcal{C}$ et l un virage. De même, soient k' une ligne droite et l' un virage de $\mathcal{C}'$.

Le joueur peut agir sur $(u', i')'_s$ en utilisant la manette de jeu, ce qui correspond à la stratégie π' définie par (2).

$$\pi'(s') = \begin{cases} (u', i')_{k'}, & \text{si } s' \text{ est une ligne droite} \\ (u', i')_{l'}, & \text{sinon} \end{cases} \tag{2}$$

Nous voulons *identifier* $\mathcal{C}$ et $\mathcal{C}'$, pour pouvoir transposer la stratégie π' de $\mathcal{C}'$ à $\mathcal{C}$. Les états de $\mathcal{C}$ sont les positions s de la voiture sur le circuit. De même, les états de $\mathcal{C}'$ sont les positions s'. Si $N = N'$ et $s_0 = s'_0$, nous pouvons définir une bijection entre $\mathcal{C}$ et $\mathcal{C}'$ et transposer π' de manière triviale. En revanche, si $N \neq N'$ ou $s_0 \neq s'_0$, il est impossible de définir une telle bijection.

Cependant, si nous transformons $\mathcal{C}$ et $\mathcal{C}'$ en catégories, en définissant des morphismes, nous serons capables de définir une équivalence de catégories $F : \mathcal{C} \to \mathcal{C}'$. Dès lors, nous définirons la stratégie π appliquée sur $\mathcal{C}$ par $\pi = \pi' \circ F$. Pour définir ces morphismes, nous utilisons la fonction observable f définie sur les états s de $\mathcal{C}$ comme suit : $f : \mathcal{C} \to \{1, 2\}$ avec $f = h \circ g$ où g et h sont telles que $g(s) = (u, i)_s$ et $h\left((u, i)_s\right) = 1$ si s est un virage, et 2 sinon. f' est définie de la même manière sur les s' de $\mathcal{C}'$, c'est-à-dire $f' : \mathcal{C}' \to \{1, 2\}$ avec $f' = h' \circ g'$ et g' et h' sont définies de la même manière que g et h, mais sur les états de $\mathcal{C}'$.

Nous définissons un isomorphisme entre deux états de $\mathcal{C}$ ssi ils ont la même image par f, et un isomorphisme entre deux états de $\mathcal{C}'$ ssi ils ont la même image par f'. Nous définissons ensuite $F : \mathcal{C} \to \mathcal{C}'$ par (3).

$$F(s) = \begin{cases} l', & \text{si } f(s) = 1 \\ k', & \text{si } f(s) = 2 \end{cases} \tag{3}$$

Il est facile de voir, si les définitions exactes sont connues, que (3) est une équivalence de catégories qui permet de transférer π de $\mathcal{C}'$ à $\mathcal{C}$. F induit une bijection F' entre les ensembles de classes (ou types de position – ici $\mathscr{C}$,$\mathscr{C}'$ sont les types de virage et $\mathscr{S}$,$\mathscr{S}'$ sont les types de lignes droites) : $F' : (\mathcal{C}/\simeq) \to (\mathcal{C}'/\simeq)$ où $(\mathcal{C}/\simeq)$ est égal à $\{\mathscr{C}, \mathscr{S}\}$ et $(\mathcal{C}'/\simeq)$ est égal à $\{\mathscr{C}', \mathscr{S}'\}$. On obtient finalement $F'(\mathscr{C}) = \mathscr{C}'$ et $F'(\mathscr{S}) = \mathscr{S}'$.

Cet exemple de catégorisation systématique et de généralisation prouve que nous ne travaillons pas à l'échelle des états, mais que nous considérons les types d'états.

Analogies entre jeux vidéo. Considérons un jeu $\mathcal{C}$ (par exemple le jeu Breakout pour Atari 2600) où une raquette horizontale est située en x_0 sur un segment $[0, N]$ d'états initiaux possibles ($0 \leq x_0 \leq N$) et peut se déplacer vers la gauche ou la droite afin d'atteindre une balle devant arriver à la position $x_1 \in [0, N]$. La stratégie π est la suivante : si $x_1 < x_0$, se déplacer vers la gauche, si $x_1 = x_0$, ne pas bouger, mais si $x_1 > x_0$, alors se déplacer vers la droite. Désormais, considérons un autre jeu $\mathcal{C}'$ (tel que Pong sur Atari 2600 par exemple) où cette fois-ci une raquette verticale se trouve en position y_0 sur le segment $[0, N']$ des positions atteignables ($0 \leq y_0 \leq N'$) et peut se déplacer de haut en bas pour toucher une balle qui atteindra la position $y_1 \in [0, N']$.

