Rapport - CEM vs PG

Réalisé par
Damien LEGROS
Hector KOHLER

Dans le cadre du cours
PANDROIDE

Travail encadré par
Oliver SIGAUD

Master ANDROIDE
Université Sorbonne Université

Table des matières

1 Introduction

Les algorithmes de recherche de politiques permettent d'optimiser le comportement d'un agent : ils permettent à cet agent d'apprendre le meilleur comportement pour effectuer une tâche dans son environnement. Par exemple, avec ces algorithmes, un agent peut apprendre à maintenir une barre en équilibre. Cet exemple est un benchmark classique des algorithmes de recherche de politiques, c'est l'environnement Pendulum, voir Figure 1. Dans ce projet, nous comparons deux classes d'algorithmes de recherche de politiques. D'une part, les algorithmes dits de recherche directe qui, à chaque épisode d'apprentissage, génèrent de nouvelles politiques selon une distribution de probabilités pour ne garder que la meilleure. D'autre part, les algorithmes basés sur l'estimation d'un gradient qui, à chaque épisode d'apprentissage, se basent sur le gradient de la performance de la politique courante pour en calculer une meilleure. Le but de notre étude est de mieux comprendre comment ces deux classes d'algorithme explorent l'espace des solutions des problèmes d'optimisation. Pour cela, nous avons tenté d'expliquer la différence de performance sur Pendulum entre les solutions trouvées par la Cross Entropy Method (algorithme élémentaire de recherche directe) et les solutions trouvées par le Policy Gradient (algorithme élémentaire de recherche par estimation de gradient).

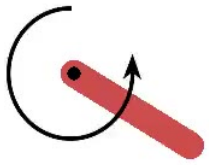


FIGURE 1 – Environnement Pendulum

2 Description des algorithmes étudiés

2.1 Cross-Entropy Method

Pour les algorithmes de recherche directe, nous étudions Cross-Entropy Method (CEM).

Algorithm 1: Cross-Entropy Method

Paramètres :

θ_i : ensemble des paramètres de la solution i

J_i : fitness de la solution i

$(\theta_i, J_i)_{i=0...N}$: ensemble de N paires des valeurs des solutions

ρ : proportion des meilleures solutions à utiliser par itération

σ^2_{bruit} : terme de bruit additionnel

for iteration=1,2,... **do**

Génère N paires (θ_i, J_i) *selon une loi normale* $\mathcal{N}(\mu, \sigma^2 + \sigma^2_{bruit})$

Calcule l'ensemble S_p *des meilleures solutions de max(1,*$N \times \rho$*)*

$\mu \leftarrow moyenne(S_\rho)$

$\sigma^2 \leftarrow écart - type(S_\rho) + \sigma^2_{bruit}$

end for

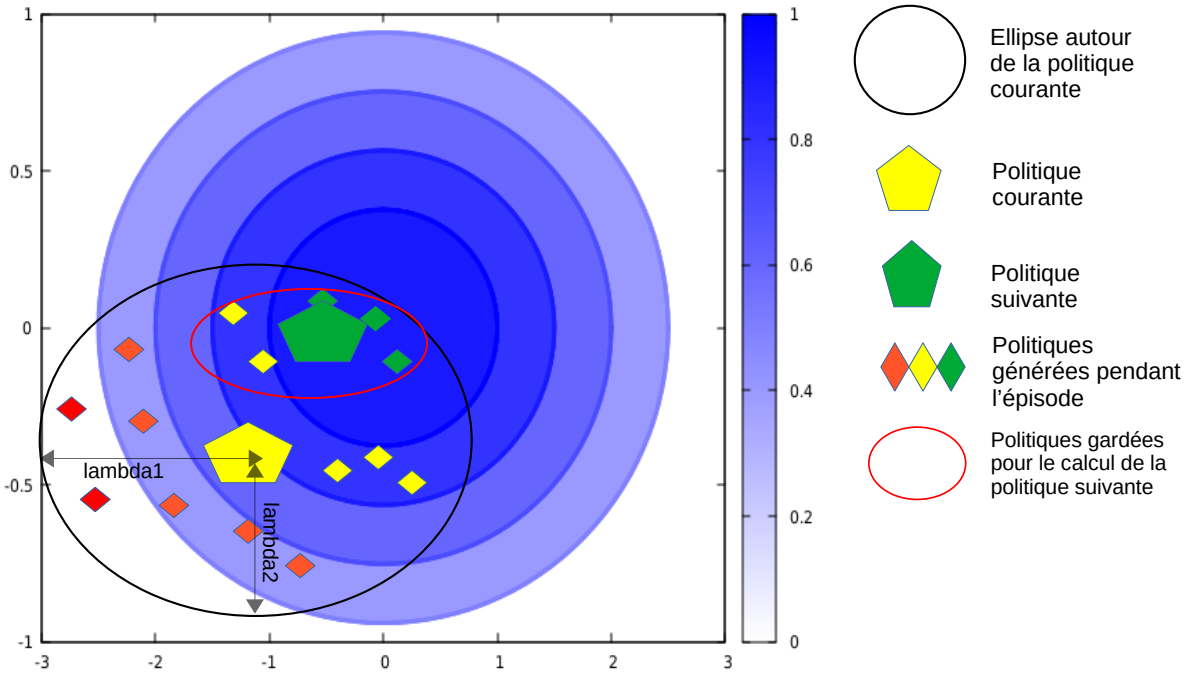


FIGURE 2 – Schéma d'une itération de CEM

Lors d'une itération k, CEM tire N politiques autour de sa politique courante selon une normale $N(moy(S_\rho^k), Cov(S_\rho^k) + \sigma^2_{bruit})$. Ces politiques se trouvent dans une ellipse ayant des demi-axes de tailles λ_1^k et λ_2^k (les valeurs propres de la covariance). Les $N \times \rho$ meilleures solutions parmi ces politiques sont gardées (ellipse rouge correspondant à l'ensemble S_p^{k+1} sur la Figure 2). Celles-ci permettent de calculer la politique suivante (la moyenne de ces meilleures solutions). Le processus est réitéré jusqu'au critère d'arrêt.

2.2 Policy Gradient

Pour les algorithmes de recherche par estimation de gradient, nous étudions Policy Gradient (PG).

Algorithm 2: Policy Gradient

Initialiser les paramètres de la politique θ
for iteration=1,2,... **do**
  Collecter l'ensemble des trajectoires $(s_t^{(i)}, a_t^{(i)}, r_t^{(i)})$, $i \in 1, H$ en exécutant la politique courante.
  A chaque pas pour chaque trajectoire, calculer la somme des récompenses $R_t = \sum_{t'=t}^{T-1} r_{t'}$
  Mettre à jour la politique, à l'aide de l'estimation du gradient de la politique $\hat{g}$, la moyenne des termes $\nabla_\theta log\pi(a_t|s_t, \theta)R_t$.
end for

