Apprentissage par Transfert reformulé via la Théorie des Catégories

Lionel Cordesses, Sarah Amar, Omar Bentahar, Aude Laurent, Kevin Poulet, Thomas Ehrmann, Ju Page

Apprentissage par Transfert reformulé via la Théorie des Catégories

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

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

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

Résumé

Les applications industrielles utilisant l'apprentissage automatique pour contrôler des systèmes embarqués doivent apprendre avec très peu d'essais. En effet, le nombre d'expériences possibles durant la phase d'apprentissage est souvent limité, par des contraintes de temps ou de budget. Cet objectif d'apprentissage rapide peut être atteint en utilisant du transfert de connaissances formalisé mathématiquement avec les outils de la théorie des catégories.
Nous présentons une approche originale de l'apprentissage automatique qui transmet ses connaissances à des situations nouvelles en s'appuyant sur les outils existants du domaine. Des résultats expérimentaux sont présentés sur un circuit de voitures miniatures.

Mots Clef

Apprentissage Automatique, Théorie des Catégories, Apprentissage par Transfert, Circuit de Voitures Miniatures.

Abstract

Industrial applications relying on Machine Learning (ML) to control robotic systems need to learn with few trials. The number of experiments in this context is limited by either the available time or budget. Such a goal can be attained by using knowledge transfer formalized mathematically with tools from category theory.
This leads to a new ML approach that transposes accumulated knowledge to new configurations, while still relying on existing ML tools. Illustrations of this innovative solution are presented using a slot car game.

Keywords

Machine Learning, Category Theory, Transfer Learning, Slotcar.

1 Introduction

La nécessité pour les algorithmes contrôlant des systèmes embarqués de devoir apprendre, avec peu de données, comment évoluer dans un environnement partiellement connu, est de plus en plus importante. Beaucoup d'approches utilisent pour cela l'apprentissage par renforcement et, plus particulièrement, les réseaux de neurones qui ont prouvé leur efficacité sur des systèmes embarqués.
Néanmoins, ces approches nécessitent beaucoup de données pour l'apprentissage, qui peuvent ne pas être disponibles en fonction de contraintes de temps ou de budget.
Pour illustrer ce besoin d'apprentissage automatique utilisant peu de données, nous utilisons le secteur automobile. Plus particulièrement, les voitures à transmission automatique proposent habituellement plusieurs modes (sport, neige, etc.) qui configurent les réglages de la transmission. Néanmoins, adapter ces réglages aux préférences de chaque conducteur reste un défi. Par exemple, considérons un algorithme ajustant les préférences du conducteur pour le mode neige. Si 100 essais sont nécessaires pour l'apprentissage, le mode neige étant utilisé 1 fois par an, il faudra un siècle pour avoir des résultats. Les algorithmes d'apprentissage automatique nécessitant des milliers d'essais ne sont donc pas envisageables dans ce cas.
L'idée de départ de ce projet était de proposer une approche alternative sous forme d'un algorithme aussi performant que les méthodes de l'état de l'art, mais capable d'apprendre avec environ 1% du temps, des données et de la puissance de calcul que demandent ces méthodes.
Nous présentons d'abord les publications utilisant les circuits de voitures miniatures comme support expérimental, la plupart portant sur le contrôle de ces systèmes. Ensuite, nous présentons la théorie de notre approche et les deux types de problèmes sur lesquels nous travaillons : le cas bijectif où le système imite un savoir élément par élément, et le cas utilisant la théorie des catégories où le système transpose un savoir au niveau des *types* d'éléments. Ces deux cas utilisent le transfert de connaissances comme dans le Lifelong Machine Learning [4]. Nous décrivons enfin le support expérimental et les résultats obtenus.

2 Travaux Antérieurs

En raison de leur facilité d'approche et de leur faible coût, les circuits de voitures miniatures (voir figure 1) sont utili-

sés dans de nombreux domaines, de la théorie du contrôle à l'Intelligence Artificielle (IA). Nous présentons ici un aperçu des utilisations de ce type de circuits dans des travaux pertinents pour notre projet, puis nous élargissons à des problèmes liés a l'apprentissage avec peu d'exemples.

