Cartes topographiques neuronales pour l'apprentissage par renforcement sur des problèmes de contrôle non-linéaire

Emmanuel Daucé, Alain Dutech

Cartes topographiques neuronales

Cartes topographiques neuronales pour l'apprentissage par renforcement sur des problèmes de contrôle non-linéaire

Emmanuel Daucé - Institut des sciences du mouvement, UMR 6233;
Faculté des sciences du sport, Université de la Méditerrannée
163 avenue de Luminy, CP 910,13288 Marseille cedex 9, FRANCE
edauce@ec-marseille.fr

Alain Dutech - Equipe MAIA - LORIA/INRIA
Campus Scientifique, BP 239, 54506 Vandoeuvres-les-Nancy, FRANCE
Alain.Dutech@loria.fr

22 juin 2010

Résumé

Nous présentons une architecture neuronale où sont combinés une nouvelle règle d'apprentissage par renforcement et un codage de l'information topographique inspiré de méthodes à noyaux. Cette architecture est capable d'apprendre à contrôler des systèmes non-linéaires définis sur des espaces continus. Nous présentons les premiers résultats obtenus dans une tâche d'atteinte de cible visuelle.

We present a neural architecture which combines a new reinforcement learning algorithm with a topographic encoding of the inputs as inspired by kernel-based methods. This architecture is able to learn to control non-linear systems defined on a continuous space. Some results on a task of reaching are also given.

keywords : AI, Neural Control, Reinforcement Learning, Kernel Methods

1 Introduction

Apprendre automatiquement à contrôler un système dans le but de l'amener ou de le maintenir dans un état voulu est une tâche générique de l'intelligence artificielle.

Nous nous intéressons ici à une classe de problèmes réputée difficile dans laquelle une fonction inconnue de l'espace des entrées vers l'espace des sorties

doit être apprise sur la base de simples signaux de récompenses (ce qui permet de se passer d'exemples explicites pourtant nécessaires dans le cas de l'apprentissage supervisé [Cornuejols and Miclet, 2002]). C'est le cadre de l'apprentissage par renforcement (AR) [Sutton and Barto, 1998]. La difficulté principale de ce type de problème vient du fait que le contrôleur doit à la fois apprendre les régularités de l'environnement et les commandes qui lui assurent la meilleure récompense. Les méthodes classiques issues de la programmation dynamiques reposent souvent sur une discrétisation de l'espace d'état et sont peu adaptées à des problèmes de grande dimension ou à des données continues.

Nous présentons ici une architecture de contrôle pour laquelle aucune étape de discrétisation n'est nécessaire. Les donnée fournies au contrôleur ainsi que les commandes produites sont à valeurs réelles. Nous nous plaçons en contrepartie dans le cadre d'un contrôle *linéaire*. Mais, afin de pouvoir traiter des problèmes non-linéaires, nous nous inspirons des méthodes à noyau [Schölkopf and Smola, 2002] en déployant le problème initial sur un espace de plus grande dimension avant de lui appliquer la technique de régression adéquate.

Notre approche est partiellement inspirée par l'étude du système nerveux et présente deux caractéristiques importantes. D'une part, nous dérivons une *nouvelle* règle d'apprentissage par renforcement locale à chaque neurone permettant au système neuronal dans son ensemble d'apprendre une politique de contrôle globale. Ainsi que nous le détaillons en section 2, cette règle est inspirée de travaux de Williams [Williams, 1992] et Baxter [Bartlett and Baxter, 1999, Baxter and Bartlett, 2000] sur des neurones à sortie binaires. Ici, nous mettons en œuvre cette même approche avec des neurones linéaires perturbés par un bruit gaussien.

D'autre part, les entrées du contrôleur sont recodées sur une base de fonctions à base radiale. Ce codage dit "topographique" de l'information est inspiré par l'organisation spatiale du cortex, où l'information sensorielle est traduite sous forme de patrons d'activité spatialement localisés, ce qui permet d'accroître la robustesse et, comme nous le verrons, de traiter des problèmes non linéaires. Les détails de ce type de codage sont donnés en section 3.