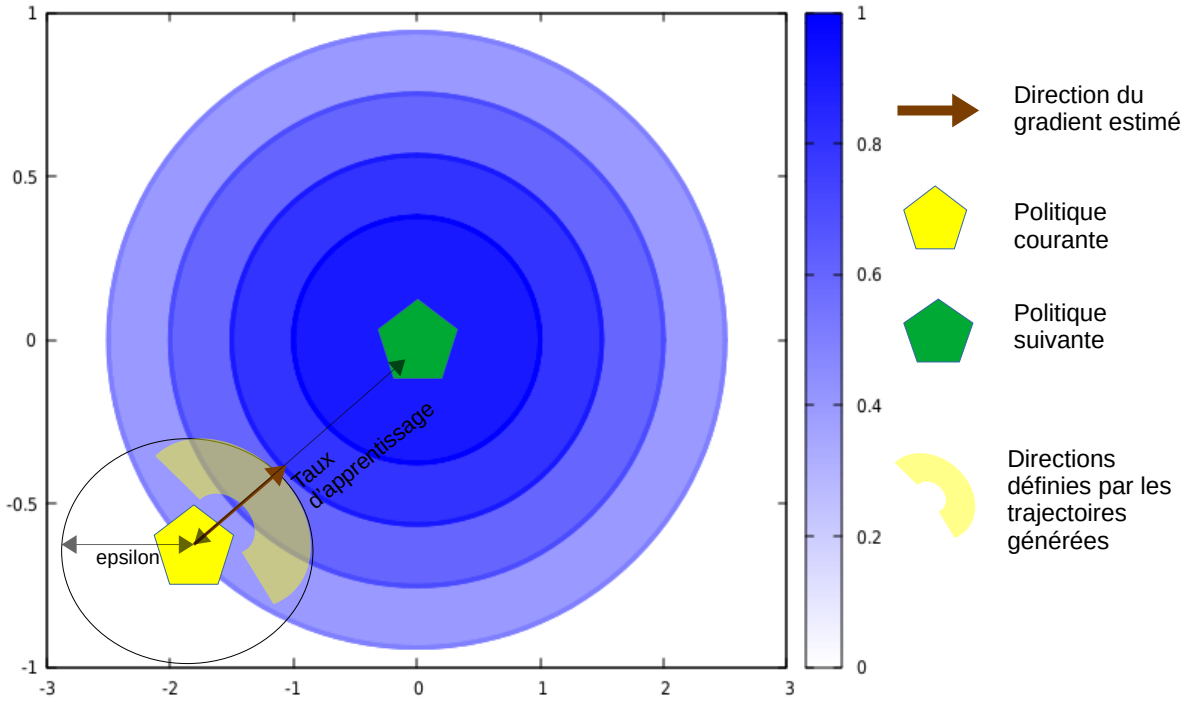


FIGURE 3 – Schéma d'une itération de PG

Lors d'une itération k, PG génère des trajectoires depuis la politique courante θ^k. Il calcule une estimation du gradient $\hat{g}^k$ (une direction dans laquelle explorer l'espace des politiques) à l'aide de l'ensemble des trajectoires collectées (la moyenne des termes $\nabla_\theta^k log\pi(a_t|s_t,\theta^k)R_t$). La politique suivante, θ^{k+1}, est le résultat d'un déplacement depuis θ^k, dans la direction $\hat{g}^k$, sur une distance égale au taux d'apprentissage . Le processus est réitéré jusqu'au critère d'arrêt.

3 Méthode

3.1 Présentation du code et ajouts

Notre code est dérivé de la librairie d'Olivier Sigaud disponible à l'adresse suivante : https ://github.com/osigaud/Basic-Policy-Gradient-Labs. Ce code permet de lancer des études de méthodes de type Policy Gradient sur les environnements de Gym. Comme nous avons ajouté la méthode CEM, il a fallu coder des fonctions de manipulation de poids de réseaux de neurones. Enfin, nous avons ajouté la possibilité d'apprendre une politique de type Beta. Le code du projet Vignette développé par Sarah Kerriche, Lydia Aguini et Yannis Elrharbi-Fleury permet une visualisation de l'espace de recherche des politiques. Il a été modifié pour prendre en entrée les modèles de sortie de notre librairie. https ://github.com/sohio92/P_androide.

3.2 Visualisations

Au-delà des courbes de performances classiques que l'on retrouve dans la littérature en apprentissage par renforcement et dans la librairie originelle d'Olivier Sigaud, nous présentons des visualisations de l'espace de recherche et de l'exploration des algorithmes dans cet espace. On visualise notamment l'évolution de la distance entre deux politiques apprises successivement : pour N épisodes d'apprentissage, un algorithme apprend les politiques $\theta_0, \theta_1, ..., \theta_N$, on peut donc visualiser les distances euclidiennes $\theta_1 - \theta_0, ..., \theta_N - \theta_{N-1}$. De même, on peut visualiser les produits scalaires $(\theta_2-\theta_1)\cdot(\theta_1-\theta_0), ..., (\theta_N-\theta_{N-1})\cdot(\theta_{N-1}-\theta_{N-2})$. Ces derniers donnent une idée des changements de directions lors de l'exploration de l'espace des politiques. On rappelle que le produit scalaire entre deux vecteurs peut s'écrire $\mathbf{u}\cdot\mathbf{v} = \mathbf{u}\times\mathbf{v}\times\cos(\mathbf{u};\mathbf{v})$. Ainsi, un produit scalaire proche de 1 indique peu de changement dans la direction, tandis qu'un produit scalaire proche de 0 indique un grand changement de direction (angle de 90°). Et un produit scalaire négatif indique un changement encore plus grand : un produit scalaire de -1 correspond à un demi-tour.

3.3 Structure d'une politique

On représente une politique par un réseau de neurones. C'est un réseau à 2 couches cachées linéaires de tailles 32 et 64, et 2 sorties avec pour activation une softplus et un décalage 1. Ce réseau prend en entrée l'état de l'agent et renvoie les paramètres (α, β)

d'une distribution Beta (voir Figure 4) avec $\alpha, \beta > 1$. On tire l'action à effectuer selon la distribution de sortie. Nous avons choisi une distribution Beta pour que l'action tirée soit bornée dans le domaine d'action de l'environnement. C'est un choix d'implémentation classique dans le cas d'un environnement au domaine d'action continu comme Pendulum (Chou2017 [2]).

