|  |
| --- |
| EPISEN, École publique d'ingénieurs de la santé et du numérique |
| Diplôme d’ingénieur systèmes d'information spécialité informatique décisionnelle |

Inférence causale et biais dans l’apprentissage automatique

Mémoire de fin d’études – Odelin Tamayo

|  |  |
| --- | --- |
| Eric Pipard | Tuteur pédagogique |
| Natixis SA | Entreprise |
| Viviane Le Naour | Maître d'apprentissage |
| 2019 - 2020 | |

Inférence causale et biais dans l’apprentissage automatique

Table des matières

[Introduction 4](#_Toc48560683)

[Réflexion 4](#_Toc48560684)

[Définitions 5](#_Toc48560685)

[Les différents cas de corrélation 7](#_Toc48560686)

[Plusieurs approches 8](#_Toc48560687)

[Effet cigogne intrinsèque aux algorithmes 8](#_Toc48560688)

[Biais induits par l’humain dans son utilisation de la technologie 8](#_Toc48560689)

[Comparaison entre types d’apprentissage automatique 8](#_Toc48560690)

[Implémenter une logique de randomisation 9](#_Toc48560691)

[Vrac 9](#_Toc48560692)

[I- Revue de littérature 11](#_Toc48560693)

[A- L’inférence causale en statistiques 11](#_Toc48560694)

[a) Approche graphique 11](#_Toc48560695)

[B- Une question triviale dans le cas de l’apprentissage automatique ? 12](#_Toc48560696)

[a) Réduction de biais dans l’apprentissage automatique 12](#_Toc48560697)

[b) Dans le cas de l’apprentissage par renforcement 13](#_Toc48560698)

[c) Exemples réels de logiciels biaisés 14](#_Toc48560699)

[C- Exemples de biais possibles dans l’apprentissage par renforcement 14](#_Toc48560700)

[a) Biais induits au sein des algorithmes 14](#_Toc48560701)

[b) Coïncidence 15](#_Toc48560702)

[c) Données d’apprentissage non représentatives 16](#_Toc48560703)

[d) Biais de confirmation 17](#_Toc48560704)

[Conclusion 17](#_Toc48560705)

[II- Description des expérimentations 18](#_Toc48560706)

[A- L’agent 19](#_Toc48560707)

[B- L’environnement 20](#_Toc48560708)

[C- Les biais à implémenter 20](#_Toc48560709)

[a) Trop paramétrer l’environnement 21](#_Toc48560710)

[b) Variable importante invisible 22](#_Toc48560711)

[c) Inertie face au changement de poids de variables 24](#_Toc48560712)

[d) Créer une corrélation sans causalité et observer l’inertie de l’agent 25](#_Toc48560713)

[e) Inciter au biais du rasoir d’Ockham 25](#_Toc48560714)

[III- Analyse des résultats 26](#_Toc48560715)

[A- Trop paramétrer l’environnement 26](#_Toc48560716)

[a) Avantage du sur-paramétrage en termes de vitesse d’apprentissage 26](#_Toc48560717)

[b) Inconvénients du sur-paramétrage en termes de résultat 27](#_Toc48560718)

[B- Variable importante invisible 28](#_Toc48560719)

[C- Inertie face au changement de poids de variables 29](#_Toc48560720)

[a) Résultats 29](#_Toc48560721)

[b) Première analyse 30](#_Toc48560722)

[c) Résolution et nouveaux tests 30](#_Toc48560723)

[d) Deuxième analyse 31](#_Toc48560724)

[e) Vérification de cette hypothèse 31](#_Toc48560725)

[f) Vérification de la seconde hypothèse 32](#_Toc48560726)

[g) Ralentir l’évolution et remesurer ? 33](#_Toc48560727)

[D- Inertie face à une corrélation temporaire 33](#_Toc48560728)

[E- Biais du rasoir d’Ockham 34](#_Toc48560729)

[Conclusion 35](#_Toc48560730)