Nous souhaitons *identifier* $\mathcal{C}$ et $\mathcal{C}'$ afin de transposer la stratégie π de $\mathcal{C}$ à $\mathcal{C}'$. Les états (objets) de $\mathcal{C}$ sont les $N + 1$ positions x_1 de $[0, N]$. De la même façon, les états (objets) de $\mathcal{C}'$ sont les $N' + 1$ positions y_1 de $[0, N']$. Si $N = N'$ et $x_0 = y_0$, alors il est facile de définir une bijection de $\mathcal{C}$ à $\mathcal{C}'$ puis de transposer π, mais si $N \neq N'$ ou $x_0 \neq y_0$ alors il est impossible de définir une telle bijection. Pour résoudre ce problème, nous transformons $\mathcal{C}$ et $\mathcal{C}'$ en catégories à travers la définition suivante de nos flèches, et ce faisant, nous pourrons définir une équivalence de catégories $F : \mathcal{C} \to \mathcal{C}'$. Pour définir ces flèches, nous utilisons les fonctions d'observation suivantes définies sur les états de $\mathcal{C}$ et $\mathcal{C}' : f : \mathcal{C} \to \{0, 1, 2\}$ et $f' : \mathcal{C}' \to \{0, 1, 2\}$ telles que $f(x_1) = 0$ ssi $x_1 < x_0$, $f(x_1) = 1$ ssi $x_1 = x_0$ et $f(x_1) = 2$ ssi $x_1 > x_0$. De la même façon, $f'(y_1) = 0$ ssi $y_1 < y_0$, $f'(y_1) = 1$ ssi $y_1 = y_0$ et $f'(y_1) = 2$ ssi $y_1 > y_0$. Ensuite, nous définissons une flèche inversible entre deux états de $\mathcal{C}$ ssi ils ont la même image par f, ainsi qu'une flèche inversible entre deux états de $\mathcal{C}'$ ssi ils ont la même image par f'. Soit $F : \mathcal{C} \to \mathcal{C}'$, telle que si $x_1 < x_0$, alors $F(x_1) = 0$; si $x_0 = x_1$, alors $F(x_1) = y_0$; et si $x_1 > x_0$, alors $F(x_1) = N'$. Cette fonction F définit une équivalence de catégories et induit une bijection F' entre les ensembles de classes : $F' : (\mathcal{C}/\simeq) \to (\mathcal{C}'/\simeq)$. $(\mathcal{C}/\simeq)$ correspond (grâce à la stratégie π) aux actions disponibles $\{Gauche, Attendre, Droite\}$ dans $\mathcal{C}$ alors que $(\mathcal{C}'/\simeq)$ correspond aux actions disponibles $\{Haut, Attendre, Bas\}$ dans $\mathcal{C}'$ (en choisissant d'orienter l'axe des y du haut vers le bas). Dans ce cas-ci $F'(Gauche) = Haut$, $F'(Attendre) = Attendre$ et $F'(Droite) = Bas$.

5 Résultats Expérimentaux

5.1 Circuit de voitures miniatures

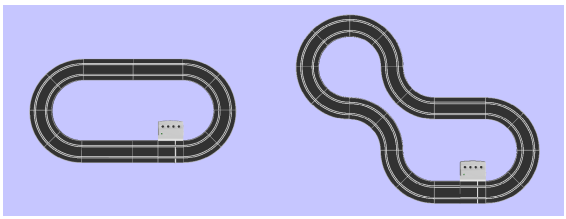

FIGURE 1 – Configuration des circuits : circuit 1 (à gauche), circuit 2 (à droite)

