

# Une contribution à l'apprentissage par renforcement; application du Computer - Go

*Sylvain Gelly a préparé sa thèse au sein du projet TAO au Laboratoire de Recherche en Informatique à Orsay sous la direction de Michèle Sebag et Nicolas Bredeche.*



L'apprentissage automatique est un domaine de l'informatique qui offre une approche différente de la programmation traditionnelle. Au lieu d'écrire un programme, on fournit à la machine des exemples résolus et la machine apprend à résoudre ce problème. La programmation par apprentissage est particulièrement appropriée quand on ne sait pas résoudre le problème algorithmiquement ou quand l'environnement est ouvert.

Cette thèse se situe plus particulièrement dans le domaine de l'Apprentissage par Renforcement (AR) qui est à l'interface entre la théorie du contrôle, l'apprentissage supervisé et non-supervisé, l'optimisation et les sciences cognitives. Elle comporte trois axes principaux correspondant à trois verrous de l'AR.

La première difficulté est le passage à l'échelle des méthodes. Le cas d'un espace d'actions ou d'états qui est continu ou fini mais en grande dimension demande des approches et stratégies spécifiques. Dans ce

contexte, il est possible de tirer avantage de la structure du problème, par exemple en utilisant des Processus De Markov (PDM) factorisés, et en représentant le PDM comme un Réseau Bayésien Dynamique. Notre contribution théorique est basée sur une nouvelle borne des nombres de couverture de l'espace des RBs, incluant l'entropie structurelle du réseau, en plus du classique nombre de paramètres. Les contributions algorithmiques concernent l'optimisation paramétrique et non paramétrique du critère d'apprentissage basé sur cette borne, dont les mérites sont démontrés empiriquement sur des problèmes artificiels.

L'interaction entre l'apprentissage et la décision, ainsi que l'impact de l'approximation de l'apprentissage sur l'optimalité de la décision constituent un autre challenge dans l'AR. La question est de savoir si l'accent doit être mis plutôt sur l'approximation (par exemple en utilisant les meilleurs outils de l'apprentissage automatique), ou sur l'optimisation. La difficulté réside dans le fait que ces étapes ne sont pas indépendantes. Cette difficulté est étudiée dans la partie de thèse dédiée à la programmation dynamique stochastique en espaces d'états et d'actions continus, se concentrant sur l'optimisation non linéaire, l'apprentissage par régression et l'échantillonnage. Ces études sont toutes conduites dans notre plateforme OpenDP, un outil pour comparer les algorithmes sur

différents problèmes et diffusé sous une licence open-source.

La dernière partie de la thèse est consacrée au domaine du «Computer-Go». Depuis la victoire de Deep Blue contre Kasparov aux échecs en 1997, le jeu de Go est devenu le nouveau défi. La complexité vient de la taille du problème (des centaines de coups possibles) et du fait qu'on n'a pas d'algorithme pour savoir si une position est bonne ou mauvaise. En d'autres termes, le jeu de Go illustre un problème important et difficile : savoir raisonner et prendre des décisions dans un environnement incertain. L'approche défrichée a conduit au programme Mogo, médaille d'or aux 12ème olympiades informatiques et premier programme battant un joueur humain professionnel (en 9 x 9 pour l'instant). Il s'agit d'une avancée importante d'autant plus intéressante que Mogo n'utilise que très peu de connaissances relatives au Go. Le progrès réalisé repose ainsi sur des avancées générales, qui ne sont pas limitées au jeu de Go.

Ces dernières années ont vu l'émergence de techniques originales et performantes pour résoudre les deux problématiques du jeu de Go : l'évaluation de la position et la recherche arborescente. Pour la première, les techniques de Monte-Carlo utilisent, à partir d'une position, une politique de simulation (un joueur automatique et très rapide) terminant la partie en jouant contre elle même, et comptent sim-

plement les points dans la position finale. Nous avons alors introduit une nouvelle politique de simulation permettant d'évaluer bien plus précisément les positions, tout en restant très rapide à exécuter. Nous avons d'autre part montré que la qualité d'une telle politique pour l'évaluation d'une position n'était pas directement liée à sa performance en tant que joueur. Pour la recherche arborescente, nous avons d'abord adapté au jeu de Go un algorithme développé pour résoudre le compromis Exploration vs Exploitation (compromis qui a une place centrale en AR) : l'algorithme UCT. Nous l'avons ensuite étendu pour pouvoir généraliser efficacement les informations provenant des simulations de Monte-Carlo,

et pouvoir combiner des données a priori sur la position aux résultats statistiques provenant des simulations. Ce nouvel algorithme a permis d'améliorer considérablement l'efficacité du programme à la fois sur les «petites» tailles de plateau de jeu (9x9), et dans la taille standard (19x19), comme l'ont démontré les résultats de Mogo dans de nombreuses compétitions internationales.

Ces différentes avancées ouvrent des perspectives d'application dans d'autres domaines que le «Computer-Go». En effet, beaucoup de problèmes peuvent être vus comme la recherche de décision optimale en environnement incertain comme la gestion de l'énergie ou la robotique.