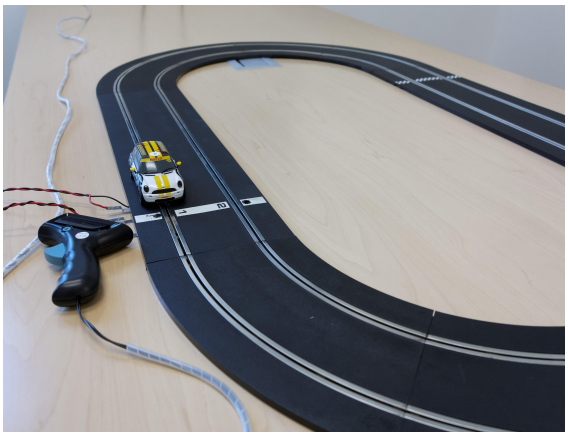

FIGURE 1 – Exemple de circuit de voitures miniatures

2.1 Outil pédagogique

Les circuits de voitures sont largement utilisés pour enseigner l'interfaçage et l'exploitation de capteurs [23] et d'actionneurs [5] avec un calculateur. Les signaux réels, contrairement aux signaux simulés, ajoutent un degré de difficulté comme le mentionne [22] : « les moteurs des voitures génèrent une quantité importante de bruit électrique » ; cette publication propose une solution concrète pour limiter ce problème. De plus, un tel système n'est pas linéaire indépendant du temps (LTI) puisque le circuit et les voitures nécessitent tous les deux de l'entretien pour maintenir les performances constantes : graissage et nettoyage internes pour la voiture, et nettoyage pour la piste. L'article met également l'accent sur les incertitudes dans les mesures des signaux, en particulier pour le capteur utilisé pour indiquer la position de la voiture. La détection des défauts des capteurs est aussi détaillée dans [10] dans le cadre de la conception et du test d'IA pilotant la voiture.
Plus généralement, les circuits de voitures sont utilisés pour enseigner les micro-contrôleurs dans [31], la mécatronique dans [14] et [11], les systèmes temps réel et l'IA dans [3].

2.2 Validation de théories du contrôle

Des codes barres imprimés sur les rails pour localiser la voiture et un micro-contrôleur embarqué pour ajuster la vitesse en conséquence peuvent être utilisés afin de contrôler de façon autonome la voiture [13]. Le contrôle de convois de véhicules, implanté sur un micro-contrôleur 32 bits, est validé dans [20] à l'aide de capteurs embarqués mesurant la vitesse et l'accération. Le développement et la validation de nouveaux concepts de « Mobilité en tant que service » sont évalués sur un circuit de voitures dans [28].
Les circuits de voitures sont également mentionnés dans [24] pour l'important couple électrique du moteur comparé au poids de la voiture. Cette caractéristique en fait un cas d'étude intéressant de lois de commande optimales limitant le couple pour protéger l'actionneur : le moteur.

2.3 Validation des algorithmes d'IA

La solution proposée par [15] contrôle la vitesse de la voiture grâce à un perceptron multicouche qui utilise un algorithme de traitement d'images pour localiser la position de la voiture. Le temps d'apprentissage de la stratégie de contrôle est de 45 minutes. La solution d'apprentissage par renforcement [17] repose sur un perceptron multicouche à convolution pour localiser la voiture, et sur un contrôle via quatre niveaux de tension. La phase d'entraînement du perceptron prend 12 heures, et celle de la stratégie de contrôle nécessite 30 minutes supplémentaires. Une autre méthode d'apprentissage par renforcement utilise un réseau de neurones pour contrôler la vitesse de la voiture : décrite dans [12], elle consiste à construire une représentation de l'état permettant à la voiture de rester sur la piste. Un algorithme évolutionniste est proposé dans [7] pour piloter la voiture grâce à une caméra neuromorphique.
Une solution plus rapide pour rendre autonome une telle voiture en quelques tours de circuit est décrite dans [27] et consiste à ajouter des accéléromètres et un micro-contrôleur embarqués pour cartographier le circuit (répartition des virages et des lignes droites). L'algorithme de contrôle est ensuite capable d'adapter la vitesse de la voiture grâce à une boucle à verrouillage de phase. Des algorithmes génétiques sont enfin comparés aux algorithmes d'apprentissage par renforcement dans [19] dans le but d'ajuster le régulateur PID (proportionnel, intégral, dérivé) embarqué pilotant la voiture.

2.4 Validation des algorithmes de prédictions

Les circuits de voitures sont également utilisés pour valider les algorithmes de prédictions basés sur des algorithmes d'apprentissage automatique comme les Modèles de Markov cachés dans [30] et [2], et les estimations de positions calculées grâce à des mesures inertielles utilisant des filtres de Bayes à processus Gaussien dans [16]. Cette dernière solution permet à la méthode de rejouer une stratégie apprise en observant un expert humain jouant sur 16 tours. Un autre estimateur exploitant des données inertielles pour piloter une voiture est proposé dans [21] et repose sur des modèles de mélanges gaussiens.

2.5 Apprentissage par imitation et analogies

En dehors du domaine du circuit de voitures, l'imitation d'un ou de plusieurs exemples est une solution au problème de l'apprentissage avec très peu de données. L'approche décrite dans [9] et utilisée dans un contexte de robotique manipulatrice consiste à transformer des exemples fournis par un humain en un programme compréhensible tant par l'humain que par la machine. Ce programme est réutilisé dans des cas similaires à ceux des exemples. La notion d'analogie est aussi exploitée en classification et dans le

contrôle d'agents [26], par exemple dans des jeux de stratégie en temps réel [29].