[Sources 36](#_Toc48560731)

[VRAC 36](#_Toc48560732)

[Résumés de publications 36](#_Toc48560733)

[Autres sources 37](#_Toc48560734)

# Introduction

## Réflexion

Tout vient d’une réflexion sur la différence entre corrélation et causalité. Puisqu’une grande part de la cognition humaine est basée sur la notion de causalité, tandis que les algorithmes sont basées sur des corrélations, c’est une question récurrente sur les questions d’intelligence artificielle. Ce problème est parfois posé comme un des freins majeurs au développement de ces dernières (https://www.nature.com/articles/s41599-020-0494-4?source=techstories.org#Sec5).

Les productions humaines qui s’appuient sur l’apprentissage automatique peuvent également subir ce biais, ne serait-ce que parce que le biais vient des données en entrée, et ce, malgré une validation croisée.

Quelle réponse peut-on apporter à ce biais ?

Dans le monde réel, une méthode souvent appliquée est la randomisation : couper aléatoirement un échantillon en deux, agir sur seulement l’une des moitiés, et comparer les résultats entre les deux groupes.

Est-ce applicable à l’apprentissage automatique ?

Il faut pour cela que l’algorithme puisse agir sur son environnement pour apprendre non seulement à partir de données initiales, mais aussi à partir du résultat de ses actions. Il s’agit justement de la définition de l’apprentissage par renforcement, un des types d’apprentissage automatique.

L’apprentissage par renforcement, en agissant sur son environnement, est-il capable de différencier une corrélation d’une causalité ?

On peut lister différentes situations biaisées et observer dans quelle mesure un algorithme d’apprentissage par renforcement (un agent) tombe dans ces pièges, c’est-à-dire dans quelle mesure ces biais rendent difficile pour cet agent d’atteindre son objectif.

## Définitions

Effet cigogne : confusion entre corrélation et causalité

Corrélation : lien mesurable statistiquement entre deux variables.

Causalité : lien de cause à effet entre deux phénomènes.

Définitions issues principalement du Journal officiel n° 0285 du 09/12/2018 :

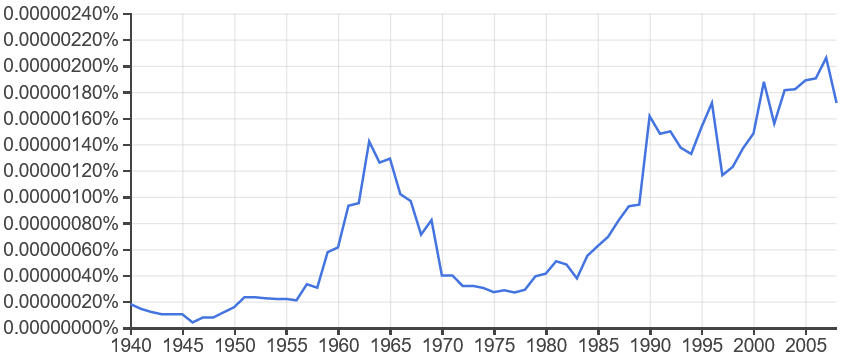
https://www.legifrance.gouv.fr/jo\_pdf.do?id=JORFTEXT000037783813

Apprentissage automatique : “Processus par lequel un algorithme évalue et améliore ses performances sans l’intervention d’un programmeur, en répétant son exécution sur des jeux de données jusqu’à obtenir, de manière régulière, des résultats pertinents”.

