Implémentation de l'algorithme IAC : Intelligent Adaptive Curiosity

Jean-Baptiste ASSOUAD
Polytech'Paris UPMC

Artemis MUCAJ
Polytech'Paris UPMC

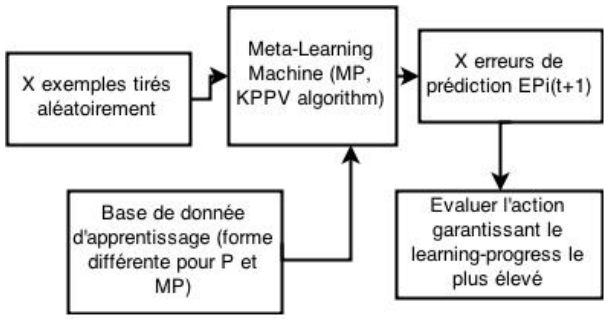
Implémentation de l'algorithme IAC (intelligent adaptive curiosity), le but étant de pousser un robot mobile a éffectuer des action qui maximisent son apprentissage. Le contexte est le suivant, nous disposons d'un robot mobile, la vitesse de chaque roue peut être controllée (réel entre -1 et 1), le robot est également capable d'émettre un son d'une fréquence comprise entre 0 et 1. L'espace d'action est donc de dimension 3 et continu [vitesse gauche, vitesse droite, fréquence]. Nous utilisons Python ainsi que le simulateur V-REP, le robot mobile est un ePuck.

Environnement de simulation

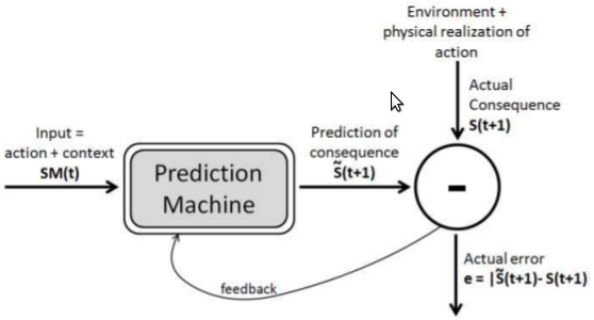
Un robot ePuck, un cube (représentant le jouet) ainsi que des murs sont placés sur la carte. Le robot ePuck émets une fréquence comprise entre 0 et 1, le comportement du jouet est différent en fonction de la fréquence émise par le robot. L'algorithme IAC permettra au robot d'apprendre cette plage de fréquences. Lorsque f=[0;0.33], le jouet se déplace de manière aléatoire sur la carte, f=[0.34;0.66], le jouet reste immobile et f=[0.67;1] le jouet sautes sur le robot. Il est important de noter que le robot percoit sa distance au jouet (il est muni d'un capteur de jouet!) et essayes de prédire la position du jouet après avoir réaliseé une certaine action, le robot choisis l'action qui lui assure un apprentissage maximal.

Algorithme

On peux décomposer l'algorithme en plusieurs sous parties, nous disponsons d'une meta machine de prédiction MP qui apprends a prédire l'erreur de prédiction d'une machine de prédiction P, capable d'évaluer la prochaine position du jouet a partir de l'action apportant un progrès d'apprentissage maximal (nous avons utilisé l'algorithme d'apprentissage KPPV, k-plus proches voisins, pour représenter ces learning-machines). P possèdes une liste d'erreures LE = E(0), E(1), E(2), E(3), ..., E(t), ainsi qu'une liste de moyennes d'erreurs LEm = Em(DELAY),Em(DELAY+1), Em(DELAY+2), ..., Em(t).



La base de donnée de MP est composée de vecteurs contenant les données suivantes [leftSpeed, rightSpeed, frequency, error]. La BDD de P est composée des données suivantes [leftSpeed, rightSpeed, frequency, distance-to-toy]. Ces données sont appelés espace sensori-moteur dans la suite de ce document.