3 Contrôler une voiture électrique par apprentissage automatique

3.1 Objectifs

Bien que les méthodes présentées en partie 2 parviennent à contrôler avec succès une voiture électrique miniature sur un circuit inconnu, ces méthodes nécessitent un grand nombre de tours d'apprentissage, ou alors compensent une durée d'apprentissage réduite par l'utilisation de capteurs embarqués, tels que des accéléromètres. Notre objectif est d'aboutir à une méthode d'apprentissage automatique qui pourrait apprendre avec peu de ressources, en particulier sans l'ajout de capteurs supplémentaires, et qui parviendrait à contrôler la voiture en temps réel avec une fraction de la puissance de calcul auparavant requise. Ces conditions aboutissent à une approche d'apprentissage automatique qui est entraînée sur quelques tours, en n'utilisant comme données que les courants et tensions mesurés sur la piste, et qui fonctionne par exemple sur un microcontrôleur 8 bits d'Arduino Mega cadencé à 16 MHz.
À notre connaissance, de tels prérequis ne sont pas vérifiés par les méthodes existantes. C'est pourquoi nous nous sommes lancés dans la conception d'une nouvelle approche d'apprentissage automatique, en s'inspirant de concepts philosophiques, puis en la formalisant rigoureusement avec les outils mathématiques de la théorie des catégories.

3.2 Solution inspirée de concepts philosophiques

Notre approche travaille à l'échelle des entités, définies cidessous, et non à celle des échantillons. Travailler à cette échelle peut s'apparenter à travailler à l'échelle du morphème en linguistique, *i.e.* le plus petit élément significatif d'un langage, conformément à la définition de [6]. Cela comprend une analyse des signaux échantillonnés pour détecter les entités. Celles-ci sont nos objets du quotidien : les tronçons du circuit (lignes droites et courbes du circuit, zones de danger pour l'algorithme) et la voiture elle-même. Comme tout objet cognitif, elles disposent de certaines propriétés : cohérence relative, continuité de mouvement, permanence de l'existence.
Toutefois, le monde qui nous entoure n'est pas seulement constitué d'objets : il comporte aussi des sujets. En particulier, dans les problèmes considérés, il existe une représentation de ce que l'on appelle le « Moi », c'est-à-dire l'entité contrôlée par nos actions. Dans le cas spécifique du circuit de voitures, trouver ce « Moi » n'est pas nécessaire, puisque l'IA contrôle une seule entité : la voiture, donc le « Moi ».
De plus, les modèles obtenus ne sont pas éternellement valides, puisque le « Moi » et l'environnement constitué des autres entités peuvent évoluer. Afin que notre approche continue de fonctionner dans un environnement en évolution, nous nous sommes inspirés de l'approche scientifique décrite dans [25] : « un système faisant partie de la science empirique doit pouvoir être réfuté par l'expérience ». Nous concevons notre logiciel de sorte qu'il puisse remettre en cause ses propres résultats, puis reconstruire ses modèles. Cette remise en cause se produit dès que l'erreur entre les mesures du système et les prédictions du modèle est supérieure à un seuil.
Enfin, en lien avec le concept heideggerien de *Dasein*, littéralement *être-là* [8], un humain confronté à la conscience de soi et de la mort percevra ce « Moi » comme plus qu'un simple système contrôlable, et le dotera ainsi d'un instinct de survie. Nous modélisons ce « Moi » à partir d'une philosophie similaire : au fur et à mesure qu'il expérimente le monde qui l'entoure, ce « Moi » classe les entités environnantes comme amies ou ennemies, selon les bénéfices ou pertes qu'elles occasionnent. Une stratégie basique est ensuite appliquée, à savoir transférer les comportements qui n'étaient pas dangereux lors de situations précédentes.
En résumé, notre algorithme prend en compte les récompenses qu'il tire de son environnement afin de classifier les entités, comme en apprentissage par renforcement. Ensuite, par analogie avec les méthodes scientifiques empiriques, cette classification et les modèles sont remis à jour lorsque les prédictions ne correspondent plus aux mesures. Enfin, le « Moi » est contrôlé dans le but de survivre, conformément à l'instinct de survie dont un joueur le dote.
Nous présentons dans la section suivante comment formaliser mathématiquement une telle approche d'apprentissage automatique, en particulier dans le but de transposer des connaissances acquises à une situation inconnue.