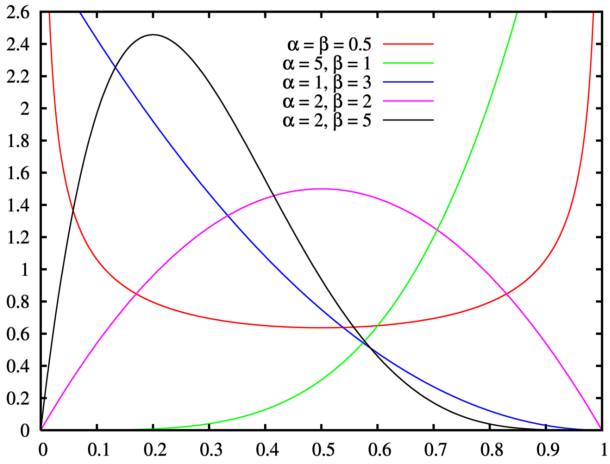


FIGURE 4 – Distribution Beta

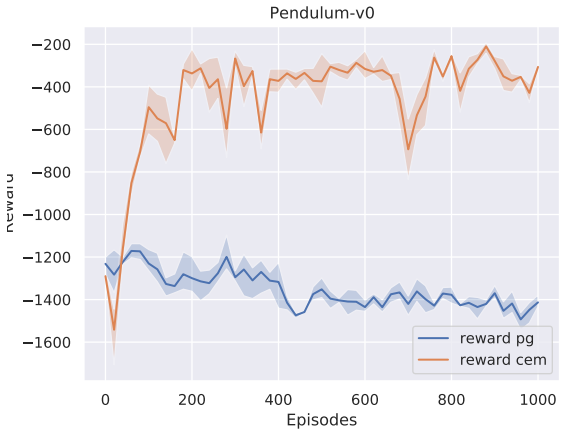
4 Expériences et résultats

4.1 Processus expérimental

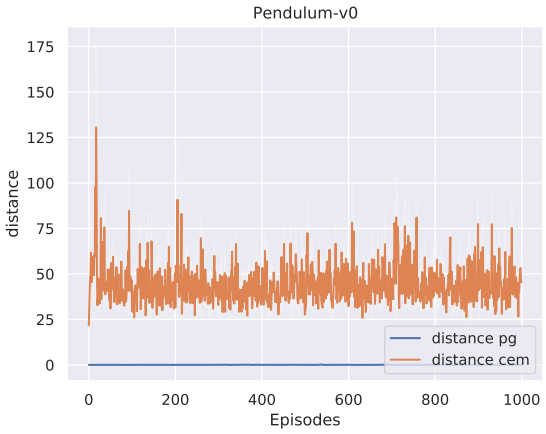
Nous avons choisi des paramètres par défaut pour PG et pour CEM. Cela permet de focaliser les résultats sur l'exploration de l'espace des politiques plutôt que sur l'optimisation des performances. On s'assure que les deux algorithmes commencent l'exploration de l'espace depuis le même point de départ : l'initialisation des poids du réseau de neurones est la même pour PG et CEM, c'est l'initialisation par défaut de PyTorch. De plus, on décide d'accorder le même budget d'apprentissage en termes de trajectoires explorées aux deux algorithmes. Ainsi, à chaque itération, CEM génère 25 nouvelles politiques chacune évaluée sur 2 trajectoires soit 50 trajectoires évaluées. PG génère 50 trajectoires à partir de la politique courante. Le taux d'apprentissage pour PG est fixé à 10^{-4} et le bruit de CEM est la matrice identité. Avec ce processus expérimental, et en répétant chaque expérience 5 fois, nous pensons avoir établi les bases d'une comparaison fiable entre les deux algorithmes.

4.2 Comparaison sur Pendulum

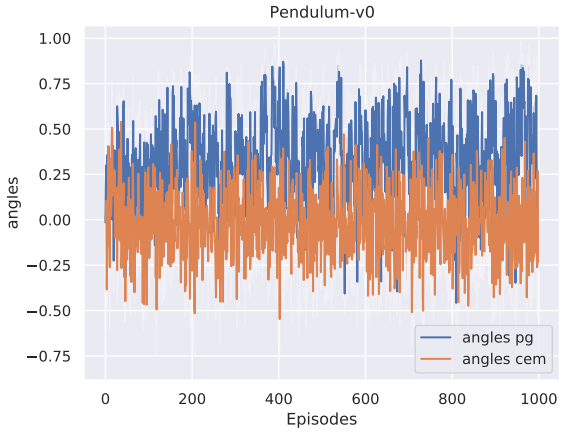
4.2.1 Comparaison des performances de CEM et PG et indicateurs de déplacement



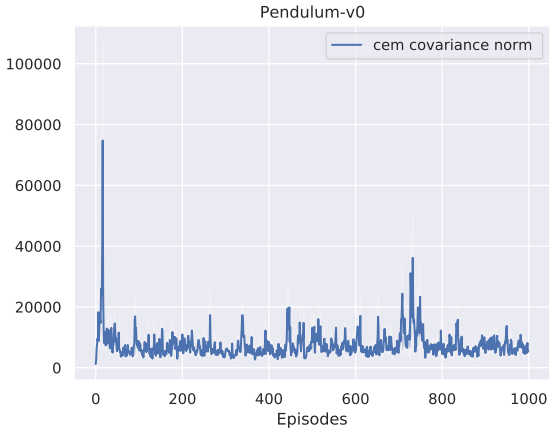
(a) Somme des récompenses



(b) Distances entre politiques apprises successivement



(c) Produit scalaire entre politiques apprises successivement



(d) Norme de la covariance utilisée pour tirer des politiques avec CEM

FIGURE 5 – Indicateurs de déplacement dans l'espace des politiques pour résoudre Pendulum

On observe sur la Figure 5a que CEM converge vers un optimum local (une somme de récompense d'environ -250) en 200 épisodes tandis que PG ne parvient pas à trouver une bonne solution même après 1000 épisodes ; et les solutions ne s'améliorent pas au cours des épisodes. Un épisode correspond à une itération d'un algorithme. La distance séparant deux politiques obtenues successivement par CEM est en moyenne 500.000 fois plus grande que la distance séparant deux politiques obtenues successivement par PG (voir Figure 5b). On rappelle que la distance séparant deux politiques obtenues successivement par CEM est de l'ordre de la longueur des demi-axes de l'ellipsoïde définie par la matrice de covariance des individus générés à chaque épisode (voir Figure 5d). Et que la distance séparant deux politiques obtenues successivement par PG est de l'ordre du taux d'apprentissage fixé à 10^{-4}. Sur la Figure 5c, on remarque aussi que les produits scalaires entre deux directions explorées successivement par CEM sont proches de 0, ce qui traduit des changements de direction fréquents et brusques. Tandis que pour PG, les

changements de direction sont aussi fréquents mais moins brusques : produits scalaires entre 0.25 et 0.75 (voir Figure 5c).

4.2.2 Visualisation de l'espace des politiques proches de l'initialisation pour Pendulum

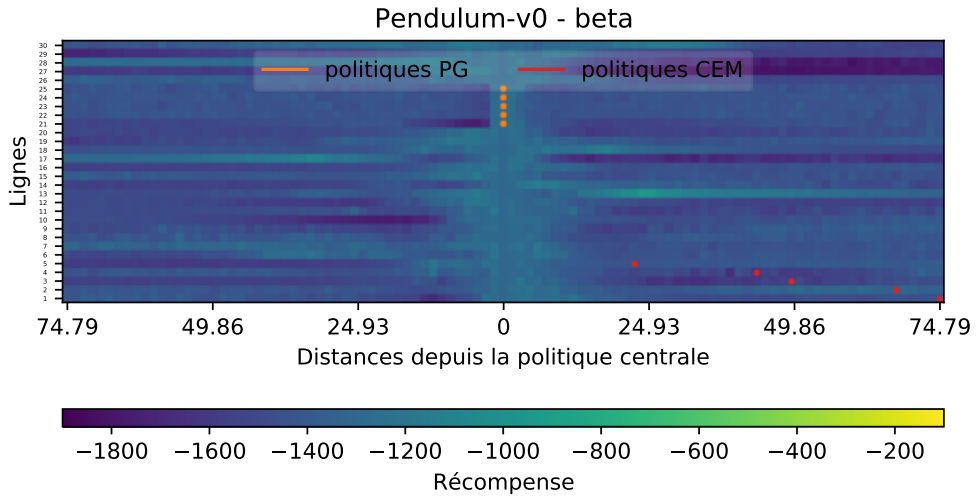


FIGURE 6 – Vignette 2D du paysage des récompenses autour des 5 premières politiques obtenues par chaque algorithme

