Apprentissage par renforcement

Méthode directe d'itération sur les politiques

Anne-Laure Jachiet - Aurélien Boffy

janvier 2007

1 Présentation de l'algorithme utilisé

Nous présentons dans ce projet un algorithme introduit en 2003 par Lagoudakis et Parr dans leur article *Reinforcement Learning as Classification : Leveraging Modern Classifiers.*

1.1 Principe

L'algorithme présenté dans ce rapport est une méthode directe d'itération sur les politiques avec approximation, qui contourne les difficultés liées à la représentation et à l'approximation de la fonction valeur, en ramenant le problème d'apprentissage de politique à un problème de classification.

En effet, au lieu de déterminer la fonction valeur pour tous les états (ce qui est particulièrement problématique pour des espaces d'états continus et de grande dimension), l'algorithme apprend une bonne politique pour un nombre restreint d'états $(x_n)_{1\leq n\leq N}$ via des méthodes de Monte-Carlo nommées *rollout*, puis utilise un classifieur pour généraliser la politique à tout l'espace d'états.

Le problème de classification se présente de la manière suivante : les entrées sont issues de l'espace d'états $\mathcal{S}$, et les sorties associées (ou « étiquettes ») font partie de l'ensemble des actions que l'on note $\mathcal{A}$. Une politique est donc en quelque sorte un classifieur qui associe à chaque état une action (le nombre de classes étant $|\mathcal{A}|$). Enfin, l'ensemble d'apprentissage est formé de l'ensemble des états $(x_n)_{1\leq n\leq N}$ et des actions qui leur ont été associées.

En théorie, n'importe quel algorithme de classification peut être utilisé, mais les auteurs proposent l'utilisation des techniques récentes comme les *Support Vectors Machines* (SVM), qui sont robustes et performantes. Les deux étapes critiques de l'algorithme sont donc plutôt le choix des états $(x_n)_{1\leq n\leq N}$ et la détermination de l'action optimale pour chacun d'eux.

1.2 Pseudo-code

L'algorithme consiste à partir d'une politique initiale π_0 qui peut être aléatoire, puis, à la k$^{\text{è}}$ itération, à estimer une nouvelle politique π_{k+1} en calculant les actions qui maximisent les Q-valeurs pour les états $(x_n)_{1\leq n\leq N}$ (Q-valeurs que l'on estime par Monte-Carlo avec la politique π_k), et en utilisant ces couples états/actions pour la phase d'apprentissage du classifieur, afin que celui-ci puisse être capable, pour l'itération suivante, d'estimer quelle est l'action optimale en chaque point de l'espace d'état.

Plus précisément, voici le pseudo-code de l'algorithme :

$\pi' \leftarrow \pi_0$
Répéter
  $\pi \leftarrow \pi'$
  $bdd\_apprentissage \leftarrow \emptyset$
  Pour $n \in \{1 \dots N\}$
    Pour $a \in \mathcal{A}$
      $\tilde{Q}^{\pi}(x_n, a) \leftarrow$ESTIMER-Q-VALEUR$(x_n,\ a,\ \pi)$
    Fin pour
    $a^* \leftarrow \text{argmax}_{a \in \mathcal{A}} \tilde{Q}^{\pi}(x_n, a)$
    Si $\forall a \neq a^*,\ \tilde{Q}^{\pi}(s, a^*) > \tilde{Q}^{\pi}(x_n, a)$
      $bdd\_apprentissage \leftarrow bdd\_apprentissage \cup \{(s, a^*)\}$
    Fin si
    $\pi' \leftarrow$APPRENTISSAGE$(bdd_apprentissage)$
  Fin pour
Jusqu'à $\pi \approx \pi'$
Retourner π

Le choix de l'action optimale associée à chacun des états x_n se fait donc en estimant par Monte Carlo une estimation $\tilde{Q}^{\pi}(x_n, a)$ de la Q-valeur pour chaque action $a \in \mathcal{A}$, c'est-à-dire en lançant K trajectoires de longueur T, et en calculant la récompense cumulée moyenne :

Pour $k \in \{1 \dots K\}$
  (s',r) $\leftarrow$ SIMULE(s,a)
  $\tilde{Q}_k \leftarrow r$
  $s \leftarrow s'$
  Pour $t \in \{1 \dots T-1\}$
    (s',r) $\leftarrow$SIMULE$(s, \pi(s))$
    $\tilde{Q}_k \leftarrow \tilde{Q}_k + \gamma^t r$
  Fin pour
Fin pour
$\tilde{Q} \leftarrow \frac{1}{K} \sum_{k=1}^{K} \tilde{Q}_k$

---

où SIMULE correspond à un appel au modèle génératif dont on dispose par hypothèse : il indique dans quel état on arrive et quelle récompense on obtient lorsqu'on part d'un certain état et qu'on effectue une action donnée.

1.3 Choix des états $(x_n)_{1 \leq n \leq N}$

Le choix des états $(x_n)_{1 \leq n \leq N}$ qui sont utilisés pour entraîner le classifieur afin qu'il puisse généraliser la politique à l'ensemble complet des états est important. Les choix les plus simples consistent à les placer sur une grille uniforme dans l'espace d'états $\mathcal{S}$ ou à les tirer selon une distribution uniforme. Ces méthodes ont l'avantage et l'inconvénient d'être indépendante du processus ou de la politique. Elles sont bien adaptées pour des problèmes en petite dimension, car elles assurent une bonne couverture de l'espace d'états, mais pour des problèmes en grande dimension, une bonne couverture par la distribution uniforme requiert un nombre d'états N très important, ce qui est prohibitif en termes de temps de calcul.

Il faut alors choisir une distribution qui privilégie les états régulièrement visités (ainsi que les états visités tôt dans la trajectoire car contribuant plus à la Q-valeur), par exemple la distribution stationnaire de la politique recherchée π_{k+1}, qui n'est par principe pas connue ! On peut utiliser la distribution stationnaire de la politique courante π_k, mais elle peut être sensiblement différente.