4 Utilisation d'analogies pour le transfert de connaissances

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 comment conduire une voiture électrique miniature sur un circuit, sera capable de transposer ces capacités à une autre configuration, même si elle est de forme ou de taille différente.
Les situations où deux problèmes ont exactement le même nombre d'états et des structures isomorphes sont rares. Néanmoins, il existe des outils mathématiques pour identifier des structures non isomorphes tels que l'équivalence de catégories en théorie des catégories [18]. La théorie des catégories est un outil mathématique puissant apparu au milieu du XX$^{\text{ème}}$ siècle en lien avec la topologie algébrique, et développé dans le but de transférer des concepts et des théorèmes entre différentes branches des mathématiques.
Dans l'esprit de 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 au travers de l'introduction de flèches (ou morphismes) entre objets, ce qui permet de nouvelles sortes

d'identifications. Comme la plupart des outils d'apprentissage automatique s'appuient sur la théorie des ensembles et non sur la théorie des catégories, nous illustrons ci-dessous la plus-value d'un tel cadre mathématique.

4.1 Analogies entre deux situations

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 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 $\mathcal{C}$. 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 $\mathcal{C}$: $\forall x, y \in \mathcal{C}, x\ R_f\ y$ si et seulement si (ssi) $f(x) = f(y)$. Transposé dans le champ lexical de la théorie 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 ensembles 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. 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.

L'utilisation de la théorie des catégories permet de formaliser tout un éventail de situations. Nous choisissons de l'illustrer sur un circuit de voitures miniatures.

4.2 Analogies entre configurations d'un circuit de voitures

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 $\mathcal{C}'$.

Dans la configuration $\mathcal{C}'$, le joueur peut agir sur $(u', i')'_s$ en utilisant la manette de jeu, ce qui correspond à la stratégie π' définie par (1).

$$\pi'(s') = \begin{cases} (u', i')_{k'}, & \text{si } s' \text{ est une ligne droite} \\ (u', i')_{l'}, & \text{sinon} \end{cases} \quad (1)$$

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 (2).

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

Il est facile de voir, si les définitions exactes sont connues, que (2) 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.

5 Résultats Expérimentaux

5.1 Matériel

Le support 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 à sortie binaire. 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 décrit en annexe, puis échantillonnés à $f_s = 100\,\text{Hz}$. Le système 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 $k \in \mathbb{N}$.

5.2 Implantation dans le cas bijectif

Le cas bijectif mentionné en 4.2 correspond au cas où $N = N'$ et $s_0 = s'_0$ et repose sur une procédure d'imitation en trois étapes. Un joueur humain commence par conduire la voiture sur n tours, avec $n = 3$ dans nos expériences. Les K valeurs de tensions $v(kt_s)$ et de courants $i(kt_s)$ $(0 \leq k < K)$ échantillonnées sur le tour le plus rapide, ainsi que le temps du tour associé t_{best}, sont sauvegardés dans la RAM, pour être ensuite rejoués par l'IA. Une méthode d'optimisation (Newton) minimise la différence entre le temps de tour de l'IA et t_{best} en ajustant la commande par modulation en largeur d'impulsion (MLI) basée sur les échantillons enregistrés $v(kt_s)$.

5.3 Implantation dans le cas basé sur la théorie des catégories

Ce cas, inspiré 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.
Comme en apprentissage par renforcement, notre approche s'appuie sur une récompense fournie par l'environnement. Celle-ci est le résultat de la combinaison de trois variables. La première variable est le temps du tour, mesuré directement par le compteur de tours. La seconde variable binaire reflète la présence de la voiture sur la piste, et la dernière variable binaire indique si la voiture se déplace ou pas. Ce module surveille constamment la voiture pour vérifier qu'elle n'est pas sortie de la piste à cause d'une vitesse trop élevée, ou 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. Le classificateur est entraîné sur 1301 échantillons avec $k = 3$. Rappelons que les données sont échantillonnées à une fréquence de $100\,\text{Hz}$: l'entraînement utilise donc $13\,\text{s}$ de jeu. Une version réduite des données du classificateur est utilisée afin d'être exécutable en temps réel sur l'Arduino.
En utilisant ce module de récompense, l'IA peut piloter avec succès 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 il 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 les deux configurations : le cas bijectif ne peut pas s'appliquer. L'IA s'appuie sur des analogies et transpose les connaissances acquises dans une première configuration en utilisant l'équation (1), conformément au formalisme décrit en Partie 4. La fonction $h((u,i)_s)$ est évaluée par un k-NN entraîné sur 2327 échantillons (u,i) avec $k = 5$ et implémenté de la même façon que le premier classificateur.
En pratique, l'approche par analogies se déroule 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 gardant l'objectif de minimiser le temps de 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 2 sont de forme et de taille différentes, et une simple reproduction d'une stratégie ou d'une commande préalablement enregistrée conduirait rapidement à une sortie de route.