Matériel et Implémentation. Le montage expérimental est constitué d'un circuit MINI Challenge Set C1320 de la marque Scalextric dont le compteur de tours mécanique a été remplacé par un capteur à effet Hall omnipolaire digital. Le courant est mesuré par une résistance shunt placée en série avec les rails d'alimentation. Le courant et la tension sont d'abord filtrés par un filtre RC passif du second ordre puis échantillonnés à $f_s = 100\,\text{Hz}$. Ce montage ne contient aucun capteur supplémentaire. Les algorithmes sont écrits en C et exécutés en temps réel sur un Arduino Mega 2560 qui dispose de 8192 octets de mémoire vive (RAM). Nous définissons la période d'échantillonnage par $t_s = 1/f_s$, et l'instant associé à l'échantillon k est kt_s où $k \in \mathbb{N}$.

Ce cas basé sur l'utilisation d'analogies, directement inspiré de notre utilisation de la théorie des catégories, repose sur deux modules : un module de récompense, ainsi qu'un module de prise de décision, tous deux décrits ci-dessous. Puisque qu'il n'y a ici qu'un seul système dynamique, c'est nécessairement le « Moi » et aucune identification n'est utile.

Comme en apprentissage par renforcement, notre approche s'appuie sur une récompense tirée de l'environnement. Celle-ci est le résultat de la combinaison de trois variables. La première variable est le temps du tour mesuré directement grâce au compteur de tours. La seconde variable binaire indique la présence de la voiture sur la piste, et la dernière variable binaire indique si la voiture est en mouvement. Ce module de récompense surveille constamment la voiture pour vérifier qu'elle n'est pas sortie de la piste à cause d'une vitesse trop élevée, mais aussi qu'elle ne s'est pas arrêtée à cause de frottements trop importants. Ces deux détecteurs reposent sur une classification par k plus proches voisins (k-NN) en utilisant les courants et tensions comme entrées.

Avec ce module de récompense, l'IA peut piloter la voiture sur des circuits qu'elle découvre sans rejouer ou manipuler des échantillons enregistrés lors d'une partie du joueur humain. La seule information conservée par l'algorithme est une vitesse de sécurité dont l'IA sait qu'elle ne provoque ni sortie de piste, ni arrêt de la voiture. Comme la configuration du circuit change, il n'y a pas bijection entre

TABLE 1 – Temps au tour en secondes (moyenne $\pm$ écart-type)(MLI = Modulation de Largeur d'Impulsion)

	HUMAIN (10 tours)	**IA (jusqu'à 10 tours)**	**RÉGLAGES DE L'IA**
CIRCUIT 1 (12 tronçons)			
Premier tour	2.99 ± 0.46	$\mathbf{3.12 \pm 0.09}$	MLI=39% de la vitesse maximale
Dernier tour	2.29 ± 0.14	$\mathbf{2.52 \pm 0.08}$	Analogies (adaptation de la vitesse)
CIRCUIT 2 (18 tronçons)			
Premier tour	4.30 ± 1.16	$\mathbf{3.66 \pm 0.03}$	MLI=39% de la vitesse maximale
Dernier tour	3.08 ± 0.54	$\mathbf{3.13 \pm 0.02}$	Analogies (adaptation de la vitesse)

les deux configurations et le cas bijectif ne peut pas s'appliquer. Il faut donc s'appuyer sur des analogies et transposer les connaissances acquises dans une première configuration en utilisant l'équation (2), conformément au formalisme des cas non bijectifs décrit dans la partie 4. La fonction $h((u,i)_s)$ est évaluée par un k-NN dont le coefficient à été choisi comme le meilleur compromis entre la performance possible sur la cible embarquée et l'obtention de résultats satisfaisants. Les deux classificateurs sont entraînés avec environ 1000 échantillons de (u,i) (ce qui correspond à seulement 10 s de jeu) et sont implantés en version condensée afin d'être exécutables sur l'Arduino.

En pratique, l'approche par analogies fonctionne comme suit : la voiture démarre sur le circuit inconnu avec cette vitesse de sécurité importée de la première configuration. La fonction $h((u,i)_s)$, évaluée par le k-NN à partir des mesures de courant et de tension, détermine si la voiture est dans une configuration que nous appelons courbe ou ligne droite. L'IA choisit ensuite la meilleure commande utilisée au cours des expériences précédentes, en vérifiant l'objectif de minimiser le temps au tour en maintenant la voiture sur la piste. L'algorithme permet donc de généraliser les connaissances acquises au cours d'expériences passées et de les appliquer dans une configuration radicalement différente. En effet, les deux circuits schématisés en figure 1 sont de forme et de taille différentes et une simple reproduction d'une commande préalablement enregistrée échouerait à fournir des résultats probants.