L'architecture de contrôle que nous présentons est ensuite testée sur un problème d'atteinte de cible, où l'information fournie au contrôleur est déformée par une rétine logaritmique. Le but de ces premiers tests est de valider les concepts mis en œuvre. Pour cela, nous avons notamment comparé les performances du codage topographique à celui d'un contrôleur linéaire classique, et montrons que le codage topographique permet de contourner le problème de la non-linéarité des entrées. Ces résultats sont détaillés en section 5 avant de conclure cet article.

2 Contrôleur neuronal à neurones linéaires stochastiques

On considère un "contrôleur" neuronal soumis à une entrée multidimensionnelle $\mathbf{x}$ et produisant une réponse scalaire ou vectorielle $\mathbf{y}$. Le but est d'apprendre

la commande – une fonction inconnue de l'espace d'entrée vers l'espace de sortie $\mathbf{y} = f(\mathbf{x})$ – en s'appuyant uniquement sur un signal de récompense scalaire R.

Les neurones du contrôleur sont "stochastiques" et "linéaires" dans le sens où leur réponse moyenne dépend linéairement de la somme pondérée de leurs entrées. L'activité d'un neurone est paramétrée par le vecteur de poids $\mathbf{W}$. Ainsi le potentiel h d'un neurone vaut $h = \mathbf{W}\mathbf{x}^T$ où $\mathbf{W}$ est un vecteur de poids et $\mathbf{x}$ est le vecteur constitué des entrées du neurone.

La réponse finale du neurone est la somme de ce potentiel et d'une variable aléatoire obéissant à une loi normale centrée d'écart type σ. La réponse d'un neurone vaut donc $y = h + \mathcal{N}(0, \sigma^2) = \mathbf{W}\mathbf{x}^T + \mathcal{N}(0, \sigma^2)$.

2.1 Algorithme d'apprentissage

Au cours de l'apprentissage, les paramètres des neurones sont modifiés pour améliorer la réponse globale du contrôleur. Les poids des neurones sont mis à jour à chaque pas de temps selon le principe de Policy Gradient de [Williams, 1992], i.e :

$$\mathbf{W}_t = \mathbf{W}_{t-1} + \alpha r_t \nabla_{\mathbf{W}_{t-1}} \log P_{\mathbf{W}_{t-1}}(y_t|\mathbf{x}_t) \tag{1}$$

où $\alpha \in]0, 1[$ est le coefficient d'apprentissage et ∇ l'opérateur gradient.

Explicitons cette règle de mise à jour. Si une variable aléatoire z suit une loi normale $\mathcal{N}(m, \sigma^2)$, sa densité de probabilité s'exprime par $\Pr(z) = \frac{1}{\sigma}\Phi(z - m)$ où $\Phi(z) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{1}{2}z^2)$. On a alors $\nabla_z \Phi(z) = -z.\Phi(z)$.

En utilisant cette propriété, on obtient que

$$\nabla_{\mathbf{W}} \log P_{\mathbf{W}}(y|\mathbf{x}) = \frac{\nabla_{\mathbf{W}} P_{\mathbf{W}}(y|\mathbf{x})}{P_{\mathbf{W}}(y|\mathbf{x})} \qquad = \frac{\nabla_{\mathbf{W}}(y - \mathbf{W}\mathbf{x}^T)\nabla_y \mathcal{N}(\mathbf{W}\mathbf{x}^T, \sigma^2)}{P_{\mathbf{W}}(y|\mathbf{x})} \tag{2}$$

$$= \frac{\mathbf{x}\frac{(y-\mathbf{W}\mathbf{x}^T)}{\sigma^2} P_{\mathbf{W}}(y|\mathbf{x})}{P_{\mathbf{W}}(y|\mathbf{x})} \qquad = \frac{(y-h)\mathbf{x}}{\sigma^2} \tag{3}$$

.