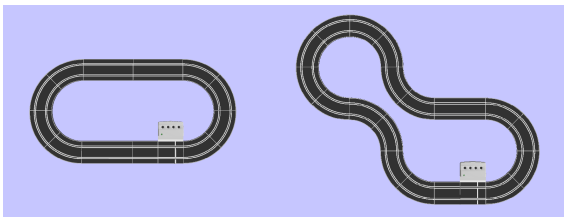

FIGURE 2 – Configuration des circuits : circuit 1 (à gauche), circuit 2 (à droite)

5.4 Résultats

Les expériences décrites dans cet article ont été menées sur deux configurations de complexité différente présentées en figure 2. La première est un anneau de 12 tronçons alors que la deuxième dispose de 18 tronçons. Les temps au tour sont décrits dans la table 1 pour le cas d'application de la théorie des catégories et dans le tableau 4 pour le cas bijectif. Ils sont donnés sous la forme (moyenne $\pm$ écart-type),

TABLE 1 – Temps de tour en secondes pour le cas d'application de la théorie des catégories (moyenne ± écart-type)

	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)

sauf pour le meilleur tour humain qui est le meilleur temps obtenu parmi les 7 participants.

Impact de l'entretien. La voiture et le circuit sont régulièrement entretenus, comme stipulé en partie 2.1, afin que le système soit presque LTI et les résultats reproductibles. L'impact de cet entretien peut être estimé sur la figure 3. La non-invariance temporelle est aussi visible, puisque le temps au tour augmente dès que l'on dépasse la cinquantaine de tours parcourus, malgré une MLI constante.

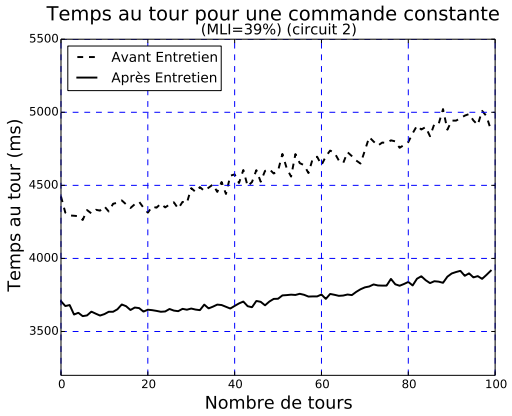

FIGURE 3 – Le système ne présente pas d'invariance temporelle

Évaluation des classificateurs. Les matrices de confusion des algorithmes de classification utilisés sont données dans le tableau 2 pour l'estimateur du signal d'arrêt de la voiture et dans le tableau 3 pour la fonction $h((u,i)_s)$, qui détermine la forme du tronçon sur lequel la voiture se trouve. Dans un souci de concision, l'estimateur de sortie de piste n'est pas analysé dans ce document, car il est similaire à celui du signal d'arrêt.

TABLE 2 – Matrice de confusion de l'estimateur "Voiture arrêtée" sur 641 échantillons (u,i)

		CLASSE ESTIMÉE	
		Arrêt	Mouvement
CLASSE RÉELLE	Arrêt	**171**	1
	Mouvement	4	**465**

TABLE 3 – Matrice de confusion de l'estimateur $h((u,i)_s)$ sur 287 échantillons (u,i)

		CLASSE ESTIMÉE	
		Droite	Courbe
CLASSE RÉELLE	Droite	**51**	5
	Courbe	1	**230**

TABLE 4 – Temps au tour en secondes pour le cas bijectif

	MEILLEUR HUMAIN	**IA**
CIRCUIT 1	2.11	$\mathbf{2.12 \pm 0.05}$
CIRCUIT 2	2.67	$\mathbf{2.65 \pm 0.02}$

Résultats dans le cas bijectif. Dans le cas bijectif, l'algorithme converge en moins de 5 tours de piste. Un exemple de cette convergence est représenté figure 4. Si le tour reproduit est le meilleur tour humain, cette approche aboutit au meilleur temps au tour. Néanmoins, cet algorithme ne peut être appliqué que si l'humain et l'algorithme pilotent sur des circuits identiques, contrairement à la solution qui s'appuie sur des analogies. En particulier, bien que le meilleur tour imité (bijectif) se fasse en 2.65 s sur le circuit 2 contre 3.13 s dans le cas où la voiture est pilotée par une approche basée sur l'établissement d'analogies, ce temps et ces vitesses ne sont viables que sur le circuit 2 et ne peuvent aboutir à une quelconque généralisation. Toutefois, ce temps est une borne inférieure du temps atteignable par un humain dans une configuration donnée.

Résultats dans le cas d'utilisation des analogies catégoriques. L'IA démarre à vitesse constante (tour 1 sur la figure 5), importée depuis la configuration précédente. À partir des mesures de courant et de tension, le k-NN détecte les tronçons du circuits qui ne présentent aucun danger. La MLI est ensuite augmentée dans le dernier de ces tronçons dès le deuxième tour. Pour aboutir à une amélioration moyenne du temps au tour due à cette accélération, le procédé est répété sur trois tours (tours 2, 3 and 4), car il n'y a pas d'invariance temporelle. Enfin, l'IA accélère dans le deuxième tronçon qui ne présente aucun danger sur les tours 5, 6 et 7 pour aboutir à un temps au tour final.