Résultats. Les expériences décrites dans cet article ont été menées sur deux configurations de complexité différente présentées figure 1, et leur résultats sont présentés tableau 1. L'IA démarre à vitesse constante (avec une Modulation de Largeur d'Impulsion de 39% de la vitesse maximale). Cette IA, qui repose sur l'établissement d'analogies entre configurations et ne rejoue pas d'enregistrement d'une quelconque partie précédente, est capable d'améliorer les temps au tour en moins de 10 tours sur une piste inconnue. L'algorithme atteint des temps similaires aux temps humains sur le circuit 2 : 3.08 s pour les derniers tours des joueurs humains contre 3.13 s pour l'algorithme. Les améliorations futures de l'algorithme sur un circuit inconnu incluront l'optimisation des vitesses transposées par l'algorithme. En effet, seule une vitesse de sécurité a été utilisée ici afin d'éviter les sorties de pistes des voitures pilotées par l'algorithme, alors que certains tours humains ont occasionné des sorties de pistes et n'ont donc pas été comptabilisés.

Le cadre théorique détaillé en section 4 permet au système d'être au niveau des meilleurs joueurs sur ce jeu en moins d'une minute, sans ajout de capteurs embarqués. Ce cadre permet d'atteindre de tels résultats sur des circuits inconnus où la simple reproduction d'un tour humain conduirait immédiatement à une sortie de piste.

5.2 Jeux vidéo Atari 2600

Configuration et implémentation de la théorie. Tandis que le circuit de voitures miniatures nous a permis de valider l'approche sur des signaux analogiques réels, Arcade Learning Environment (ALE, cf [2]) nous a permis de la valider sur des configurations plus complexes, avec des signaux déjà échantillonnés provenant de l'émulateur. Les concepts d'entités avec le « Moi » introduits en 3.2 sont également utilisés pour jouer aux jeux Atari 2600. On détecte les entités grâce à des algorithmes de traitement d'images : le filtre de Sobel (image du centre en figure 2) et la détection des boîtes englobantes (image de droite en figure 2). Elle repose sur la librairie OpenCV [19]. Le « Moi » est trouvé en utilisant un algorithme d'identification de système. Des séquences d'actions pseudo-aléatoires [14] sont envoyées comme commande dans ALE pour d'abord identifier quelles entités sont affectées par ces signaux, puis pour construire les modèles dynamiques décrits par les équations (1) avec $N_d = 2$. Peu d'entités sont contrôlables : ce sont les « Moi ». Leurs formes peuvent changer en cours de jeu, comme la raquette dans Breakout, d'où la possibilité d'identifier plusieurs entités comme étant le « Moi ». Ces mesures mettent également à jour les fonctions $p(E,F)$ qui expriment la probabilité que le contact entre les entités E et F change le score, d'une manière similaire à une fonction de récompense en apprentissage par renforcement. À partir de ces fonctions p, nous déduisons les entités amies et ennemies, conduisant à la stratégie de survie basique décrite en partie 3.2.

Nous choisissons d'utiliser le DQN comme référence, car cette publication obtient de bons résultats par rapport à un joueur humain pour un grand nombre de jeux. Les tests ont donc été effectués avec les paramètres décrits dans [18] : l'IA joue pendant une durée maximale de 5 minutes. Même si notre objectif est de contrôler des systèmes cyber-

TABLE 2 – Scores après 10 000 images d'entraînement (moyenne ± écart type pour 8 parties)

JEU	ALÉATOIRE	JOUEUR HUMAIN	DQN	MLCT
Breakout	1.7	31.8	1.25 ± 1.02	**414.37 ± 88.47**
Pong	−20.7	9.3	−21.00 ± 0.00	**0.25 ± 2.48**