La règle d'apprentissage s'écrit donc :

$$\mathbf{W}_t = \mathbf{W}_{t-1} + \alpha \frac{r_t}{\sigma^2}(y - h)\mathbf{x}. \tag{4}$$

2.2 Propriétés de la règle d'apprentissage

Les méthodes de Policy Gradient sont des méthode de gradient stochastique, et la règle que nous proposons ici présente donc des propriétés de convergence analogues à celle du perceptron linéaire (convergence vers le minimum au sens des moindres carrés). Le bruit participe ici au guidage de l'algorithme de gradient, et la vitesse de convergence de l'algorithme est directement liée son écart-type : plus le bruit est faible, plus l'algorithme converge lentement, mais

plus il est précis. Il n'y a pas de contrainte particulière concernant le signal de récompense : celui-ci doit être en adéquation avec la tâche que l'on souhaite accomplir.

3 Projection des entrées sur une carte topographique

Dès lors que la fonction à estimer est non-linéaire (ce qui correspond aux cas intéressants), il convient de mettre en œuvre une stratégie visant à prendre en compte les déformations.

Nous proposons ici une stragégie développée dans la communauté de machines à vecteurs supports consistant à projeter les données d'entrée sur un espace de plus grande dimension sur lequel l'existence d'une dépendance linéaire des entrées aux sorties est très probable. Cet accroissement de dimension est cependant maîtrisé grâce aux propriétés des fonctions noyau utilisées [Schölkopf and Smola, 2002].

3.1 Codage topographique de l'information

Une carte topographique représente l'information de manière spatiale : chaque entrée $\mathbf{x}$ est projetée sur un espace de fonctions à base radiale. Nous présentons ici une version discrète de cette projection pour des entrées de dimension 2.

Nous utilisons une carte de neurones topologiquement organisés selon une grille à deux dimensions. Chaque neurone i a un champ récepteur distinct centré en (x_1^i, x_2^i) et les champs récepteurs sont répartis régulièrement sur la carte en deux dimensions. Lorsqu'un stimulus (x_1, x_2) est présenté, chaque neurone perceptif se voit attribuer une activité selon la fonction noyau : $y^i = K((x_1, x_2) - (x_1^i, x_2^i))$ (projection d'un signal (x_1, x_2) sur une base de fonctions noyaux). Les noyaux typiquement choisis sont des noyaux gaussiens (voir fig. 1,gauche).

3.2 Propriétés du codage topographique

Par rapport à un codage scalaire (voir figure 1 (droite)), le codage topographique apporte une plus grande robustesse : l'information n'est plus codée par un seul neurone mais par une population de neurones, si l'un ou plusieurs d'entre eux sont défectueux, l'information reste tout de même présente sur la carte topographique.

Si l'on emploie un codage scalaire, la précision de l'information est liée à l'intervalle des valeurs possibles pour le neurone alors que dans une carte topographique, cette précision est liée au nombre de neurones de la carte. D'un point de vue "biologique", l'intervalle des valeurs possibles d'un neurone étant fini, une carte neuronale est aussi un moyen de coder plus précisément l'information.

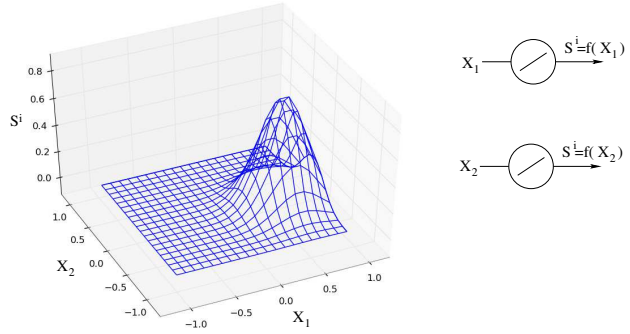

FIGURE 1 – **Codage topographique et scalaire**. Un stimulus en $(0.55, 0.25)$ est représenté par plusieurs neurones dont les champs récepteurs sont gaussiens dans un codage topographique (à gauche) et par deux neurones de valeurs $f(0.55)$ et $f(0.25)$ dans un codage scalaire (à droite).