Dans les cas où le nouveau temps au tour est plus élevé

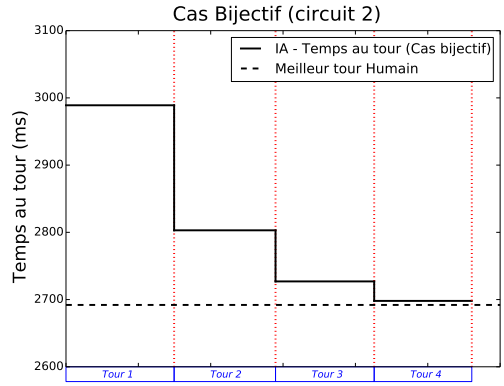

FIGURE 4 – Exemple de Cas Bijectif

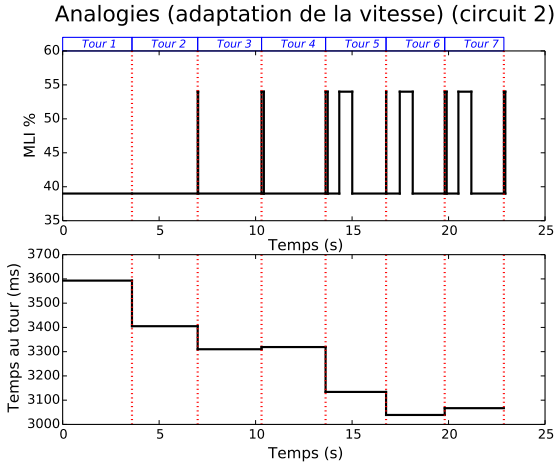

FIGURE 5 – Un exemple d'utilisation des analogies

que celui attendu, l'IA remet en cause ses connaissances, comme décrit en partie 3.2, en adoptant par exemple une vitesse plus élevée au travers de la MLI.

Cette IA, qui repose sur l'établissement d'analogies entre configurations et ne rejoue pas d'enregistrement d'une partie précédente, est capable d'améliorer les temps au tour en moins de 10 tours sur une piste inconnue. Sur le circuit le plus compliqué, l'IA atteint des temps similaires aux temps humains : 3.08 s pour les derniers tours des joueurs humains (*a fortiori* les meilleurs, car bénéficiant des stratégies mises en place lors des tours précédents) contre 3.13 s en moyenne pour l'algorithme sur le circuit 2.

Les améliorations futures de l'algorithme sur un circuit inconnu incluront l'optimisation des vitesses transposées par la fonction $h((u,i)_s)$. En effet, seule une vitesse de sécurité a été utilisée ici afin d'éviter les sorties de piste de la voiture pilotée par l'algorithme, alors que certains tours humains ont occasionné des sorties de piste et n'ont donc pas été comptabilisés.

Pour résumer les travaux sur le circuit de voitures, le cadre théorique détaillé en section 4 permet au système d'être au niveau des meilleurs joueurs humains sur ce jeu, le tout en moins d'une minute, et 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.

6 Conclusion

Nous avons illustré l'utilisation de concepts élémentaires de la théorie des catégories dans un contexte d'apprentissage automatique pour un système embarqué : un circuit de voitures miniatures. Aucun capteur n'a été ajouté à la voiture. Les seules informations utilisées sont le courant et la tension du circuit, ainsi qu'un compteur de tours. Nous montrons comment transférer la connaissance acquise d'une situation à une autre lorsqu'il n'y a pas de bijection entre elles, en utilisant les analogies définies grâce à cet outil mathématique.

Les résultats expérimentaux prouvent qu'il est simple de mettre en œuvre un algorithme d'apprentissage automatique basé sur la théorie des catégories, y compris sur un calculateur à faible puissance de calcul. Cette théorie permet de transférer les connaissances à des situations jamais rencontrées auparavant, tout en utilisant des outils classiques et connus, ce qui permet à l'IA d'apprendre avec peu de données.

Annexe 1 : filtre en échelle RC

Bien que les filtres actifs soient omniprésents de nos jours, l'approche de « conception par les coûts », usuelle dans des industries de production en grandes série, mène au choix d'un filtre anti-repliement passif en échelle utilisant uniquement des résisteurs (R) et des condensateurs (C). Notons que si les filtres passifs en échelles composés de solénoïdes et de condensateurs sont classiques, les versions sans solénoïdes sont assez peu usuelles, d'où les précisions apportées par cette annexe.

