Procesus de decision de Markov

Calic Petar, Muyang Shi
Sorbonne Universite, Paris
Encadré par Emmanuel Hyon

petarcalic99@gmail.com
muyangforwork@gmail.com

Abstract

A short (up to 250 words) summary of the main contributions of the paper and the context of the research. Full length papers discuss and illustrate methods, challenges, and limitations in the creation, collection, management, access, processing, or analysis of data in humanities research, including standards and formats. This template provides a general outline for full length papers and authors can adapt the headings and include subheadings as they find appropriate.

Keywords: keyword 1; keyword 2; lower case except names, max 6

Author roles: For determining author roles, please use following taxonomy: https://casrai.org/credit/. Please list the roles for each author.

1 Introduction

En théorie des probabilités, un processus de décision markovien (MDP) est un modèle stochastique (de fonction aléatoire) où un agent prend des décisions et où les résultats de ses actions sont aléatoires. Les MDPs sont une extension des chaînes de Markov avec plusieurs actions à choisir par état et où des récompenses sont gagnées par l'agent. Les MDPs sont utilisés pour étudier des problèmes d'optimisation à l'aide d'algorithmes de programmation dynamique ou d'apprentissage par renforcement, que nous allons étudier et comparer.
Tout d'abord il faut savoir reconnaître les situations dans lesquelles un problème peut être modélisé par un MDP. Ensuite il va falloir savoir modéliser un problème sous forme de MDP, adapté au solveur qu'on souhaite utiliser. L'objectif principal de ce projet est de comparer 3 solveurs de MDP. Nous allons observer les performances de Gurobi, M.. et le solveur Toulousain. Nous allons faire tourner les 3 solveurs sur un même problème et observer les résultats pour ensuite comparer leurs caractéristiques. A la fin, Nous allons déterminer aussi pour chaque type de problème, quel solveur serait le plus adapté et performant.

2 Explication des MDP

2.1 Définition formelle

Un MDP est un quadruplet S, A,T, R définissant :

- un ensemble d'états S, qui peut être fini, dénombrable ou continu; cet ensemble définit l'environnement tel que perçu par l'agent (dans le cas d'un robot, on peut voir cela comme l'ensemble produit des valeurs de ses différents capteurs);
- un ensemble d'actions A, qui peut être fini, dénombrable ou continu et dans lequel l'agent choisit les interactions qu'il effectue avec l'environnement (dans le cas d'un robot on peut voir cela comme l'ensemble produit des paramètres de ses différentes commandes);
- une fonction de transition T : S x A x S $\rightarrow$ [0;1]; cette fonction définit l'effet des actions de l'agent sur l'environnement: T(s, a, s') représente la probabilité de se retrouver dans l'état s' en effectuant l'action a, sachant que l'on était à l'instant d'avant dans l'état s;
- une fonction de récompense R : S x A x S x $\Re$ $\rightarrow$ [0;1]; elle définit la récompense (positive ou négative) reçue par l'agent: R(s,a,s',v) est la probabilité d'obtenir une récompense v pour être passé de l'état s à s' en ayant effectué l'action a. Ici encore cette définition est très générale, bien souvent on se contentera par exemple des cas particuliers suivants :

– R : S x A x S → ℜ (récompense déterministe, c'est le choix que nous adopterons dans la suite) ;
– R : S x A → ℜ (récompense déterministe rattachée à l'action en ignorant son résultat) ;
– R : S → ℜ (récompense déterministe rattachée à un état donné).

2.2 Processus dynamique du MDP

Soit l'état initial d'un agent s0, on choisit une action a0 dans A à exécuter. Après la exécution, l'agent a transféré aléatoirement à l'état prochain s1 par rapport à la proba de T, s1 ⊆ T0a0. Et puis l'action a1, passer à s2, et puis a2... On peut représenter cette processus comme la figure ci-dessous : (see here).

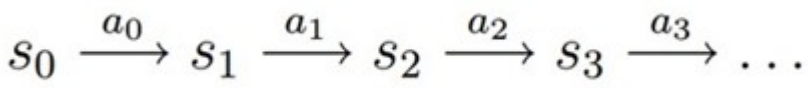

Figure 1: Réprésentation 1 du MDP.

Si la récompense a obtenu par rapport aux états s et aux actions a, le MDP peut être représenté comme la figure ci-dessous :

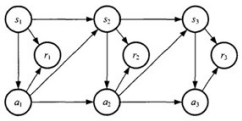

Figure 2: Réprésentation 2 du MDP.