TABLE 3 – Effet du transfert de connaissances de Breakout vers Pong. Méthode (a) : Aucun transfert d'amis donc actions aléatoires. Méthode (b) : Stratégie de survie basique. Méthode (c) : Transfert de connaissances de Breakout à Pong

MÉTHODE	SCORE	NOMBRE D'IMAGES D'ENTRAÎNEMENT
(a)	−20	500 (Pong)
(b)	0	10 000 (Breakout) + 10 000 (Pong)
(c)	−2	10 000 (Breakout) + 500 (Pong)

physiques, nous souhaitions valider la versatilité de notre approche en la testant d'abord sur cet environnement de référence. Nous avons entièrement reproduit la configuration de cet article en utilisant le code mis à disposition par les auteurs, et avons obtenu des résultats similaires à ceux annoncés. Cela nous a permis de calculer un score moyen pour le DQN avec un faible nombre d'images d'entraînement, pour le comparer au score moyen obtenu avec notre approche.

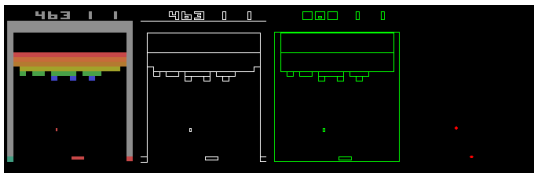

FIGURE 2 – Traitement d'images sur Breakout. De gauche à droite : original, contours, boîtes englobantes, prédictions des vitesses.

Résultats. Les résultats pour Breakout [4] sont présentés dans la table 2 pour un temps d'entraînement de 10 000 images (moins de 3 minutes). La figure 3 compare cette approche avec le DQN et montre que des scores similaires sont atteignables en utilisant 20 000 fois moins d'images.
Les étapes d'apprentissage sur Breakout sont les suivantes : durant les premières centaines d'images, l'IA utilise une séquence pseudo-aléatoire pour identifier le « Moi ». Une fois que cette identification a convergé vers le seul système que l'IA contrôle – la raquette – l'algorithme recherche des entités amies et ennemies. En quelques milliers d'images, il détecte que la balle est un ami car le score augmente quand elle casse une brique.
Les scores autour de 400 points correspondent à des parties où quasiment toutes les briques du premier niveau ont été cassées. L'écart-type élevé est dû à certaines parties atteignant des scores de plus de 600 points, obtenus grâce à la

4. Les résultats pour le joueur humain et le mode aléatoire sont tirés de [18]. Le temps d'entraînement pour le joueur humain est de 2 heures soit 432 000 images.

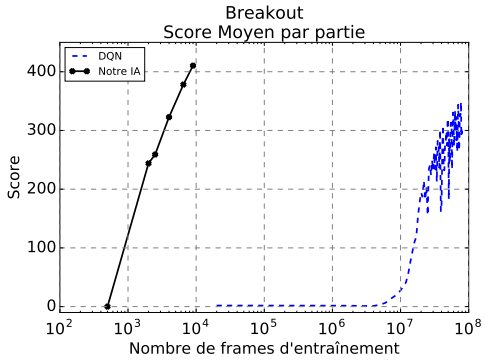

FIGURE 3 – Scores sur Breakout

capacité de l'IA à *réfuter* ses modèles.
Les résultats dans Pong ont été obtenus en utilisant trois approches : la première (a) identifie le « Moi » (mais pas la balle comme ami) en 500 images. La seconde (b) correspond à la procédure suivie pour Breakout : elle identifie le « Moi » puis les amis sur chacun des deux jeux en 10 000 images. La troisième (c) illustre le transfert de connaissances formalisé en utlisant les similarités décrites dans la partie 3.1 : l'IA commence par apprendre à jouer à Breakout en 10 000 images, puis elle identifie le « Moi » sur Pong en 500 images, et enfin elle transfère (sans images d'entraînement supplémentaires) la connaissance de l'ami vers Pong. L'expérience (c) permet d'obtenir un système capable de jouer à Breakout *et* Pong avec seulement 10 500 images d'entraînement comme mis en évidence en table 3.