4 Application à l'orientation vers une cible visuelle

La tâche que nous avons utilisée pour valider notre approche est inspirée de l'apprentissage de saccades visuelles vers une cible en mouvement. Cette tâche possède comme caractéristique de faire intervenir une transformation non-linéaire liée au caractère anisotrope de la rétine, avec une résolution importante au centre de la rétine (fovea) et une resolution faible à la périphérie. Cette tâche reste simple et permet de bien illustrer les propriétés de notre algorithme.

Notre système doit apprendre à contrôler les déplacements d'un système pour atteindre une cible en déplacement. Ce concept est modélisé par le fait que les entrées de notre contrôleurs seront des n-uplets $(x_1, x_2, \dot{x}_1, \dot{x}_2)$ où (x_1, x_2) représente la position de l'objet et $(\dot{x}_1, \dot{x}_2)$ sa vitesse. En sortie, le contrôleur doit produire un signal de commande (u_1, u_2) permettant de déplacer le système. Dans notre cas le but du contrôleur est de produire une réponse la plus proche possible de la position finale de la cible, à savoir $c_1 = x_1 + \dot{x}_1 \Delta t$ et $c_2 = x_2 + \dot{x}_2 \Delta t$ où $\Delta t = 500$ ms.

4.1 Entrée du système

Nous considérons deux modes de codage différents du signal d'entrée pour notre système.

- **Codage scalaire**. La couche d'entrée est constituée de 4 neurones : 2 neurones dits "P" codant la position $S^{(P)} = (x_1, x_2)$, 2 neurones "V" codant la vitesse : $S^{(V)} = (\dot{x}_1, \dot{x}_2)$.
- **Codage topographique**. La couche d'entrée constituée de 2 cartes topographiques de 16×16 neurones :

– 256 neurones pour la couche "P". Coordonnées des neurones : entre -7.5/8 (-0.9375) et 7.5/8 (0.9375) par pas de 1/8 (0.125) en abcisse et ordonnée. Le rayon du noyau gaussien de chacun des neurones est $\rho = 0.125$.
– 256 neurones pour la couche "V". Coordonnées des neurones : idem entre -7.5/8 et 7.5/8 par pas de 1/8 en abcisse et ordonnée. Rayon du noyau guassien des neurones est $\rho = 0.125$.

Nous avons considéré le cas d'une rétine "linéaire" où la position subjective (x_1, x_2) est directement transmise au contrôleur, et une rétine "logarithmique" pour laquelle la position $(x_1 = r.\cos(\theta), x_2 = r.\sin(\theta))$ est déformée en $(\tilde{r}cos(\theta), \tilde{r}sin(\theta))$ avec $\tilde{r} = \frac{\log(1+B.r)}{log(1+B)}$ et $B = 4$.

4.2 Sortie du système

Les neurones P et V projettent directement sur les neurones de la couche de sortie, que nous appelons les neurones moteurs ("M"). Il y a 4 neurones moteurs codant respectivement pour les directions (Haut,Gauche,Bas,Droite) autrement dit $(\pi/2, \pi, 3\pi/2, 2\pi)$.

Le signal des couches sensorielles est projeté sur les neurones moteurs :

$$\mathbf{h}^{(M)} = W^{(P)}\mathbf{S}^{(P)} + W^{(V)}\mathbf{S}^{(V)}.$$

Comme précisé en section 2, la sortie des neurones moteurs est le résultat d'un tirage aléatoire $\forall i, S_i^{(M)} \sim h_i^{(M)} + \mathcal{N}(0, \sigma)$ où $\sigma = 0.01$.

La commande motrice finale est la combinaison des sorties des 4 neurones.

$$u_1 = \sum_{i=1}^{4} S_i^{(M)} \cos(i\pi/2) \qquad u_2 = \sum_{i=1}^{4} S_i^{(M)} \sin(i\pi/2).$$

4.3 Apprentissage

A chaque itération du système, un 4-uplet d'entrée est tiré aléatoirement selon la loi normale $\mathcal{N}(0, 0.3)$. A l'aide de cette entrée, le contrôleur produit un déplacement (u_1, u_2) qui fait apparaître la cible sur la rétine à la position $e = (e_1, e_2)$. La distance de cette position au centre du repère $(0, 0)$ est une mesure de l'erreur et permet de générer un signal de récompense immédiat $R = 0.2 - |e|$.