Le filtre a une fréquence de coupure f_c, et sa fonction de transfert est $G(s) = 1/(sRC+1)^2$ où s est la variable de Laplace. Elle est approximée par un double réseau RC de Cauer décrit dans [1] dont les valeurs de R et C du premier réseau RC vérifient $f_c = 1/(2\pi RC)$. Les valeurs du second circuit RC, R/d et Cd avec $\{d \in \mathbb{R} : d > 0\}$, sont calculées pour vérifier une condition sur l'erreur maximale sur le gain $e(d)$ entre $G(s)$ et $G_a(s)$, la fonction de transfert du réseau de Cauer définie par $G_a(s) = 1/\left((sRC)^2 + s(d+2)RC + 1\right)$. La fonction $e(d)$ est définie par l'équation (3). Nous choisissons pour notre application $d = 0.1$, ce qui conduit à une erreur inférieure à 0.5 dB, sans répercution sur les calculs qui en dépendent. Une application avec deux réseaux RC identiques donnerait $e(1) = 3.5\,\mathrm{dB}$, ce qui diminuerait la performance globale du système. La tension et le courant sont tous les deux filtrés grâce à de tels filtres RC en échelle.

$$e(d) = 20\log_{10}\left(\frac{d+2}{2}\right) \tag{3}$$

References

[1] N. Balabanian, *Network Synthesis*. Prentice-Hall Electrical Engineering Series, 1958.

[2] B. Boots and G. J. Gordon, "An Online Spectral Learning Algorithm for Partially Observable Nonlinear Dynamical Systems," in *AAAI Conference on Artificial Intelligence*, August 2011, pp. 293–300.

[3] M. Brejl and J. Necesany, "Student's contest: Self-driven slot car racing," in *IMCSIT 2008*, Oct 2008, pp. 589–592.

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

[5] D. R. Cheriton, M. A. Malcolm, L. S. Melen, and G. R. Sager, "Thoth, a Portable Real-time Operating System," *Commun. ACM*, vol. 22, no. 2, pp. 105–115, Feb 1979.

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

[7] T. Delbruck, M. Pfeiffer, R. Juston, G. Orchard, E. Muggler, A. Linares-Barranco, and M. W. Tilden, "Human vs. computer slot car racing using an event and frame-based DAVIS vision sensor," in *IEEE ISCAS*, May 2015, pp. 2409–2412.

[8] H. Dreyfus, *Being-in-the-world: A commentary of the Heidegger's Being and Time, Division I*. MIT Press, 1990.

[9] A. Feniello, H. Dang, and S. Birchfield, "Program synthesis by examples for object repositioning tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 4428–4435.

[10] J. A. Fulcher, "Fun and games and microcomputer interfacing (laboratory exercises)," *IEEE Micro*, vol. 11, no. 1, pp. 18–21, Feb 1991.

[11] P. Gober, "Experiences with a Project to Design Autonomous Slotcars in a Mechatronics Master's Program," in *Proceedings of the WESE'15: Workshop on Embedded and Cyber-Physical Systems Education*. ACM, Oct 2015, pp. 5:1–5:3.

[12] R. Jonschkowski and O. Brock, "State representation learning in robotics: Using prior knowledge about physical interaction," in *Proceedings of Robotics: Science and Systems*, July 2014.

[13] S. Kane and J. B. Scott, "The slot car stig: Performance and consistency of a slot car driven by a heuristic algorithm in an embedded microcontroller," in *The 16th Electronics New Zealand Conference*, Dunedin, New Zealand, Nov 2009, pp. 177–180.

[14] V. Kapila and S.-H. Lee, "Science and mechatronics-aided research for teachers," *IEEE Control Systems*, vol. 24, no. 5, pp. 24–30, Oct 2004.

[15] T. C. Kietzmann and M. Riedmiller, "The Neuro Slot Car Racer: Reinforcement Learning in a Real World Setting," in *ICMLA '09*, Dec 2009, pp. 311–316.

[16] J. Ko and D. Fox, "Learning GP-Bayes Filters via Gaussian process latent variable models," *Autonomous Robots*, vol. 30, no. 1, pp. 3–23, Jan 2011.

[17] S. Lange, M. Riedmiller, and A. Voigtlander, "Autonomous reinforcement learning on raw visual input data in a real world application," in *IJCNN 2012*. IEEE, 2012, pp. 1–8.

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

[19] D. Martinec and M. Bundzel, "Evolutionary algorithms and reinforcement learning in experiments with slot cars," in *International Conference on Process Control*, June 2013, pp. 159–162.

[20] D. Martinec, M. Sebek, and Z. Hurak, "Vehicular platooning experiments with racing slot cars," in *CCA 2012*, Oct 2012, pp. 166–171.

[21] L. McCalman, S. O'Callaghan, and F. Ramos, "Multi-modal estimation with kernel embeddings for learning motion models," in *ICRA 2013*, May 2013, pp. 2845–2852.

[22] P. J. McKerrow, "Micro-computers, slotcars and education," University of Wollongong, Australia, Working Paper 82-7, 1982.