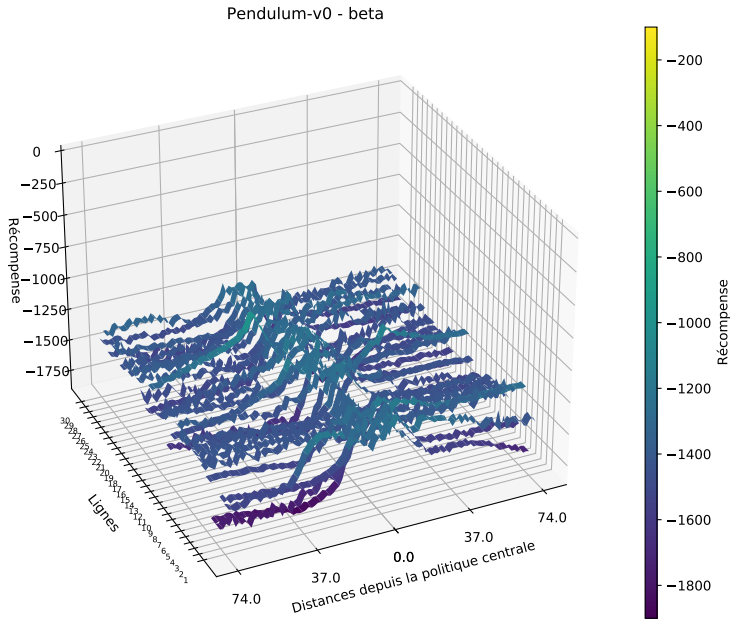


FIGURE 7 – Vignette 3D du paysage des récompenses autour des 5 premières politiques trouvées par CEM et PG

Les Vignettes permettent une visualisation 2D ou 3D du l'espace des politiques (dimensions du réseau de neurones) et des récompenses associées. Une Vignette 2D est un ensemble de pixels organisés sur plusieurs lignes. Chaque ligne est une direction tirée dans l'espace des politiques. Le pixel en son centre correspond à la politique à partir de

laquelle la direction est tirée. Les récompense associées à chaque politique de l'espace sont représentées par une échelle de couleur. Dans le cas d'une Vignette 3D, les récompenses associées aux politiques définissent la hauteur sur l'axe Z.

La Vignette 2D des 5 premières politiques obtenues par PG et CEM dans l'espace de celles-ci (voir Figure 6) confirme les résultats précédents : CEM fait de grands déplacements tandis que PG fait de petits déplacements (non-apparents à l'échelle choisie). Les politiques obtenues avec PG sont concentrées dans la zone d'initialisation. La Vignette 3D (voir Figure 7) dépeint un paysage de récompenses plat (récompenses comprises entre -1500 et -1200).

4.2.3 Visualisation de l'espace des politiques le long d'une pente améliorante trouvées avec CEM sur Pendulum

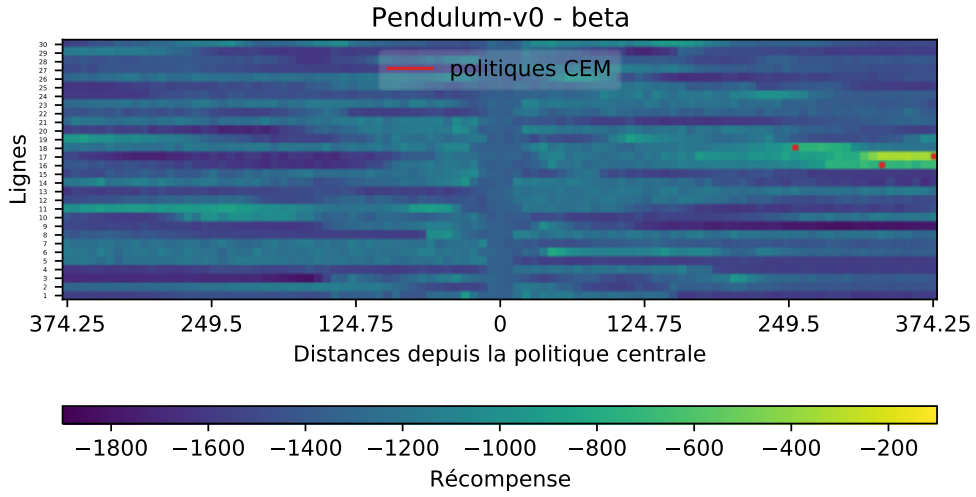


FIGURE 8 – Vignettes 2D de 3 politiques de CEM dans une pente améliorante

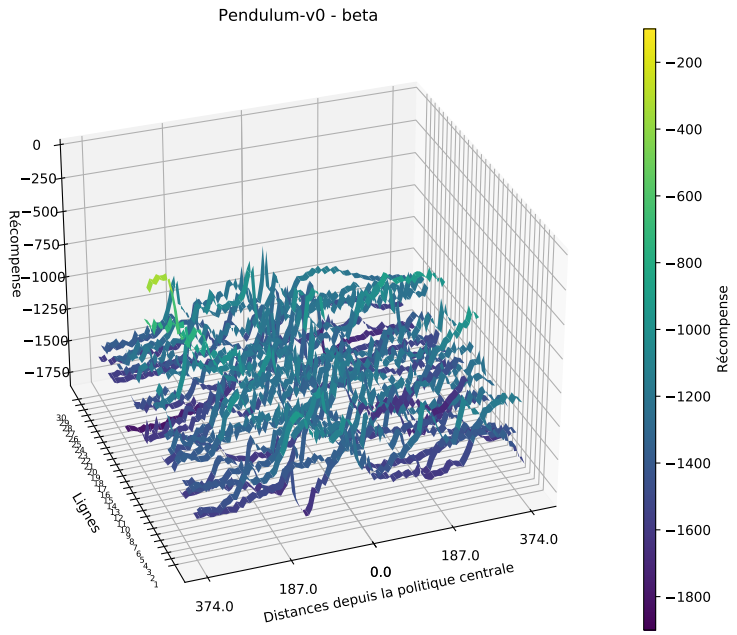


FIGURE 9 – Vignette 3D de 3 politiques de CEM dans une pente améliorante

On remarque la présence de directions le long desquelles la récompense augmente (Figure 8). Les premières politiques avec des grandes récompenses sont à une distance de 250 de la politique de départ. Les pentes le long de ces directions sont visibles sur la Figure 9.