1.4 Algorithme d'apprentissage

Comme nous l'avons déjà évoqué précédemment, la phase d'apprentissage peut se faire grâce à de multiples méthodes d'apprentissage classiques qu'on utilise en fait comme une « boîte noire ».

Nous en avons utilisé deux différentes : la méthode des k plus proches voisins car elle est facile à implémenter, et les SVM multi-classes avec noyaux gaussiens, pour lesquels nous avons utilisé une toolbox de Matlab. Évidemment, la méthode des SVM est bien plus robuste et performante, mais pour le problème simple que l'on considère (voir plus bas), nous avons constaté des résultats presque similaires.

Notons aussi que l'algorithme original tel qu'il est présenté dans l'article utilise en fait des exemples positifs et des exemples négatif lors de la phase d'apprentissage, c'est-à-dire qu'en plus de dire quelle action est à privilégier en tel état, il utilise aussi le fait que des actions peuvent être particulièrement à éviter. Nous n'avons pas utilisé cette technique.

1.5 Limites

Comme nous le constatons, cet algorithme permet d'éviter les difficultés liées à l'approximation de la fonction valeur sur l'espace d'états tout entier. En contre-partie, il requiert l'utilisation intensive d'un modèle génératif qui est donc supposé connu. Or, celui-ci peut parfois être très coûteux à faire tourner.

Plus précisément, la complexité de calcul de l'algorithme est importante, car il fait appel $N * |\mathcal{A}| * K * T$ fois au modèle génératif. Or la précision de l'approximation des Q-valeurs dépend évidemment du nombre de trajectoires K et de leur longueur T, tandis que celle de la classification et donc de la politique estimée dépend fortement de N.

De plus, comme pour toutes les méthodes qui fonctionnent par itération sur les politiques *avec approximation*, il n'y a pas de garantie de l'amélioration de la performance des politiques successives (contrairement à l'algorithme d'itération sur les valeurs) ni de la convergence vers une politique optimale.

Il est aussi très difficile de pouvoir conduire une analyse précise des performances d'une telle méthode. Ceci est lié d'une part au fait qu'on utilise des techniques d'approximation de Monte-Carlo pour l'estimation des Q-valeurs (ce qui est malgré tout assez classique en apprentissage par renforcement), mais aussi au fait qu'on utilise une méthode d'apprentissage automatique pour généraliser ce qu'on a appris grâce à un nombre fini d'exemples, à l'ensemble d'états complet. L'utilisation de telles méthodes constitue une des principales originalités de l'algorithme présenté dans cet article, mais elle rend son analyse assez complexe, notamment car la politique déduite de la base de données que l'on a constituée grâce aux états $(x_n)_{1\leq n\leq N}$ dépend de l'algorithme d'apprentissage que l'on a utilisé comme « boîte noire ».

2 Le pendule inversé

2.1 Présentation

Nous avons appliqué cet algorithme au problème du pendule inversé : nous considérons un pendule représenté par une tige fixée en un point à un chariot qui roule sur des rails. Nous notons θ l'angle que forme la tige avec l'axe vertical.

Le problème du pendule inversé consiste à maintenir la tige à la verticale ($\theta \approx 0$), sans qu'elle tombe, c'est-à-dire sans qu'elle dépasse l'horizontale ($|\theta| > \pi/2$), en appliquant simplement une force au chariot. Trois actions sont possibles : une poussée vers la gauche, une poussée vers la droite, ou aucune force. À chaque action est ajouté un bruit aléatoire uniforme. La difficulté physique du problème vient bien-sûr du fait que la position verticale est un équilibre instable pour le pendule, que l'action bruitée vient perturber.

La simulation des mouvements du pendule (ce que l'on a appelé plus haut le « modèle génératif ») requiert d'en connaître les dynamiques. Les transitions dépendent de l'angle θ, de la vitesse angulaire $\dot{\theta}$, et du contrôle (bruité) u au travers de l'équation suivante :

$$\ddot{\theta} = \frac{g\sin(\theta) - \alpha ml(\dot{\theta})^2 \sin(2\theta)/2 - \alpha\cos(\theta)u}{4l/3 - \alpha ml\cos^2(\theta)} \tag{1}$$

Les valeurs des différents paramètres physiques sont regroupées dans ce tableau (on a noté $\alpha = \frac{1}{m+M}$) :

Constante de gravité	$g = 9,8$ m.s^{-2}
Masse du pendule	$m = 2$ kg
Masse du chariot	$M = 8$ kg
Longueur du pendule	$l = 0,5$ m

FIG. 1 – Paramètres physiques du pendule inversé

En revanche, on ne suppose pas les dynamiques et paramètres physiques du pendule (longueur, masse...) connus par l'agent ; la seule connaissance dont il dispose provient des essais qu'il peut faire.

Une première stratégie naïve consiste à appliquer une poussée vers la gauche au chariot si l'angle est négatif, et une poussée vers la droite si l'angle est positif. Cependant, celle-ci ne fonctionne pas comme le montre la figure ci-dessous : le pendule tombe presque immédiatement (après 2 secondes).