Avec un coefficient $\alpha = \frac{0.01 \times \sigma^2}{16}$, on met à jour les poids selon :

$$\mathbf{W}_t^{(P)} = \mathbf{W}_{t-1}^{(P)} + \frac{\alpha R_t}{\sigma^2}(\mathbf{S}_t^{(M)} - \mathbf{W}_{t-1}^{(P)}{\mathbf{S}_t^{(P)}}^T)\mathbf{S}_t^{(P)}$$
$$\mathbf{W}_t^{(V)} = \mathbf{W}_{t-1}^{(V)} + \frac{\alpha R_t}{\sigma^2}(\mathbf{S}_t^{(M)} - \mathbf{W}_{t-1}^{(V)}{\mathbf{S}_t^{(V)}}^T)\mathbf{S}_t^{(V)}.$$

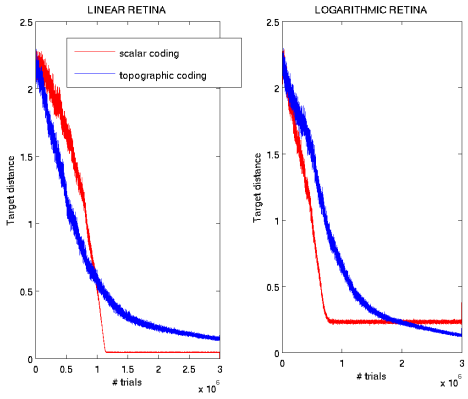

FIGURE 2 – **Erreur d'apprentissage**. Evolution des erreurs d'apprentissage (distance finale à la cible après le mouvement) en fonction du nombre d'itérations. Dans les deux cas, le codage topographique (linéaire à gauche et logarithmique à droite) est comparé au codage scalaire.

5 Résultats et discussion

La figure 2 permet de voir l'évolution des erreurs d'apprentissage pour les différents types de codage pendant 3.10^6 exemples. On y indique une erreur moyenne calculée sur une fenêtre glissante sur les 500 dernières itérations. L'avantage d'un codage topographique n'est bien sûr manifeste que dans le cas d'une rétine logaritmique. En effet, dans ce cas, le contrôleur linéaire atteint sa limite de précision du fait du caractère non-linéaire de la transformation opérée. Dans le cas du contrôleur topographique, il reste cependant une erreur résiduelle qui provient des grandes saccades qui sont sous-représentées dans la base d'apprentissage. Il reste donc une marge de progression.

Pour estimer la précision des deux contrôleurs, on mesure la réponse de chaque contrôleur à des saccades de différentes amplitudes, de différentes directions avec une vitesse nulle ou positive. La figure 3 donne une idée de cette précision en indiquant avec une croix les résultats désirés. La réponse du contrôleur élémentaire à codage scalaire donne le résultat attendu : les petites saccades sont sur-estimées, les grandes saccades sont sous-estimées. Seuls les exemples situés dans une zone linéaire du logarithme sont correctement apprises. L'estimation du déplacement lié à la vitesse est correct. Dans le cadre du contrôleur à codage topographique, La réponse est précise pour les petites saccades, un peu surestimée pour les saccades moyennes et sous-estimée pour les grandes saccades. L'estimation du déplacement lié à la vitesse est lui-aussi correct.