4.2.4 Visualisation de l'espace des politiques autour d'un optimum local trouvé avec CEM sur Pendulum

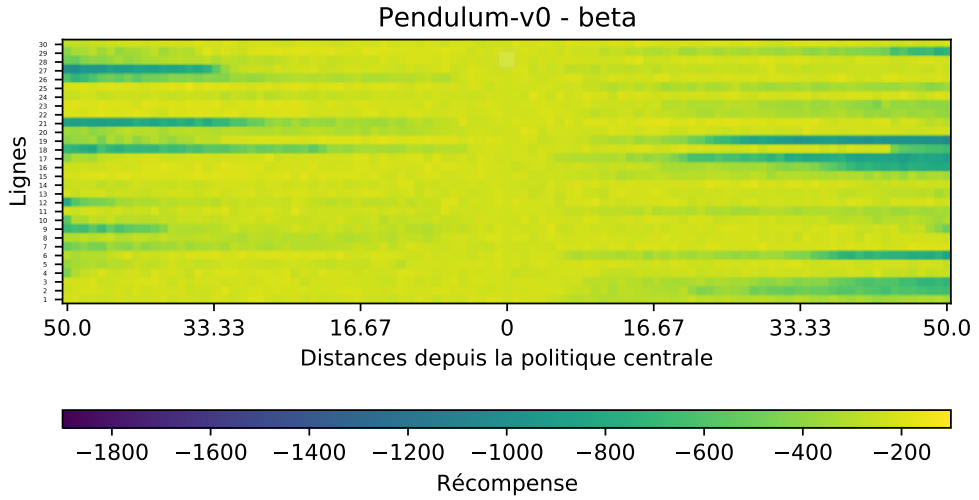


FIGURE 10 – Vignette 2D du paysage des récompenses autour de la meilleure politique obtenue avec CEM

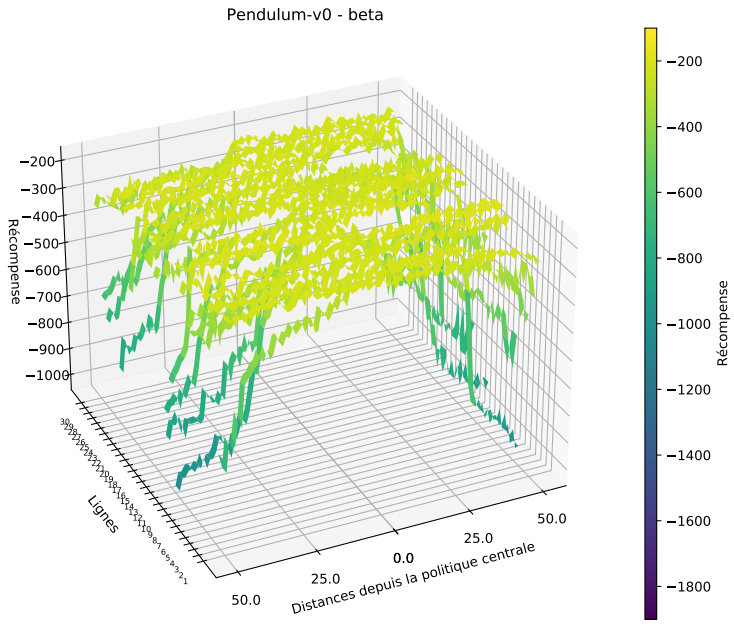
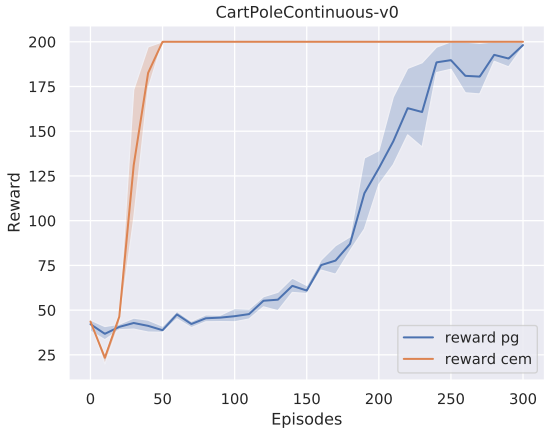


FIGURE 11 – Vignette 3D du paysage des récompenses autour de la meilleure politique obtenue avec CEM

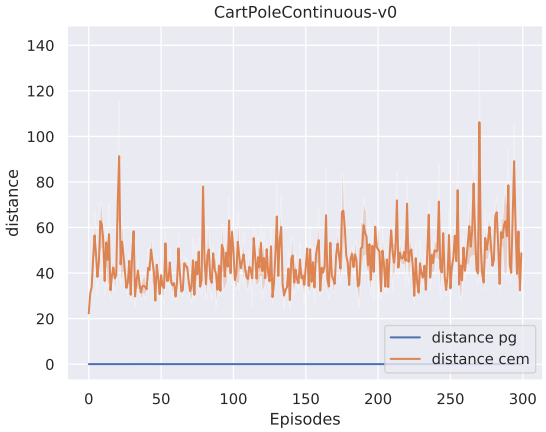
On observe, sur la vignette 2D (voir Figure 10), que l'espace des politiques comprend des grandes zones offrant de bonnes solutions. Ces zones s'apparentent à des hauts plateaux sur les vignettes 3D (voir Figure 11).

4.3 Comparaison sur CartPoleContinuous (CartPole avec domaine d'action continu)

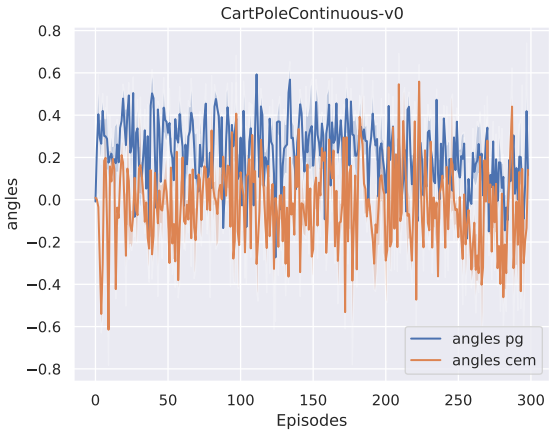
4.3.1 Comparaison des performances de CEM et PG et indicateurs de déplacement



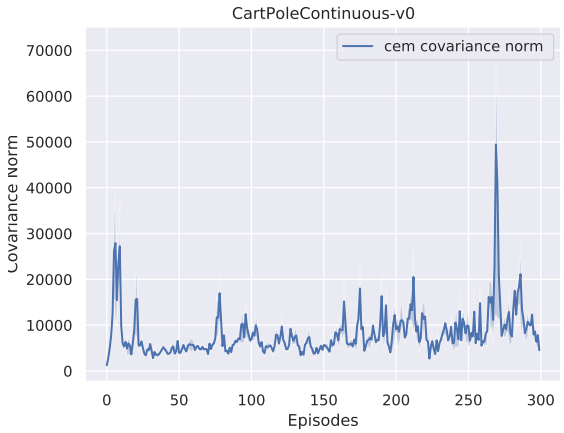
(a) Somme des récompenses



(b) Distances entre politiques apprises successivement



(c) Produit scalaire entre politiques apprises successivement



(d) Norme de la covariance utilisée pour tirer des politiques avec CEM

FIGURE 12 – Indicateurs de déplacement dans l'espace des politiques pour résoudre CartPoleContinuous

Sur CartPoleContinuous, CEM et PG convergent vers de bonnes solutions. CEM converge en 50 épisodes tandis que PG converge en plus de 300 épisodes (voir Figure 12a). Les déplacements des deux algorithmes suivent le même schéma que sur Pendulum.