6 Conclusion

À partir de concepts empruntés aux Sciences Sociales et Cognitives et d'enseignements tirés de caractéristiques propres aux méthodes scientifiques, ces travaux utilisent des outils élémentaires de la théorie des catégories pour faciliter l'établissement d'analogies et le transfert de connaissances entre situations non bijectives. Ces outils permettent aussi de formaliser une approche aboutissant à une IA capable de contrôler un agent dans un monde en

perpétuelle évolution.
Les résultats des expériences conduites sur un système cyber-physique mais aussi sur des jeux vidéo Atari 2600 confirment nos attentes. En effet, ce système a été capable d'identifier, puis de contrôler le « Moi » dans différentes situations, que ce soient des circuits de configurations différentes, ou des jeux Atari 2600 différents, en s'appuyant sur un transfert de connaissances et une faible quantité de données.

References

[1] P. Bartha, "Analogy and analogical reasoning," in *The Stanford Encyclopedia of Philosophy*, winter 2016 ed., E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2016.

[2] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[3] M. Bogdanovic, D. Markovikj, M. Denil, and N. de Freitas, "Deep apprenticeship learning for playing video games," in *AAAI Workshop on Learning for General Competency in Video Games*, 2015.

[4] C. Chang and H. J. Keisler, *Model Theory*, 3rd ed. North Holland, 1990.

[5] Z. Chen and B. Liu, *Lifelong Machine Learning*. Morgan & Claypool Publishers, 2016.

[6] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*, ser. Harper Perennial Modern Classics. Harper Collins, 2009.

[7] J. Culbertson and K. Sturtz, "Bayesian machine learning via category theory," *arXiv:1312.1445v1 [math.CT] 5 Dec 2013*, 2013.

[8] F. de Saussure, *Cours de linguistique générale*. Payot, 1916.

[9] B. Falkenhainer, K. Forbus, and D. Gentner, "The structure-mapping engine: Algorithm and examples," *Artificial Intelligence*, vol. 41, pp. 2–63, 1989/90.

[10] R. E. Kalman, "On the general theory of control systems," in *Proceedings of the First International Congress on Automatic Control*. Butterworth, London, 1960, pp. 481–492.

[11] K. Kansky, T. Silver, D. A. Mély, M. Eldawy, M. Lázaro-Gredilla, X. Lou, N. Dorfman, S. Sidor, D. S. Phoenix, and D. George, "Schema networks: Zero-shot transfer with a generative causal model of intuitive physics," in *ICML 2017*, 8 2017, pp. 1809–1818.

[12] S. Lange, M. Riedmiller, and A. Voigtlander, "Autonomous reinforcement learning on raw visual input data in a real world application," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2012, pp. 1–8.

[13] G. Lee, M. Luo, F. Zambetta, and X. Li, "Learning a Super Mario controller from examples of human play," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, July 2014, pp. 1–8.

[14] W. S. Levine, *The Control Systems Handbook: Control System Advanced Methods*, 2011.

[15] Y. Liang, M. C. Machado, E. Talvitie, and M. Bowling, "State of the art control of Atari games using shallow reinforcement learning," in *Proceedings of the 2016 ICAAMAS*, ser. AAMAS '16, 2016, pp. 485–493.

[16] N. Lipovetzky, M. Ramirez, and H. Geffner, "Classical planning with simulators: Results on the Atari video games," in *IJCAI'15*, 2015, pp. 1610–1616.

[17] S. Mac Lane, *Categories for the working mathematician*, 2nd ed. Springer-Verlag, 1998.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, February 2015.

[19] OpenCV, "Open source computer vision library," https://opencv.org/, 2017.

[20] J. Ortega, N. Shaker, J. Togelius, and G. N. Yannakakis, "Imitating human playing styles in Super Mario Bros," *Entertainment Computing*, vol. 4, no. 2, pp. 93 – 104, 2013.

[21] T. Pepels, M. H. M. Winands, and M. Lanctot, "Real-Time Monte-Carlo Tree Search in Ms. Pac-Man," *IEEE Trans. on Computational Intelligence and AI in Games*, vol. 6, no. 3, pp. 245–257, Sept 2014.

[22] J. Piaget, *The construction of reality in the child*. Basic Books, 1954.

[23] K. Popper, *The Logic of Scientific Discovery*. Hutchinson & Co, London, 1959.