La learning-machine P est composée de plusieurs experts, chacun spécialisé dans une partie de l'espace sensori-moteur, on peux donc représenter la base de donnée P comme un arbre, dont les feuille sont les différents experts. Lorsqu'il est nécessaire de prédire la position du jouet suite a une action, il est donc nécessaire de faire appel a l'éxpert spécialisé dans l'espace sensori-moteur défini par l'action. Pour chacun des X exemples tirés alátoirement, MP nous fournis une erreur de prédiction Ep, nous pouvons calculer le facteur d'apprentissage (learning-progress) de la manière suivante : LP(t) = -(Em(t) - Em(t-DELAY)), Em(t) étant la moyenne des DELAY dernières erreures (Ep comprise). L'action ayant le learning-progress le plus élevé est gardée en mémoire. L'action est réalisée par le simulateur puis P est ensuite executée avec l'action optimale en paramètre, ce qui permet de prédire la position du jouet et de calculer l'erreur de prédiction. On peut ensuite ajouter le vecteur [action, erreur] a la base de donnée de MP et [action, distance au jouet] a la base de donnée de P.

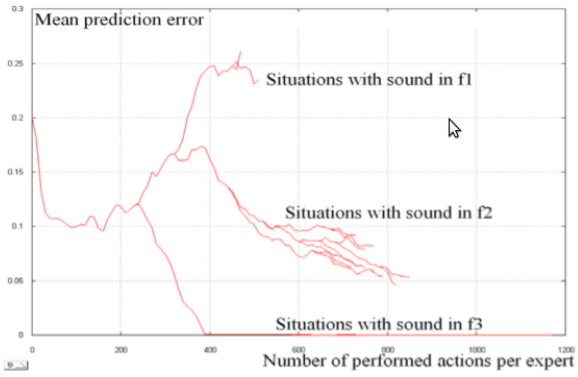
P doit génerer de nouveaux experts au cours de la simulation, pour séparer un expert en deux nouveaux experts nous

utilisons l'algorithme suivant : nous avons deux critères C1 et C2. Le critère C1 est relativement simple, il consiste a diviser un expert lorsque la base de donnée de celui-ci atteint un certain seuil (dans notre simulation nous utilisons 300). Le second critére définit comment séparer les données. Pour chaque élement de l'espace sensori-moteur, [leftSpeed, rightSpeed, frequency, distance], nous calculons la valeur médiane afin de séparer notre base en deux, nous calculons ensuite la variance interclasse. La dimension séparatrice sélectionnée est celle ayant la plus petite variance.

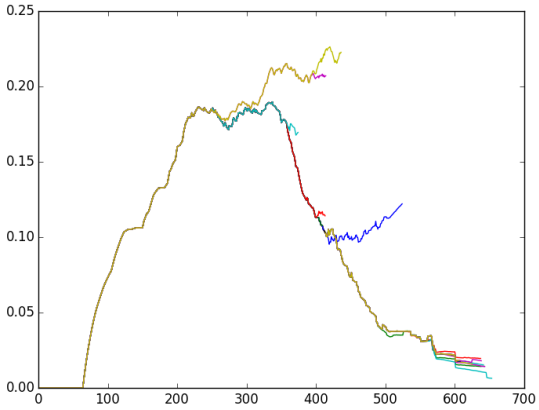
Résultats

La simulation a été effectuée dans les conditions suivantes, DELAY = 150, le nombre d'exemples aléatoires tirés pour déterminer l'action optimale est de 20, l' algoithme de prédiction utilisé est un 20-ppv. Les experts de la learning-machine P sont divisés pour un critére C1 = 300.

Les résultats théoriques attendus de l'algorithme selon le document original de Pierre-Yve Oudeyer et Frederic Kaplan du Sony Computer Science Lab. (Oudeyer & Kaplan, n.d.)



Ci-dessous, les résultats obtenus par notre implémentation de l'algoithme IAC, le graphique est assez similaires aux données exposés par le document du Sony Computer Lab.



La moyenne d'erreur nulle pour 150 premières valeurs s'explique par le fait que nous manquons de données pour les DELAY premiéres iterations, nous sommes donc incapable de prédire l'erreur. (nous avons donc fixé la distance prédite a 0)

References

Oudeyer, & Kaplan. (n.d.). *Intelligent adaptive curiosity: a source of self developpment.* Retrieved from `http://cogprints.org/4144/1/oudeyer.pdf`