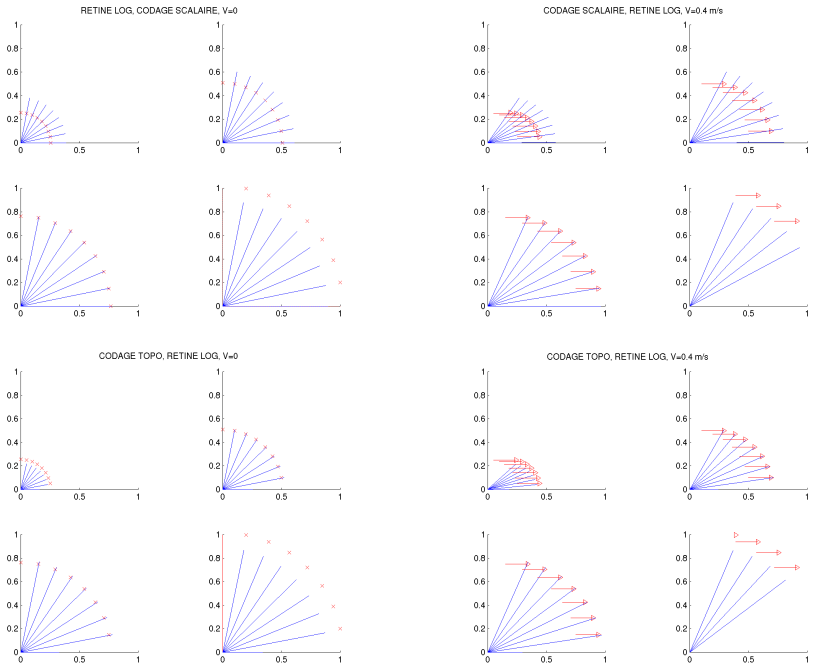

FIGURE 3 – **Précision du contrôle**. Après avoir appris, on teste les différents contrôleurs avec des saccades de différentes amplitudes avec une vitesse nulle (à gauche) ou positive (à droite) pour un codage scalaire (haut) et toppgraphique (bas). Les croix indiquent les déplacements désirés. Le bruit de sortie est nul.

6 Conclusion

Nous avons présenté une architecture de contrôle neuronale qui s'appuie essentiellement sur une loi d'apprentissge par renforcement originale et locale à chaque neurone linéaire stochastique du réseau. Notre architecture utilise un codage topographique des informations inspiré de l'organisation du cortex et qui permet une plus grande robustesse dans la représentation. Les concepts mis en œuvre dans cette architecture ont été testés sur un exemple de contrôle simple mais assez générique. Les résultats obtenus, bien que préliminaires, nous confortent dans l'idées d'explorer plus avant le potentiel de cette architecture.

Les perspectives offertes par nos travaux sont nombreuses et de plusieurs sortes, notamment en ce qui concerne l'exploration des concepts utilisés. Pour cela, nous allons utiliser des récompenses moins informatives voire binaires (tâche réussie ou non) et retardées dans le temps. Nous allons aussi essayer de mieux comprendre l'influence de l'aléa sur les capacités d'exploration et de vitesse de convergence de l'apprentissage. En ce qui concerne la structure neuronale, nous testerons l'apport d'un codage en sortie avec plus de neurones et la

possibilité d'utiliser des neurones dans une couche cachée. Enfin, il sera important de confronter notre architecture à des problèmes plus complexes comme, par exemple, le contrôle d'un bras avec plusieurs degrés de liberté.

Remerciements - Cet article est le fruit d'une coopération au sein de l'ANR MAPS.

Références

[Bartlett and Baxter, 1999] Bartlett, P. and Baxter, J. (1999). Hebbian synaptic modifications in spiking neurons that learn. Technical report, The Australian National University, Canberra, Australia.

[Baxter and Bartlett, 2000] Baxter, J. and Bartlett, P. (2000). Reinforcement learning in POMDP's via direct gradient ascent. In *Proc. 17th International Conf. on Machine Learning (ICML'00)*.

[Cornuejols and Miclet, 2002] Cornuejols, A. and Miclet, L. (2002). *Apprentissage artificiel - Concepts et Algorithmes*. Eyrolles.

[Schölkopf and Smola, 2002] Schölkopf, B. and Smola, A. (2002). *Learning with kernels*. MIT Press, Cambridge, MA.

[Sutton and Barto, 1998] Sutton, R. and Barto, A. (1998). *Reinforcement Learning*. Bradford Book, MIT Press, Cambridge, MA.

[Williams, 1992] Williams, R. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8 :229–256.