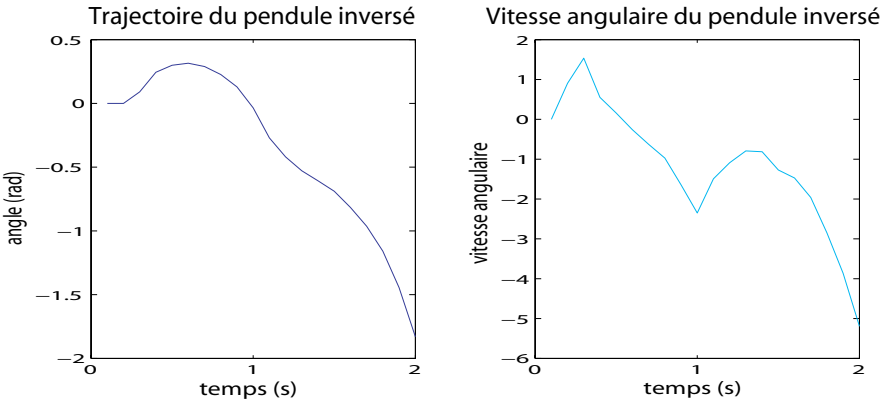


FIG. 2 – Trajectoire du pendule inversé selon la politique naïve

Puisqu'on ne dispose que d'un modèle génératif et que l'on peut constater avec cette stratégie naïve qu'il n'est pas évident de trouver une réponse efficace au problème, l'élaboration d'une stratégie optimale est un problème d'apprentissage par renforcement.

2.2 Formulation du processus de décision markovien

Le problème du pendule inversé se met sous la forme d'un processus de décision markovien à horizon temporel infini avec critère actualisé. On prend comme espace d'états continu $\mathcal{S}$ le plan de phase $(\theta, \dot{\theta})$; l'espace d'actions $\mathcal{A}$ est constitué des trois actions qui consistent à appliquer une force vers la gauche ($u = -50$ N), vers la droite ($u = 50$ N), ou à ne pas appliquer de force du tout ($u = 0$ N). On considère, dans un premier temps, un problème à temps discret, l'intervalle entre deux instants de décision successifs étant constant égal à δt.

La dynamique d'état est alors donnée par la dynamique du pendule. Étant donné l'état $s = (\theta(t), \dot{\theta}(t))$ et l'action a, l'équation (1) donne la dérivée seconde de l'angle $\ddot{\theta}(t)$; on en déduit par « intégration discrète » l'état $s' = (\theta(t+1), \dot{\theta}(t+1))$:

$$\begin{aligned} \theta(t+1) &= \theta(t) + \delta t\, \dot{\theta}(t) \\ \dot{\theta}(t+1) &= \dot{\theta}(t) + \delta t\, \ddot{\theta}(t) \end{aligned}$$

Puisque l'action a est bruitée par l'ajout d'un bruit aléatoire uniforme dans l'intervalle [-10 N ; 10 N], la dynamique est également aléatoire.

Enfin, une récompense de 1 est attribuée tant que l'angle θ ne dépasse pas $\pi/2$ en valeur absolue ; si l'angle devient plus grand que $\pi/2$, la trajectoire s'arrête et la récompense est nulle. L'ensemble $\left\{(\theta, \dot{\theta})\,;\, |\theta| > \pi/2,\, \dot{\theta} \in \mathbb{R}\right\}$ forme donc un état absorbant pour le système. Le cœfficient d'actualisation est noté γ.

2.3 Implémentation

2.3.1 Importance des différents paramètres

Nous nous sommes livrés à de multiples séries d'expériences pour tester l'algorithme décrit dans la première partie sur ce rapport, mais nous nous sommes rapidement rendus compte de l'importance du réglage des paramètres, comme le nombre N d'états utilisés pour l'apprentissage, le nombre et la longueur des trajectoires utilisées pour l'estimation des Q-valeurs, le pas de temps, etc.

De plus, les résultats variaient beaucoup d'une simulation à l'autre à paramètres constants. En l'absence d'améliorations notables et étant donné les temps de calculs extrêmement importants, nous nous sommes donc restreints à des valeurs modérées résumées dans le tableau ci-dessous :

Nombre N d'états d'apprentissage	200
Distribution des états d'apprentissage $\rightarrow$ angle $\rightarrow$ vitesse angulaire	uniforme $[-\pi/4, \pi/4]$ $[-3, 3]$
Nombre de trajectoires	100
Longueur des trajectoires	10 à 100
Cœfficient d'actualisation	0,95
Pas de temps	0,1 s

Fig. 3 – Réglages du RCPI

Notons en particulier que nous avons tirés les états initiaux uniquement pour des angles compris entre $-\pi/4$ et $\pi/4$, alors que la trajectoire ne s'arrête que si l'angle dépasse $\pi/2$ en valeur absolue. Cependant il s'avère qu'en pratique, si l'angle dépasse $\pi/4$, il est très difficile de ramener le pendule à une position d'équilibre, et une politique optimale devrait normalement empêcher l'angle de devenir si important. Ainsi, nous avons préféré nous restreindre à cet interval afin de tirer plus d'états que l'on peut qualifier d'« utiles » car ils sont plus susceptibles de se produire pendant la simulation d'une bonne politique : nous limitons ainsi les temps de calcul et améliorons les performances de l'algorithme (notons aussi que la méthode d'apprentissage permet de toute façon d'attribuer une action à n'importe quel état, même si celui-ci ne se trouve pas dans l'intervalle des états utilisés pour entraîner le classifieur).

2.3.2 Critère d'arrêt

Nous avons essayé d'établir des critères d'arrêt relatifs à l'amélioration des Q-valeurs estimées. Cependant, celles-ci fluctuent beaucoup d'une itération à l'autre et l'évaluation d'un critère d'arrêt efficace s'est avéré assez compliquée. Lagoudakis précise dans l'article que pour ce problème très simple, une très bonne politique est de toute façon obtenue après 2 ou 3 itérations. Dans notre cas, c'est un peu plus long, notamment car nous ne prenons pas en compte d'exemples négatifs pour entraîner le classifieur. Nous avons généralement limité à 5 ou à 10 le nombre d'itérations.

2.3.3 Constitution de la base de données d'apprentissage

Rappelons que la récompense attribuée est de 1 lorsque le pendule est au dessus de l'horizontale, et est de 0 lorsqu'il est tombé. Ainsi, la récompense totale correspond (au facteur d'actualisation près) à la durée pendant laquelle le pendule maintient un angle inférieur à $\pi/2$ en valeur absolue.