4.3.2 Visualisation de l'espace des politiques proches de l'initialisation pour CartPoleContinuous

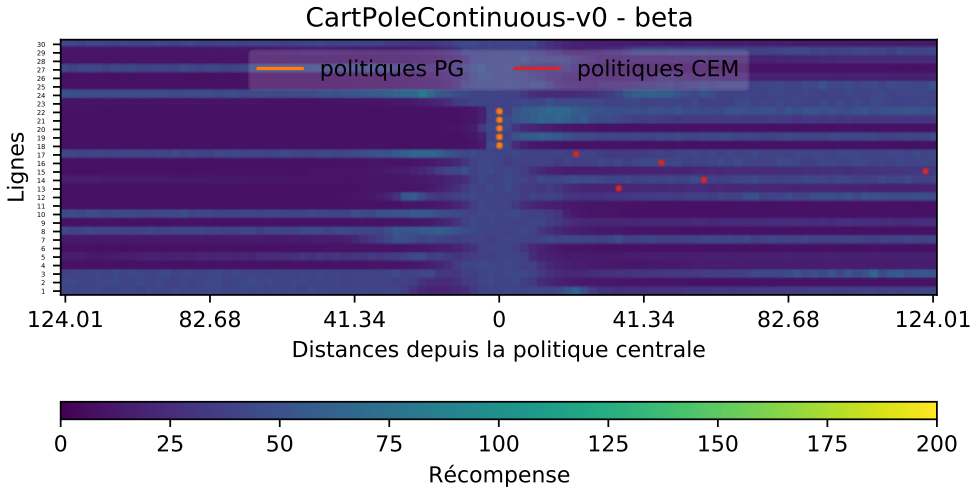


FIGURE 13 – Vignette 2D du paysage des récompenses autour des 5 premières politiques obtenues par chaque algorithme

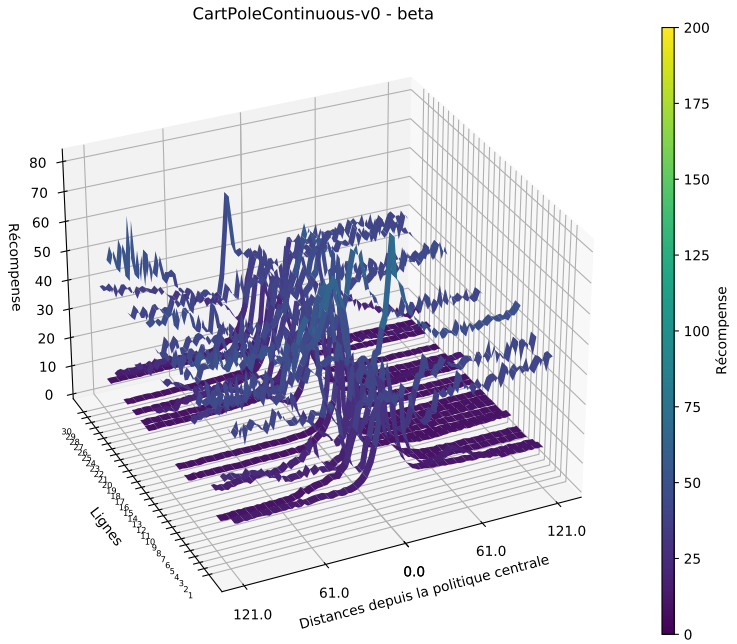


FIGURE 14 – Vignette 3D du paysage des récompenses autour des 5 premières politiques obtenues par chaque algorithme

Comme sur Pendulum, les premières politiques apprises par PG sur CartPoleContinuous sont proches de l'initialisation et celles apprises avec CEM sont plus éloignées les une des autres (voir Figure 13). Cependant, contrairement au paysage des récompenses sur Pendulum, le paysage sur CartPoleContinuous est beaucoup plus pentu et s'apparente à de multiples collines (voir Figure 14).

4.3.3 Visualisation de l'espace des politiques le long d'une pente améliorante obtenues avec PG sur CartPoleContinuous

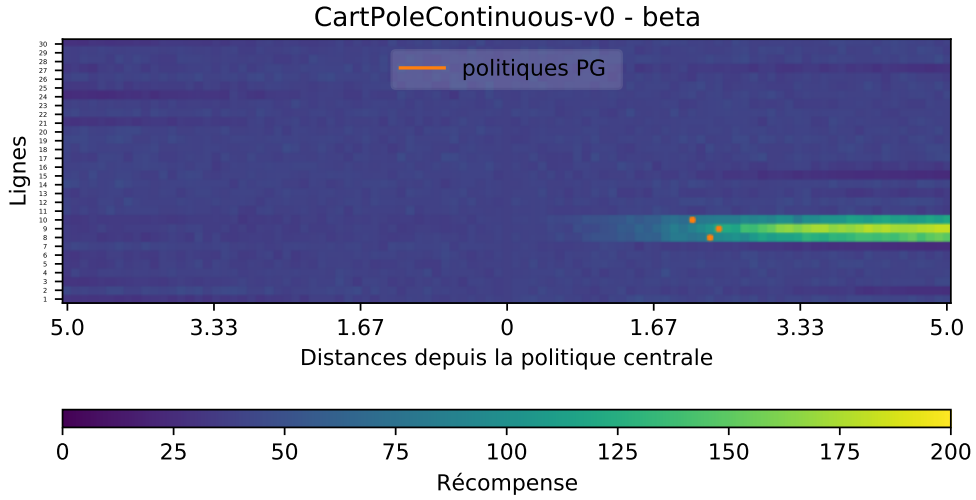


FIGURE 15 – Vignettes 2D de 3 politiques de PG dans une pente améliorante

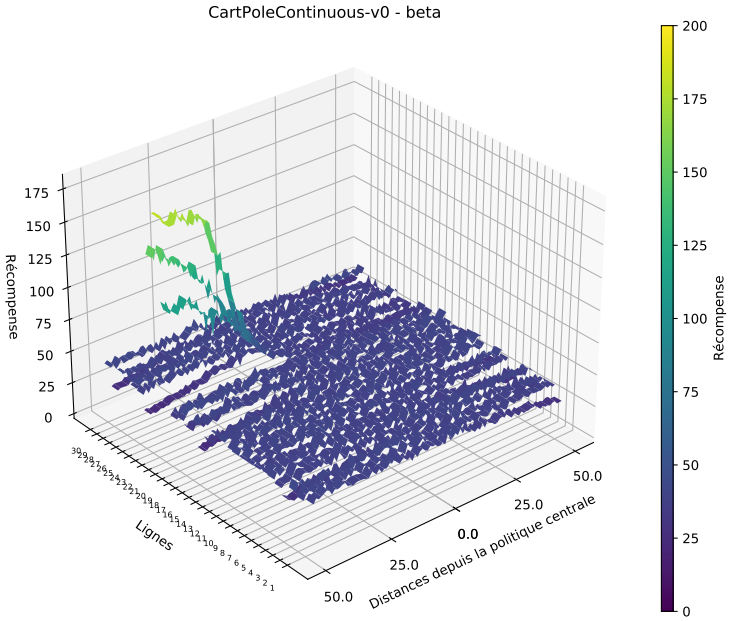


FIGURE 16 – Vignettes 3D de 3 politiques obetnues avec PG dans une pente améliorante

Sur CartPoleContinuous, il existe des pentes améliorantes proches de l'initialisation (Figures 15 et 16).