`\citet` → Jenset and McGillivray (2017)
`\citet` → Shree, Liu, Gordon, and Hobbs (2019)
`\citep` → (Fabricius-Hansen & Haug, 2012)
`\citealp` → (Eckhoff et al., 2018; Fabricius-Hansen & Haug, 2012; Shree et al., 2019)

3 Objectifs du Projet

3.1 Maitriser les MDP

Avant de pouvoir manipuler les solveurs, il est crucial de comprendre les fondements théoriques des MDP. Tout d'abord il faut savoir reconnaître les situations dans lesquelles un problème peut être modélisé par un MDP. Des hypothèses telles que l'hypothèse de Markov doivent être vérifiées. Ensuite il va falloir savoir modéliser un problème sous forme de MDP, adapté au solveur qu'on souhaite utiliser.

3.2 Maîtriser les Solveurs

L'objectif principal de ce projet est de comparer 3 solveurs de MDP. Nous allons observer les performances de Gurobi(programme mathématique) MarmotMDP et ToolboxMDP. Avant de faire cela il va falloir bien s'approprier les 3 solveurs et savoir les manipuler. Trois grandes familles de méthodes existent pour résoudre de tels MDP en cherchant la politique optimale:

- Value Iteration: L'approche la plus classique qui se base aussi sur la résolution directe de l'équation d'optimalité de Bellman V. On fait des Itérations sur les états jusqu'à convergence.
- Policy Iteration: Iteration sur la politique; On propose une politique de départ et chaque itération fait une évaluation puis amélioration de la politique jusqu'à convergence.
- Programmation linéaire: Transformation de l'MDP en un problème linéaire et son Dualpour le résoudre.

Selon les critères, les algos ne sont pas les mêmes à utiliser. Les types de résultats(politiques) ne sont pas les mêmes aussi.

3.3 Modéliser un problème MDP

Pour pouvoir comparer les solveurs, nous allons prendre un exemple de problème modélisable comme un MDP et le résoudre. Il y a une étape intermédiaire importante et c'est la modélisation du problème et donc son adaptation ou traduction pour un solveur spécifique. En effet, chaque solveur requiert une modélisation particulière d'un problème pour pouvoir le résoudre.

3.4 Analyse des solveurs

C'est l'étape finale et clé du projet. Nous allons faire tourner les 3 solveurs sur un même problème et observer les résultats. Nous allons construire un benchmark (Un ensembled'exemples pour tester les caractéristiques) afin de comparer leurs caractéristiques comme la vitesse d'exécution, l'optimalité de la solution, la précision, la complexité d'utilisation.
Nous allons ensuite faire une synthèse des caractéristiques de chaque solveur et noterleurs avantages et inconvénients. Nous allons déterminer aussi pour chaque type de problème, quel solveur serait le plus adapté et performant.

3.4.1 Other simple functions

To add bullet points:

- Some point
- Another point

Or numbered points:

1. Some numbered point
2. Another numbered point

This is an example of footnote[1].

This is a simple table:

Table 1: A caption.

1	2	3	4
a	b	c	d
e	f	g	h

Please refer to your table using: Table 3.4.1.

To add a figure, upload the figure into the `images` folder, and then embed it:
To resize the figure:

Figure 3: JOHD's logo.

Please refer to your figures as: Figure 1, Figure 2, etc.

4 Method

Describe the methods used in the study.

5 Results and discussion

Describe and discuss the results of the study.

6 Implications/Applications

Provide information about the implications of this research and/or how it can be applied.

Acknowledgements

Please add any relevant acknowledgements to anyone else that assisted with the project in which the data was created but did not work directly on the data itself.

Funding Statement

If the research resulted from funded research please list the funder and grant number here.

Competing interests

If any of the authors have any competing interests then these must be declared. If there are no competing interests to declare then the following statement should be present: The author(s) has/have no competing interests to declare.

Notes

[1]This is a footnote

References

Eckhoff, H. M., Bech, K., Bouma, G., Eide, K., Haug, D., Haugen, O. E., & Jøhndal, M. (2018). The PROIEL treebank family: A standard for early attestations of Indo-European languages. *Language Resources and Evaluation*, *52*(1), 29–65. DOI: 10.1007/s10579-017-9388-5

Fabricius-Hansen, C., & Haug, D. T. T. (2012). Co-eventive adjuncts: Main issues and clarifications. In C. Fabricius-Hansen & D. T. T. Haug (Eds.), *Big events, small clauses* (pp. 21–54). Berlin-Boston: De Gruyter.

Jenset, G., & McGillivray, B. (2017). *Quantitative historical linguistics: A corpus framework*. Oxford: Oxford University Press.

Shree, J., Liu, E., Gordon, A., & Hobbs, J. (2019). Deep natural language understanding of news text. In *Proceedings of the first workshop on narrative understanding* (pp. 19–27). Minneapolis, Minnesota: Association for Computational Linguistics. DOI: 10.18653/v1/W19-2403

Supplementary Files (optional)

Any supplementary/additional files that should link to the main publication must be listed, with a corresponding number, title and option description. Ideally the supplementary files are also cited in the main text. Note: supplementary files will not be typeset so they must be provided in their final form. They will be assigned a DOI and linked to from the publication.