Si l'on considère des trajectoires trop courtes (une longueur de 10 correspond à 1 seconde puisque le pas de temps est de 0,1 seconde), les trois Q-valeurs correspondants aux trois actions possibles seront très souvent égales. Il convient ainsi d'utiliser d'assez longues trajectoires, mais même dans ce cas, les Q-valeurs sont toujours très proches les unes des autres. Ceci s'explique entre autre par le fait que la plupart des états initiaux considérés aboutissent à une chute presque immédiate du pendule. En effet, même si nous avons déjà fait attention à limiter les intervalles de θ et de $\dot{\theta}$ (comme nous l'avons expliqué plus haut), la grande majorité des configurations tirées sont trop difficiles à rééquilibrer étant donné que l'éventail des forces disponibles est très limité.

Nous avons donc dû utiliser quelques « astuces » afin de gérer ces Q-valeurs trop souvent égales : dans un premier temps, nous avions décidé de ne conserver pour la phase d'apprentissage que les états x_n pour lesquels une Q-valeur était strictement supérieure aux deux autres (voire supérieure à s fois les deux autres, où $s > 1$). Cependant, le nombre d'états qui respectaient de tels critères était trop petit, et donc le classifieur était entraîné avec un nombre faible d'exemples, ce qui aboutissait à des politiques qui variaient énormément d'une itération à l'autre. Nous avons donc décidé de conserver tous les états $(x_n)_{1 \leq n \leq N}$ pour entraîner le système, en conservant les actions issues de la politique précédente pour les états x_n pour lesquels aucune Q-valeur n'était strictement supérieure aux deux autres. Nous avons alors obtenu une plus grande stabilité des politiques d'une itération à la suivante.

Une deuxième méthode que nous avons utilisée pour éviter l'égalité des Q-valeurs a consisté à modifier les récompenses attribuées. Nous avons utilisé une récompense qui était à chaque pas de temps d'autant plus élevée que l'angle θ était faible en valeur absolue.

2.4 Résultats

Dans les trois pages qui suivent, nous présentons différents résultats obtenus lors de nos simulations : des politiques mauvaises (le pendule tombe presque tout de suite, voir figure 4), moyenne (le pendule reste en équilibre pendant près de 40 secondes puis finit par tomber, voir figure 5), ou excellente (le pendule est toujours en équilibre au bout de 100 secondes, et y reste en fait certainement indéfiniment, voir figure 6). Pour chacune, nous avons représenté sur la même page la politique dans le plan de phase, un exemple de trajectoire en suivant cette politique et la fonction valeur estimée (c'est à dire la meilleure des trois Q-valeurs) dans le plan de phase.

On constate sur ces exemples que c'est le comportement de la politique près du point $(0, 0)$ qui est déterminant. En effet, une mauvaise décision à cet endroit peut conduire à une chute immédiate du pendule. Or, le point $(0, 0)$ se trouve à proximité de la séparation entre les différentes actions, ce qui rend la performance de la politique obtenue très sensible à la simulation et explique ces résultats contrastés. On constate également que l'action nulle n'est pas nécessairement utilisée pour obtenir une bonne politique.

Enfin, l'examen des fonctions valeurs montre une domination de la deuxième diagonale : si l'angle est assez grand en valeur absolue, mais que la vitesse est également importante et de signe contraire, le pendule parvient à maintenir son équilibre, ce qui n'est pas le cas avec un angle et une vitesse modérés mais dans le même sens.

Remarquons au passage que dans le cas de la politique « moyenne », la fonction valeur est plus importante que pour les deux autres exemples (culmine à environ 15 contre 8). Ceci provient du fait que cette simulation a été effectuée avec des trajectoires de longueur 30 et donc que si le pendule reste au dessus de l'horizontale sur toute la trajectoire, la récompense totale est $\sum_{t=0}^{29} \gamma^t \approx 15$, alors que la longueur de la trajectoire était fixée à 10 pour les deux autres simulations (et $\sum_{t=0}^{9} \gamma^t \approx 8$). Ce n'est donc pas en contradiction avec sa performance inférieure à la « bonne » politique, mais cela montre que les Q-valeurs estimées sont différentes des vraies Q-valeurs (il faudrait dans ces cas utiliser des trajectoires plus longues pour que l'ajout de γ^t devienne négligeable), et permet d'expliquer là encore la variabilité de l'algorithme.