[24] L. Pusman and K. Kosturik, "Control algorithm based on phase locked loop," in *Telecommunications Forum (TELFOR), 2013 21st*, Nov 2013, pp. 605–607.

[25] J. M. Schaeffer, *L'expérience esthétique*. Gallimard, 2015.

[26] A. Schutz, *The phenomenology of the social world*, ser. Northwestern University studies in phenomenology & existential philosophy. Northwestern University Press, 1967.

[27] D. Sperber, *La Contagion des idées*. Éditions Odile Jacob, 1996.

[28] D. I. Spivak, *Category Theory for the Sciences*. The MIT Press, 2014.

FIGURE 4 – Algorithme utilisé.

Require: $bdc = [entites, modeles, strategies]$
Require: $strategie\_par\_defaut$
Require: $seuil$, N_{max}
1: **for** $N_{init} = 0$ to N_{max} **do**
2: $entites$.append(donnees_prelevees)
3: $action \leftarrow strategie\_par\_defaut$
4: applique($action$)
5: **end for**
6: $jeu\_similaire \leftarrow$ recherche jeu avec des $entites$ similaires
7: $modeles, strategies \leftarrow$ extrait($bdc, jeu\_similaire$)
8: bdc.append($[entites, modeles, strategies]$)
9: $f\_refute\_modeles \leftarrow$ **false**
10: **if** $modeles$ est vide **then**
11: $f\_refute\_modeles \leftarrow$ **true**
12: **end if**
13: **while** experimentation en cours **do**
14: $entites$.append(donnees_prelevees)
15: **if** $f\_refute\_modeles$ **then**
16: $action \leftarrow strategie\_par\_defaut$
17: $modeles, strategies, f\_id \leftarrow$ identification($entites$)
18: **if** f_id **then**
19: $f\_refute\_modeles \leftarrow$ **false**
20: bdc.append($[entites, modeles, strategies]$)
21: **end if**
22: **else**
23: $action \leftarrow$ cherche_action($entites, strategies$)
24: $predictions \leftarrow$ cherche_predictions($entites, modeles$)
25: $erreur \leftarrow predictions - entites$
26: **if** $|erreur| > seuil$ **then**
27: $f\_refute\_modeles \leftarrow$ **true**
28: **end if**
29: **end if**
30: applique($action$)
31: **end while**
32: **return** bdc

Annexe : algorithme

L'algorithme utilisé est décrit en pseudo-code sur la figure 4. Il repose sur une base de connaissance bdc, une stratégie par défaut $strategie\_par\_defaut$, un seuil $seuil$ de rejet du modèle et un nombre d'itérations N_{max} correspondant à la durée de l'initialisation.

La base bdc, initialement vide, se remplit progressivement des stratégies obtenues au cours des parties jouées. La stratégie par défaut $strategie\_par\_defaut$ est, dams le cas du circuit de voitures, une MLI constante (39% dans nos expériences). Dans le cas des jeux Atari 2600, c'est une séquence pseudo-aléatoire d'actions envoyées en entrée du système.

L'algorithme commence par acquérir les échantillons puis les assimile à des types d'entités en ligne 2. À partir de ces entités, l'algorithme détermine la situation qui présente les objets les plus similaires en ligne 6 avant d'extraire en ligne 7 les modèles et stratégies obtenus lors de cette situation. La base bdc est ensuite complétée.

Lors de la partie, le système continue à acquérir des échantillons et à les transformer en entités. Si le modèle est valide (lignes 23 à 27), le système calcule à partir des entités présentes et stratégies disponibles la meilleure action à effectuer. Avec les entités et les modèles, une prédiction de l'état de l'environnement est aussi calculée. Si ces predictions sont trop éloignées de l'état mesuré, alors la variable f_refute_savoir devient vraie. Dans ce cas, dès l'itération suivante, la procédure d'identification (lignes 15 à 20) se lance. De nouveaux modèles et stratégies sont calculés. Cette procédure est mise en oeuvre tant que de nouveaux modèles et stratégies n'ont pas été trouvés puis validés. Une fois validés, la base de connaissances bdc est enrichie de ces résultats.

Notons que l'algorithme est identique en phase d'apprentissage et en phase d'exploitation (test). Il continue de réfuter son savoir, si besoin est, même en phase d'exploitation.