[23] A. T. Mogill, "Calibration and use of a small motor as a slot car speedometer." *Physics Teacher*, vol. 13, no. 5, pp. 304–305, 1975.

[24] J. B. Moore and B. D. O. Anderson, "Optimal linear control systems with input derivative constraints," *Electrical Engineers, Proceedings of the Institution of*, vol. 114, no. 12, pp. 1987–1990, Dec 1967.

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

[26] H. Prade and G. Richard, "Analogical proportions and analogical reasoning – an introduction," in *Case-Based Reasoning Research and Development*, D. W. Aha and J. Lieber, Eds. Springer, 2017, pp. 16–32.

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

[28] D. Richter, A. Grapentin, and A. Polze, "Mobility-as-a-service: A Distributed Real-Time Simulation with Carrera Slot-Cars," in *IEEE 18th International Symposium on Real-Time Distributed Computing*, Apr 2015, pp. 276–279.

[29] G. Robertson, "Applying learning by observation and case-based reasoning to improve commercial RTS game AI," in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.

[30] L. Song, B. Boots, S. M. Siddiqi, G. J. Gordon, and A. J. Smola, "Hilbert Space Embeddings of Hidden Markov Models," in *ICML*, June 2010, pp. 991–998.

[31] B. Sprunt, "A Novel Race Track Platform For Teaching Microcontroller System Design Concepts," in *Proc. of the American Society for Engineering Education Annual Conference*, June 2003.

FIGURE 6 – Algorithme utilisé.

Require: $bdc = [entites, modeles, strategies]$
Require: $strategie\_par\_defaut = MLI\_constante$
Require: $seuil$, N_{max}
1: **for** $N_{init} = 0$ to N_{max} **do**
2: $entites$.append(k-NN(u,i))
3: $action \leftarrow strategie\_par\_defaut$
4: applique_MLI($action$)
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: $f\_refute\_savoir \leftarrow$ **false**
10: **if** $modeles$ est vide **then**
11: $modeles \leftarrow$ identification_parametrique($entites$)
12: **end if**
13: **while** experimentation en cours **do**
14: $entites$.append(k-NN(u,i))
15: **if** f_refute_savoir **then**
16: $jeu\_similaire \leftarrow$ cherche jeu avec des $entites$ similaires
17: $modeles, strategies \leftarrow$ extrait($bdc, jeu\_similaire$)
18: $f\_refute\_savoir \leftarrow$ **false**
19: bdc.append($[entites, modeles, strategies]$)
20: **else**
21: $action \leftarrow$ cherche_action($entites, strategies$)
22: $predictions \leftarrow$ cherche_predictions($entites, modeles$)
23: $erreur \leftarrow predictions - entites$
24: **if** $|erreur| > seuil$ **then**
25: $f\_refute\_savoir \leftarrow$ **true**
26: **end if**
27: **end if**
28: applique_MLI($action$)
29: **end while**
30: **return** bdc

Annexe 2 : algorithme

L'algorithme utilisé sur le circuit de voitures est décrit en pseudo-code sur la figure 6. 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} pour le ou les premiers tours du circuit.

La base bdc est initialisée avec une stratégie obtenue en analysant une précédente course lorsqu'un humain pilotait le véhicule sur un autre circuit : une MLI par type de tronçon. La stratégie par défaut $strategie\_par\_defaut$ est une MLI constante, par exemple la plus petite MLI mesurée sur les courses précédentes et qui permet à la voiture d'avancer (39% dans nos expériences). La valeur de N_{max} correspond à 3 tours de circuit.

L'algorithme commence par acquérir la tension u et le courant i en ligne 2, puis assimile la paire (u,i) à un type d'entité (courbe ou droite) via le k-NN. Lors de la première utilisation, il n'existe aucun modèle dans la base bdc : un modèle affine donnant le temps au tour en fonction de la MLI est ensuite calculé en ligne 11. Si la base bdc contient des informations issues d'une précédente course sur un circuit, alors la stratégie de la course la plus similaire à la configuration actuelle est transférée. Ici, la stratégie se résume à une MLI par type de tronçon.

Lorsque la voiture avance sur le circuit, le k-NN transforme à nouveau tension et courant en entités en ligne 14. Cette entité permet, à partir de la stratégie, de choisir une des deux actions suivantes : MLI pour le tronçon droit ou MLI pour le tronçon courbe. Une prédiction du temps au tour est calculée en ligne 22. Si l'erreur sur cette prédiction est supérieure à $seuil$ ($50\,\mathrm{ms}$), la variable f_refute_savoir devient vraie en ligne 25. À l'itération suivante en ligne 16, une nouvelle stratégie, voire un nouveau modèle seront calculés à partir de la base bdc puis appliqués et testés, et ainsi de suite jusqu'à la fin de l'expérience. La base de connaissance bdc est alors retournée et pourra être réutilisée sur un circuit de configuration différente.

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.