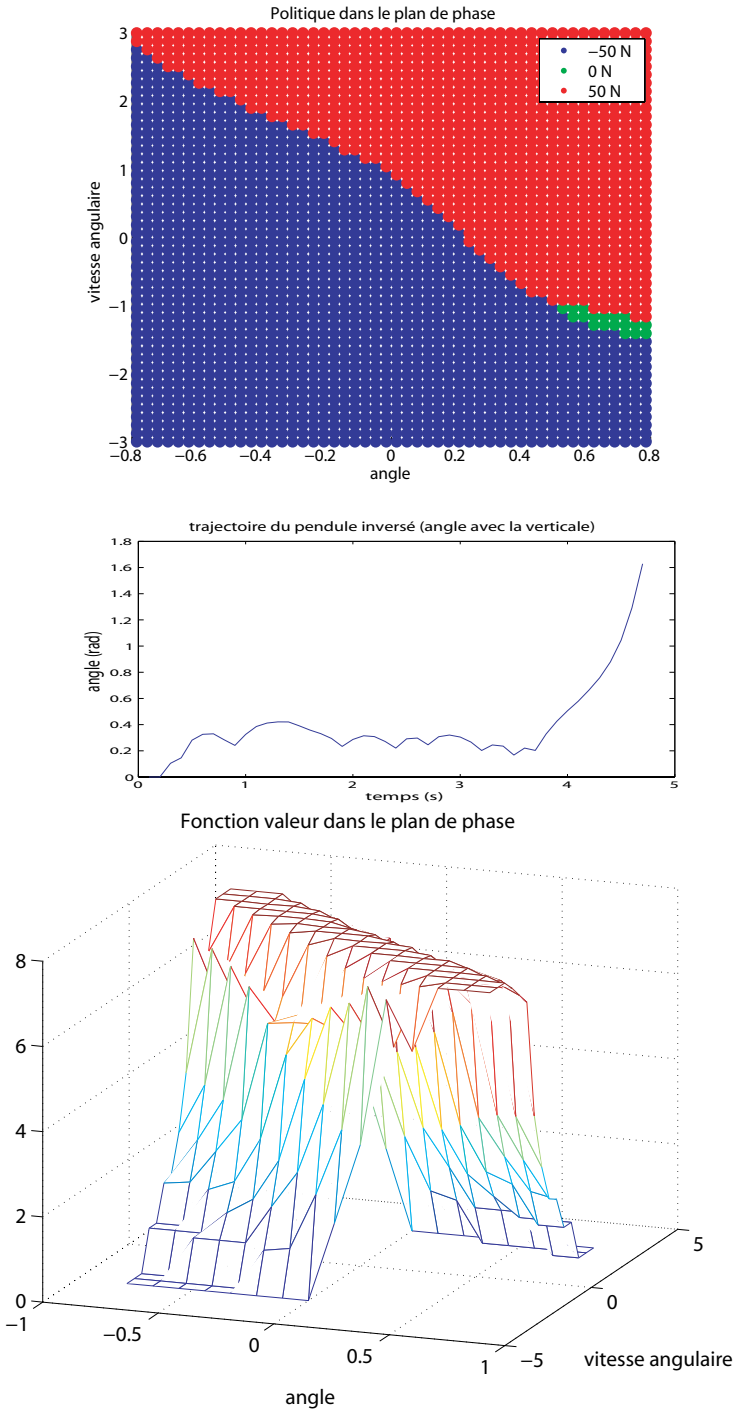


FIG. 4 – Mauvaise politique : tombe presque tout de suite

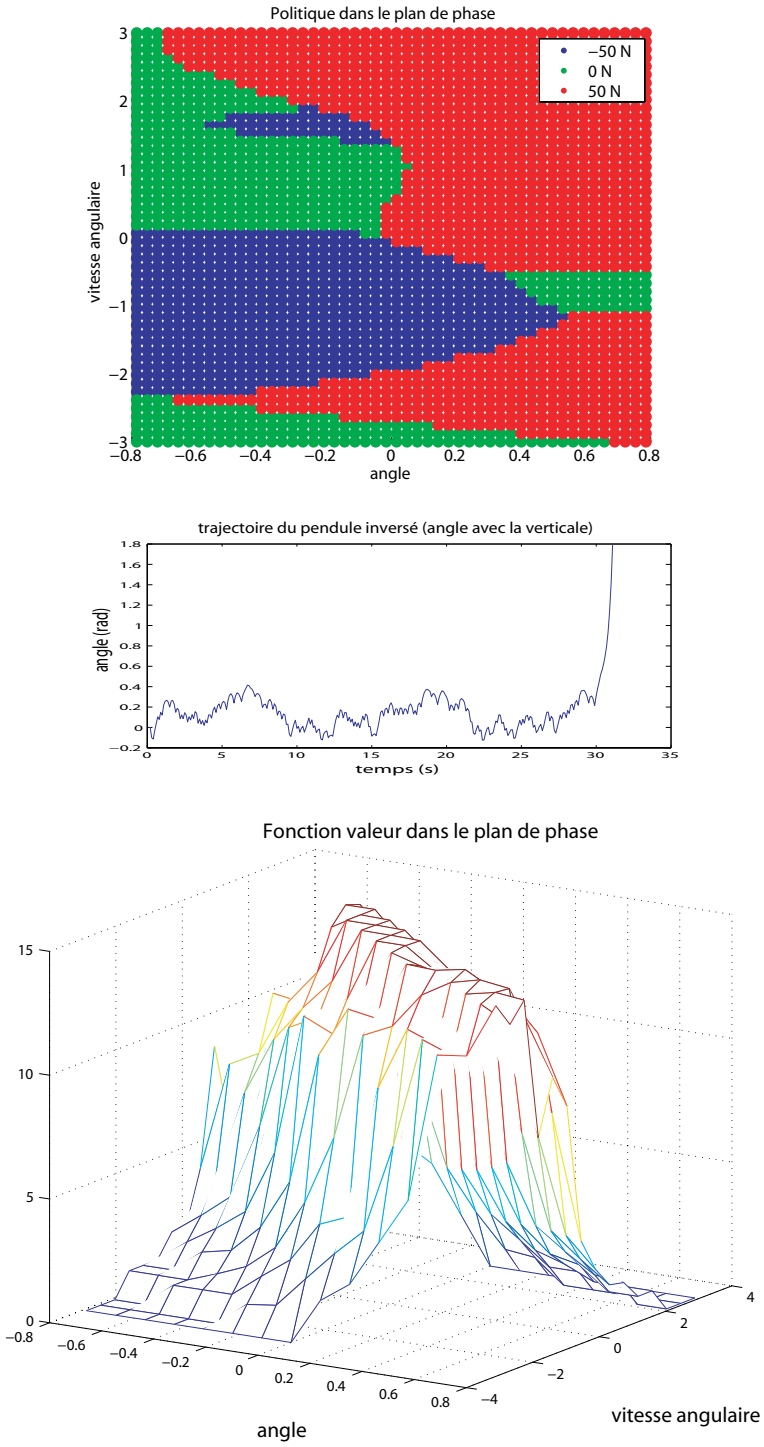


FIG. 5 – Politique moyenne : tient en équilibre pendant 40 secondes

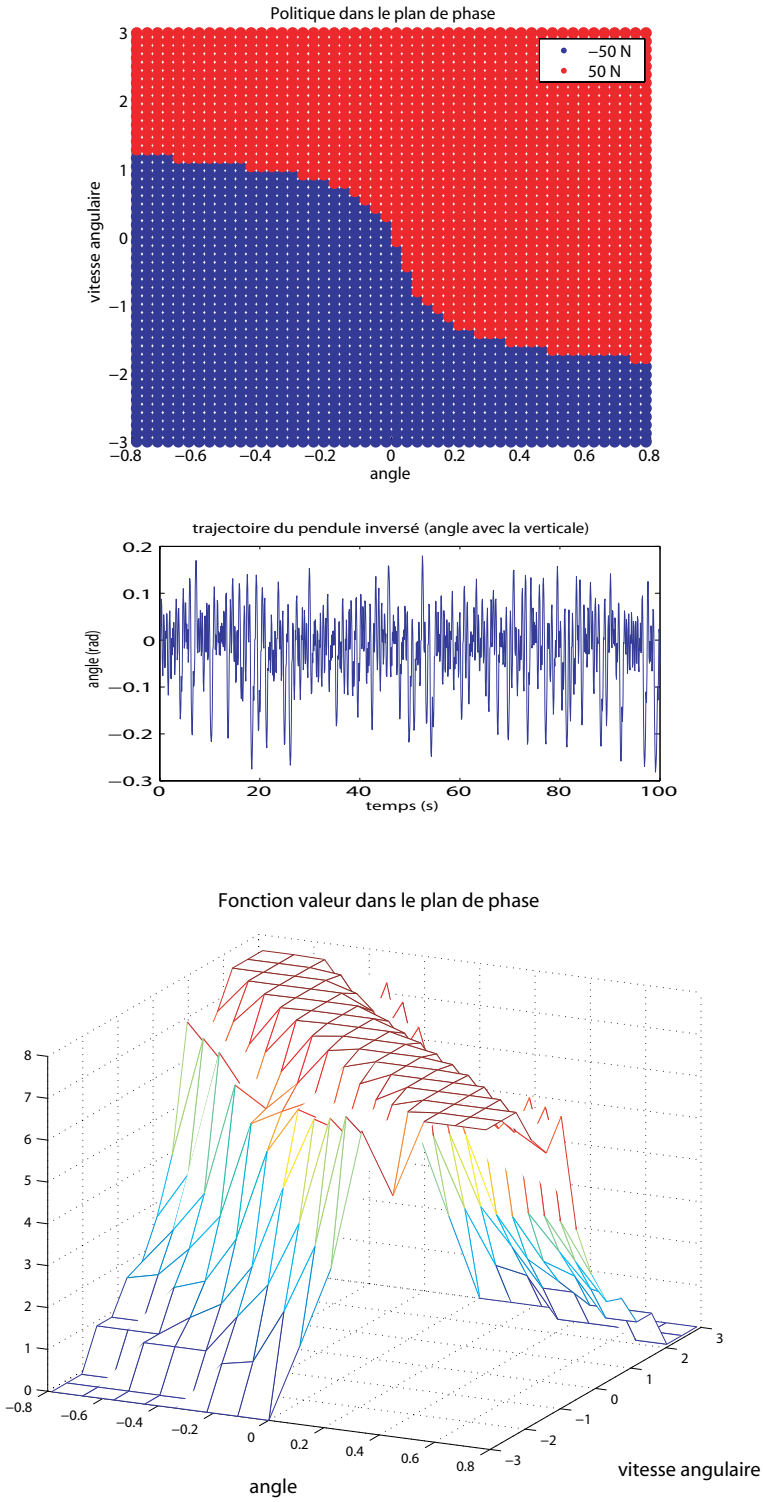


FIG. 6 – Bonne politique : tient en équilibre certainement indéfiniment

3 Version à temps continu

On considère désormais le même problème mais en temps continu, dont la dynamique est donnée par

$$\frac{dx(t)}{dt} = f(x(t), a(t))$$

où $x(t) = (\theta(t), \dot{\theta}(t))$ est l'état à l'instant t et l'action $a(t)$ suit la politique courante, $a(t) = \pi_k(x(t))$.

3.1 Choix de l'action maximisante

Étant donné un état $x_n = x(0)$, on ne peut plus calculer la Q-valeur des couples (x_n, a) pour les différentes actions a. Au lieu de lancer des trajectoires pour chacun de ces couples (x_n, a), $a \in \mathcal{A}$, on lance uniquement des