4.3.4 Visualisation de l'espace des politiques autour d'un optimum local trouvé avec CEM sur CartPoleContinuous

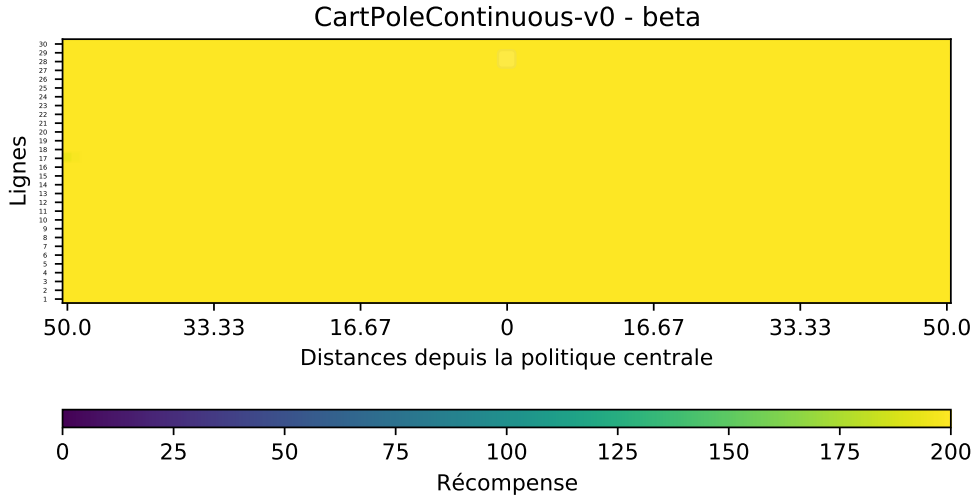


FIGURE 17 – Vignette 2D de l'espace des politiques autour de l'optimum obtenu avec CEM

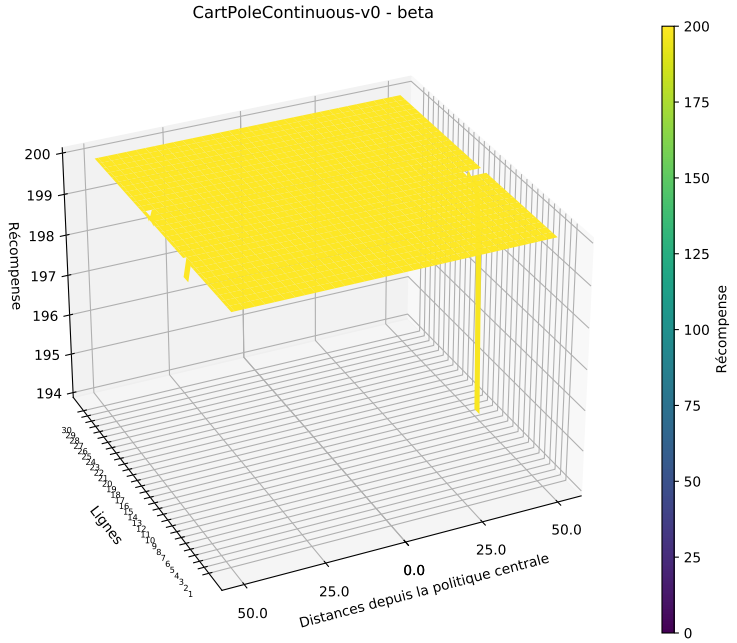


FIGURE 18 – Vignette 3D de l'espace des politiques autour de l'optimum avec CEM

Sur CartPole, le paysage autour de l'optimum obtenu avec CEM est un vaste plateau (Figures 17, 18).

5 Discussion

Nous pensons que la raison pour laquelle CEM obtient de meilleurs résultats sur Pendulum que PG, est la capacité de CEM à faire de grands déplacements dans l'espace de recherche des politiques. En effet, contrairement à PG, celui-ci est capable de s'échapper de la zone plate de l'initialisation (voir Figure 6). PG navigue dans cette zone de plat durant tout l'apprentissage, ce qui explique ses récompenses stagnantes entre -1200 et -1500 (voir Figure 5a). Cela coïncide avec les performances sur CartPoleContinuous. Sur CartPoleContinuous. , les recompenses obtenues par PG ne stagnent pas comme sur Pendulum (Figures 12a et 5a) car l'espace proche de l'initialisation comprend des pentes (Figure 14). La convergence vers une bonne solution de PG sur CartPoleContinuous s'explique par la présence de pentes améliorantes proches de l'initialisation dans l'espace des politiques (voir Figure 16). En effet, comme les pentes améliorantes sur Pendulum sont loin de la zone d'initialisation (voir Figure 9), seul CEM qui fait des grands déplacement est capable de s'y rendre. PG reste une méthode basée sur le gradient, la présence de dénivelés dans l'espace des politiques est nécessaire pour qu'il y ait apprentissage. On peut considérer PG comme un algorithme de Reward Weighted Regression (RWR). Dans ce cas, une interprétation de notre résultat est que, parmi les trajectoires générées à partir de la politique courante, aucune ne met en avant une action mieux récompensée qu'une autre pour un état donné : les politiques dans la zone de plat, et donc les trajectoires générées à partir de celles-ci, sont toutes aussi mauvaises les unes que les autres. Et donc, lors de la régression, toutes les trajectoires se voient attribuées le même poids. Un résultat surprenant est la norme de la covariance quasi-constante au cours des épisodes d'apprentissage sur Pendulum et CartPoleContinuous (voir Figure 5d et Figure 12d). En effet, on aurait pu s'attendre à ce que celle-ci diminue en fin d'apprentissage. Cela exprimerait la réduction du rayon de recherche de nouvelles solutions par CEM après que celui-ci a trouvé une politique optimale. Mais on a observé de manière explicite que les zones d'optimum sont vastes (voir Figure 10 et Figure 17). Donc, quand les individus sont générés autour de la politique courante par CEM, plusieurs d'entre eux offrent de bonnes récompenses. Et cela dans de multiples directions partant de la politique courante. Une interprétation évolutionniste est qu'il existe une certaine diversité parmi les politiques optimales. On confirme d'ailleurs ce résultat en annexe (voir Figure 19) illustrant les performances de CEM lorsque l'on fixe certaines couches du réseau de neurones.

6 Conclusion

Après étude de l'espace des politiques sur Pendulum et CartPoleContinuous, nos résultats montrent que les différences de performances entre les méthodes CEM et PG sur l'environnement Pendulum sont en partie dues à une différence d'exploration. CEM explore sur de plus grandes distances et fait des changements de directions brusques, contrairement à PG qui explore sur de faibles distances et avec des changements de directions moins brusques. Ces algorithmes étant élémentaires, cette différence d'exploration peut être étendue aux méthodes de recherche directe et aux méthodes basées sur le gradient. Il est évident que nos résultats ne suffisent pas à expliquer totalement la différence

de performance entre CEM et PG sur Pendulum. On suppose que pour que PG converge vers de bonnes solutions sur Pendulum, il faudrait que la politique initiale soit dans une zone de l'espace avec un certain dénivelé et qu'il existe un "chemin" dans l'espace des politiques menant vers une politique optimale sans passer par des zones d'optima locaux plates dans lesquelles PG resterait coincé. Il serait donc intéressant d'étudier les différentes alternatives d'initialisation et leurs conséquences sur les performances de PG sur Pendulum. Enfin, il est important de noter que nous n'avons pas optimisé les hyper-paramètres des algorithmes étudiés. Nous n'avons donc exploré que peu de conséquences des hyper-paramètres sur les performances de PG sur Pendulum. Nous avons seulement constaté que changer l'ordre de grandeur du taux d'apprentissage et/ou augmenter le nombre de trajectoires pour l'estimation du gradient n'améliore pas les performances. Il serait par exemple intéressant d'étudier les performances de PG en fonction de la structure du réseau de neurones. Enfin, nous pensons qu'il existe un continuum entre CEM et PG. À savoir, nous pensons qu'il serait possible de formuler le calcul de la politique suivante par rapport à la politique courante effectué par CEM, comme le calcul effectué par PG. L'intuition derrière cette idée est discutée en annexe A.2.