trajectoires pour le couple (x_n, a_n) avec $a_n = \pi_k(x_n)$, et on calcule le gradient de la fonction valeur par rapport à l'état initial $x_n = (\theta_n, \dot{\theta}_n)$.

Puisque

$$V^{\pi_k}(x_n) = \int_0^\infty \gamma^t r(x(t))dt,$$

on a

$$\nabla_{x_n} V^{\pi_k}(x_n) = \int_0^\infty \gamma^t \nabla_{x_n} r(x(t))dt = \int_0^\infty \gamma^t \nabla_x r(x(t)) \nabla_{x_n} x(t)dt.$$

Une récompense constante par morceaux valant 1 pour un angle inférieur à $\pi/2$ et 0 pour un angle plus grand ne permettrait pas de donner un sens à l'expression précédente. On considère donc une récompense affine par morceau $r(x) = r(\theta, \dot{\theta}) = (\pi/2 - |\theta|)\,\mathbf{1}_{|\theta|<\pi/2}$, d'où

$$\nabla_x r(x) = \left(\frac{\partial r}{\partial \theta}(x), \frac{\partial r}{\partial \dot{\theta}}(x)\right) = \left(-\mathrm{sgn}(\theta)\frac{2}{\pi}\mathbf{1}_{|\theta|<\pi/2}, 0\right).$$

Reste à exprimer $\nabla_{x_n} x(t) = \begin{pmatrix} \frac{\partial \theta}{\partial \theta_n}(x(t)) & \frac{\partial \theta}{\partial \dot{\theta}_n}(x(t)) \\ \frac{\partial \dot{\theta}}{\partial \theta_n}(x(t)) & \frac{\partial \dot{\theta}}{\partial \dot{\theta}_n}(x(t)) \end{pmatrix}$

Comme

$$\frac{d}{dt}\nabla_{x_n} x(t) = \nabla_x \frac{dx(t)}{dt} \nabla_{x_n} x(t) = \nabla_x f(x(t), a(t)) \nabla_{x_n} x(t),$$

On en déduit que :

$$\nabla_{x_n} x(t + \delta t) = \nabla_{x_n} x(t) + \delta t \nabla_x f(x(t), a(t)) \nabla_{x_n} x(t),$$

avec $\nabla_{x_n} x(0) = I_2$ et $f(x(t), a(t)) = (\dot{\theta}, \ddot{\theta})$

Ainsi :

$$\nabla_x f(x(t), a(t)) = \begin{pmatrix} \frac{\partial \dot{\theta}}{\partial \theta}(t) & \frac{\partial \dot{\theta}}{\partial \dot{\theta}}(t) \\ \frac{\partial \ddot{\theta}}{\partial \theta}(t) & \frac{\partial \ddot{\theta}}{\partial \dot{\theta}}(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \frac{\partial \ddot{\theta}}{\partial \theta}(t) & \frac{\partial \ddot{\theta}}{\partial \dot{\theta}}(t) \end{pmatrix},$$

ces deux dernières dérivées étant calculées à l'aide de l'équation (1).

On définit alors l'action a_n comme étant celle qui maximise la produit scalaire :

$$a_n := \arg\max_{a \in \mathcal{A}} \nabla_{x_n} V^{\pi_k}(x_n) \,.\, f(x_n, a).$$

En effet, cela revient à choisir la dérivée de l'état initial dans la direction du gradient de la fonction valeur, ou du moins le plus près possible, et donc choisir la direction qui va maximiser la fonction valeur.

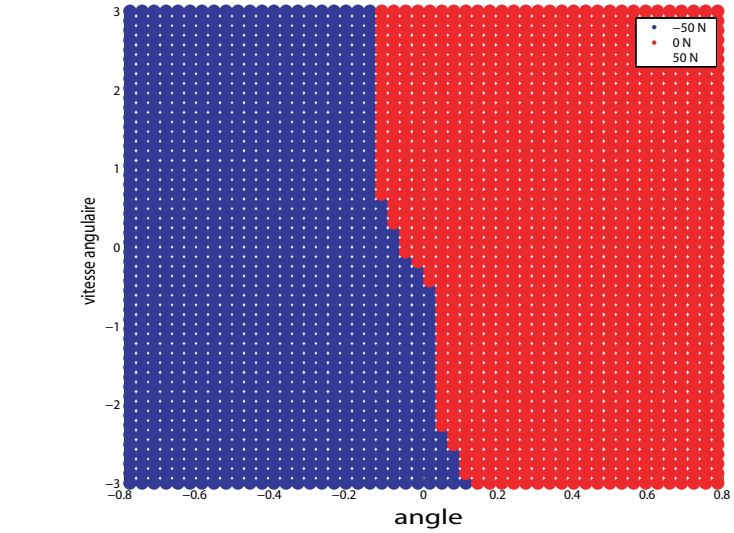
3.2 Résultats

La version à temps continu remplit son principal objectif, à savoir la diminution du temps de calcul, qui est au moins divisé par deux, les autres paramètres étant constants.

En ce qui concerne les performances, elles ne sont pas sensiblement meilleures que celles de la version discrète, puisque l'on obtient des politiques excellentes qui maintiennent le pendule en équilibre certainement indéfiniment, mais pas systématiquement non plus.

La figure 7 page suivante représente une bonne politique obtenue par l'algorithme en temps continu ainsi qu'une trajectoire qui en est issue. À première vue, la politique est très proche de la politique naïve, dont nous avons vu au paragraphe 2.1 qu'elle faisait tomber le pendule tout de suite. Mais en fait, à proximité du point $(0, 0)$, la séparation entre les actions « gauche » et « droite » se fait dans la deuxième diagonale. Ceci confirme à nouveau la sensibilité autour de ce point.

Quant à la trajectoire obtenue, on constate qu'elle oscille nettement moins que celle de la « bonne » politique à temps discret. Curieusement, le pendule se maintient autour d'un angle faible mais non nul de $-0,06$ radians.



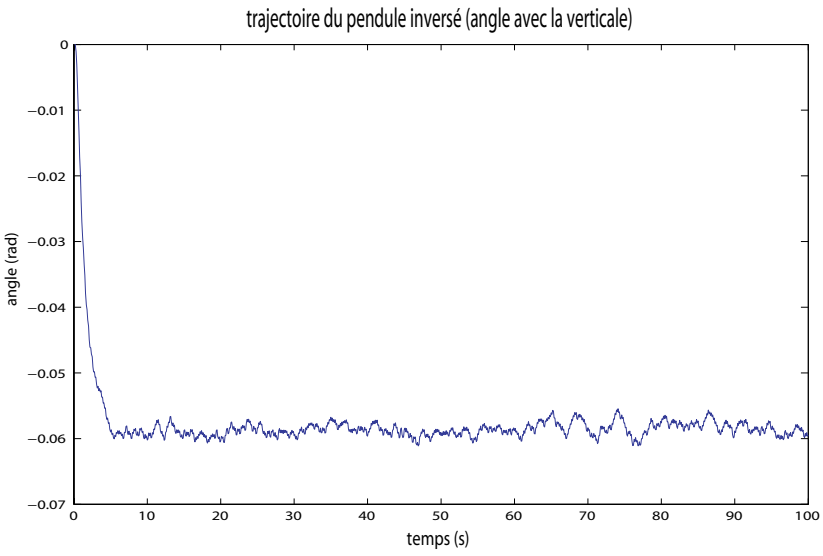


FIG. 7 – Version temps continu, bonne politique

4 Problème de contrôle de dimension 4

Nous avons implémenté un code qui permet de voir l'évolution du pendule et du chariot pour le modèle précédent, et nous avons donc pu observer leurs trajectoires dans le cas des politiques que nous avons trouvées.