7 Remerciements

Nous remercions les élèves Sarah Kerriche, Lydia Aguini et Yannis Elrharbi-Fleury de nous avoir aidé pour la prise en main de leur code. Nous remercions notre encadrant Olivier Sigaud de nous avoir guidé tout au long du projet et de nous avoir accordé de son temps pour répondre à nos questions d'orientation.

Références

[1] Greg Brockman et al. "OpenAI Gym". In : *arXiv :1606.01540 [cs]* (5 juin 2016). arXiv : 1606.01540. URL : http://arxiv.org/abs/1606.01540 (visité le 26/03/2021).

[2] Po-Wei Chou, Daniel Maturana et Sebastian Scherer. "Improving Stochastic Policy Gradients in Continuous Control with Deep Reinforcement Learning using the Beta Distribution". In : *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN : 2640-3498. PMLR, 17 juil. 2017, p. 834-843. URL : http://proceedings.mlr.press/v70/chou17a.html (visité le 21/05/2021).

[3] Patryk Chrabaszcz, Ilya Loshchilov et Frank Hutter. "Back to Basics : Benchmarking Canonical Evolution Strategies for Playing Atari". In : *arXiv :1802.08842 [cs]* (24 fév. 2018). arXiv : 1802.08842. URL : http://arxiv.org/abs/1802.08842 (visité le 26/02/2021).

[4] Marc Peter Deisenroth. "A Survey on Policy Search for Robotics". In : *Foundations and Trends in Robotics* 2.1 (2011), p. 1-142. ISSN : 1935-8253, 1935-8261. DOI : 10.1561/2300000021. URL : http://www.nowpublishers.com/articles/foundations-and-trends-in-robotics/ROB-021 (visité le 26/03/2021).

[5] Yan Duan et al. "Benchmarking Deep Reinforcement Learning for Continuous Control". In : *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN : 1938-7228. PMLR, 11 juin 2016, p. 1329-1338. URL : http://proceedings.mlr.press/v48/duan16.html (visité le 26/03/2021).

[6] Scott Gigante et al. "Visualizing the PHATE of Neural Networks". In : *arXiv :1908.02831 [cs, stat]* (7 août 2019). arXiv : 1908.02831. URL : http://arxiv.org/abs/1908.02831 (visité le 26/03/2021).

[7] Shie Mannor, Reuven Y. Rubinstein et Yohai Gat. "The Cross Entropy Method for Fast Policy Search". In : ICML. 1er jan. 2003. URL : https://openreview.net/forum?id=rkbyRo-_ZH (visité le 26/03/2021).

[8] Paolo Pagliuca, Nicola Milano et Stefano Nolfi. "Efficacy of Modern Neuro-Evolutionary Strategies for Continuous Control Optimization". In : *Frontiers in Robotics and AI* 7 (28 juil. 2020). ISSN : 2296-9144. DOI : 10.3389/frobt.2020.00098. URL : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7805676/ (visité le 26/02/2021).

[9] Jan Peters et Stefan Schaal. "Reinforcement learning by reward-weighted regression for operational space control". In : *Proceedings of the 24th international conference on Machine learning*. ICML '07. New York, NY, USA : Association for Computing Machinery, 20 juin 2007, p. 745-750. ISBN : 978-1-59593-793-3. DOI : 10.1145/1273496.1273590. URL : https://doi.org/10.1145/1273496.1273590 (visité le 26/02/2021).

[10] Tim Salimans et al. "Evolution Strategies as a Scalable Alternative to Reinforcement Learning". In : *arXiv :1703.03864 [cs, stat]* (7 sept. 2017). arXiv : 1703.03864. URL : http://arxiv.org/abs/1703.03864 (visité le 26/02/2021).

[11] Olivier Sigaud et Freek Stulp. "Policy Search in Continuous Action Domains : an Overview". In : *arXiv :1803.04706 [cs]* (13 juin 2019). arXiv : 1803.04706. URL : http://arxiv.org/abs/1803.04706 (visité le 26/02/2021).

[12] Richard S. Sutton et Andrew G. Barto. *Reinforcement Learning, second edition : An Introduction*. Google-Books-ID : uWV0DwAAQBAJ. MIT Press, 13 nov. 2018. 549 p. ISBN : 978-0-262-35270-3.

[13] István Szita et András Lörincz. "Learning Tetris Using the Noisy Cross-Entropy Method". In : *Neural Computation* 18.12 (1er déc. 2006), p. 2936-2941. ISSN : 0899-7667. DOI : 10.1162/neco.2006.18.12.2936. URL : https://doi.org/10.1162/neco.2006.18.12.2936 (visité le 26/03/2021).

[14] Tingwu Wang et al. "Benchmarking Model-Based Reinforcement Learning". In : *arXiv :1907.02057 [cs, stat]* (3 juil. 2019). arXiv : 1907.02057. URL : http://arxiv.org/abs/1907.02057 (visité le 26/03/2021).

[15] Nir Ben Zrihem, Tom Zahavy et Shie Mannor. "Visualizing Dynamics : from t-SNE to SEMI-MDPs". In : *arXiv :1606.07112 [cs, stat]* (22 juin 2016). arXiv : 1606.07112. URL : http://arxiv.org/abs/1606.07112 (visité le 26/03/2021).

A Annexes

A.1 CEM et PG avec couches fixées

FIGURE 19 – Performance des algorithmes étudiés et de leurs variantes avec couches fixées sur Pendulum

Pour obtenir cette figure (Figure 19), nous avons d'abord initialisé le réseau de neurones de la même manière que dans les expériences principales. Puis nous avons fixé les poids de toutes les couches sauf ceux de la dernière. On observe que CEM avec couches fixées trouve quand même des politiques avec des scores proches de -250. Cela signifie que, même dans une portion restreinte de l'espace de recherche de politiques, il existe de bonnes politiques. Il existe donc plusieurs zones de l'espace comportant des optima et donc il y a une diversité des comportements permettant de maintenir le pendule en équilibre.

A.2 Continuum entre PG et CEM

PG est une méthode basée sur le gradient : en théorie, si le gradient est estimé avec une infinité de trajectoires générées avec la politique courante, celui-ci doit correspondre à la direction de la plus grande amélioration. L'intuition est la suivante : pour CEM, si on fait tendre les valeurs propres de la covariance vers $\epsilon > 0$, si on génère une infinité d'individus dans l'ellipse de demi-axes ϵ autour de la politique courante, et si l'on ne prend que le meilleur individu généré pour calculer la politique suivante, alors, la direction entre la politique courante et la politique suivante correspond elle aussi à la direction du gradient. Si cela s'avérait, on pourrait considérer PG et CEM comme deux instances d'une même classe de méthode. Le paramètre qui change d'une instance à l'autre serait alors le rayon de la boule autour de la politique courante dans l'espace des politiques, dans laquelle ces instances tirent des politiques et génèrent des trajectoires pour estimer la direction dans laquelle explorer. Un CEM classique tire alors dans un rayon défini par les valeurs propres de la covariance tandis que PG tire dans un rayon infiniment petit.