Pour ce faire, il a fallu ajouter au programme l'équation de la dynamique du chariot suivante :

$$\ddot{x} = \frac{\left[\dot{\theta}^2 \sin(\theta) - \ddot{\theta}\cos(\theta)\right] ml + u}{m + M} \tag{2}$$

en utilisant les mêmes notations que précédemment, avec x représentant la position du chariot sur l'axe horizontal.

Nous avons ainsi pu constater que pour toutes les « bonnes » politiques que nous avions trouvées, c'est-à-dire celles pour lesquelles le pendule semblait rester presque indéfiniment à l'équilibre, le chariot se déplaçait quant à lui beaucoup, dans un sens ou dans l'autre.

Pour empêcher le chariot de trop bouger, nous avons donc essayé de résoudre un problème de contrôle de dimension 4, l'espace d'état devenant $[\theta, \dot{\theta}, x, \dot{x}]$.

Les dynamiques sont connues puisqu'il suffit de combiner les équations 1 et 2. En ce qui concerne les récompenses, nous avons essayé d'attribuer une récompense de 1 lorsque l'angle θ restait inférieur à $\pi/2$ en valeur absolue et que l'abscisse du chariot restait dans l'intervalle $[-10; 10]$, et une récompense nulle si l'une de ces deux conditions venait à ne plus être respectée. De façon analogue à ce que nous avons expliqué dans le cas du problème en dimension 2, nous avons aussi essayé, afin de mieux séparer les Q-valeurs, d'attribuer une récompense égale à

$$r = \mathbf{1}_{|x|<10}(10 - |x|)\,\mathbf{1}_{|\theta|<\frac{\pi}{2}}\left(\frac{\pi}{2} - \theta\right)$$

Malheureusement, nous n'avons pas obtenu de résultats satisfaisants, la raison tenant certainement au fait que le nombre N d'états à considérer doit être beaucoup plus élevé en dimension 4 qu'en dimension 2. Par exemple, si l'on utilise $N = 200$ pour le problème initial, il faudrait, pour pouvoir couvrir l'espace d'états de façon aussi dense, utiliser $N = 200^2 = 40000$ états pour le problème en dimension 4. Même si ce nombre est évidemment trop élevé, nous avons voulu l'essayer sur 1 ou 2 itérations mais nous n'avons réussi qu'à obtenir des erreurs *Out Of Memory* par Matlab avec notre implémentation et plus particulièrement avec la toolbox SVM. Dans l'article initial, les auteurs utilisent déjà $N = 5000$ états lorsqu'ils appliquent l'algorithme à un problème de dimension 4, ce qui est aussi un nombre très élevé

et demande des temps de calculs très longs avec notre implémentation naïve sous Matlab.

Nous avons déjà évoqué au début de ce rapport le problème du choix des états $(x_n)_{1\leq n\leq N}$. L'utilisation d'une distribution uniforme n'est clairement pas adaptée lorsqu'on travaille en dimension élevée (et en fait nous nous rendons compte que des problèmes apparaissent dès le cas de la dimension 4), car le nombre d'états à considérer évolue de façon exponentielle avec la dimension.

Comme cela est expliqué dans l'article de Lagoudakis, différentes propositions ont été suggérées pour tenter d'éviter ce problème, comme la méthode qui consiste à choisir les points $(x_n)_{1\leq n\leq N}$ en fonction de la politique actuelle π_k, en favorisant les états qui sont les plus régulièrement visités dans des trajectoires « classiques ». Cependant, la politique π_{k+1} peut être très différente de la politique π_k. D'où l'idée d'introduire une astuce permettant de maintenir les politiques successives assez proches les unes des autres, d'une façon un peu similaire à ce dont nous avons parlé dans la section 2.3.3.

Lorsqu'on observe une politique avec le diagramme de phase, on peut aussi considérer qu'il est important d'avoir un nombre conséquent de points x_n dans les zones de séparation entre 2 actions différentes. Ce sont en effet ces points qui sont les plus discriminants et qui donc doivent être le plus souvent pris en compte par le classifieur pendant la phase d'apprentissage afin d'obtenir une politique précise. Cependant, comme précédemment, 2 politiques successives peuvent être très différentes et les zones de séparation varient donc elles aussi fortement.

Conclusion

L'algorithme que nous avons implémenté démontre de très bonnes performances dans le cas du problème très simple du pendule inversé à deux dimensions, puisqu'on a pu obtenir d'excellentes politiques après quelques itérations seulement, et surtout sans avoir eu à estimer la fonction valeur, ce qui est le principal intérêt de la méthode.

Cependant, le problème de la « malédiction de la dimension » n'a pas du tout été résolu par notre projet, car nous n'avons pas su adapter l'algorithme lorsque nous avons travaillé sur un problème de dimension 4, et même en suivant les préconisations des auteurs de l'article, il paraît difficile d'utiliser cet algorithme pour un problème de dimension encore plus élevée (comme $d = 10$).

Enfin, il était intéressant de voir comment il était possible de combiner ce que nous avons appris dans ce cours d'apprentissage par renforcement avec les autres méthodes que nous voyons dans d'autres cours plus « classiques » d'apprentissage supervisé, comme les k plus proches voisins ou les SVM.