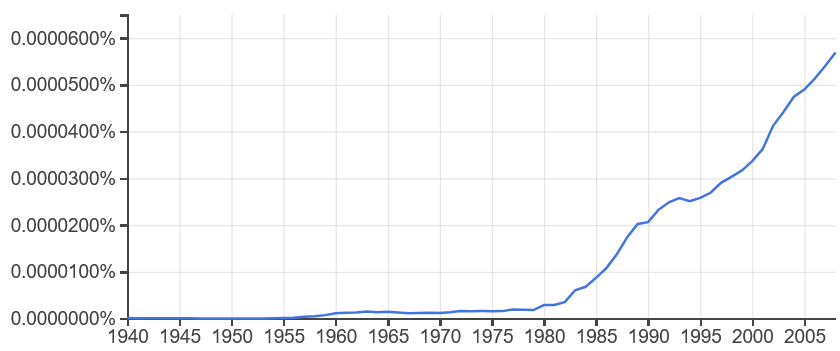
Le journal officiel le traduit par “machine learning” en anglais. Cela n’est pas certain quand on compare les fréquences de ces expressions au cours du temps avec Ngrams viewer.

Soit l’expression “apprentissage automatique” était porteuse d’un tout autre sens dans les années 1960, soit il s’agit de deux notions différentes avec le même nom.

Apprentissage automatique dans le corpus francophone :



Fréquences de “apprentissage automatique” en français

Machine learning dans le corpus anglophone : 

Apprentissage par renforcement : “Apprentissage automatique dans lequel un programme extérieur évalue positivement ou négativement les résultats successifs de l’algorithme, l’accumulation des résultats permettant à l’algorithme d’améliorer ses performances jusqu’à ce qu’il atteigne un objectif préalablement fixé.”

Agent : algorithme d’apprentissage par renforcement.

Rasoir d’Ockham : aussi appelé principe de parcimonie, est un principe de raisonnement consistant à préférer le modèle auto-suffisant le plus simple pour expliquer un phénomène. On le trouve souvent sous cet énoncé : “les hypothèses suffisantes les plus simples doivent être préférées”.

Biais du rasoir d’Ockham : privilégier les modèles les plus simples peut éliminer une explication qui est tout aussi vraie.

Biais (définition générale) : une démarche ou un procédé qui engendre des erreurs.

Biais (en statistiques) : différence entre la valeur de l’espérance d’un estimateur et la valeur qu’il est censé estimer.

Dilemme biais-variance : le biais diminue à mesure que la complexité du modèle s’approche de la complexité (ni plus simple ni plus complexe) que l’environnement.

Inférence causale : L’inférence causale désigne le processus par lequel on peut établir une relation de causalité entre un élément et ses effets.

https://fr.wikipedia.org/wiki/Inf%C3%A9rence\_causale

### Les différents cas de corrélation

#### a) Théoriques

Sachant que A est corrélé à B, il y a plusieurs explications possibles :

* A cause B
* B cause A
* A cause B et B cause A
* C connu cause A et B
* C inconnu cause A et B
* Coïncidence.

Si on ne peut observer C, peut-on différencier le cas 5 du cas 6 ? Peut-on estimer que nos données semblent répondre à une variable supplémentaire inconnue à partir d’une certaine quantité de données permettant d’écarter l’idée d’une coïncidence ? Si oui, peut-on mesurer cette variable ? (sachant qu’on risque de la confondre avec le vrai bruit statistique)

#### b) Illustration de ces cas

Dans le cas où A, la quantité vendue, est corrélé à une variable B

* La quantité vendue cause le résultat net
* Le prix de vente cause la quantité vendue
* Le nombre de ventes cause la fréquentation à venir, qui cause le nombre de ventes à venir
* Les jours d’affluence causent la quantité vendue d’un produit et celle d’un autre produit
* La complémentarité entre la farine et la levure cause une corrélation entre les quantités vendues de l’un et de l’autre
* Une variable n’ayant aucun lien de causalité avec quoi que ce soit (exemple : l’horoscope des sagittaires) peut néanmoins se retrouver corrélé avec d’autres variables si on ne dispose pas d’un échantillon suffisamment grand.

## Plusieurs approches

### Effet cigogne intrinsèque aux algorithmes

La définition formelle en statistique du terme biais, “différence entre la valeur de l’espérance d’un estimateur et la valeur qu’il est censé estimer” correspondrait à un problème de régression.

Pour un algorithme d’apprentissage par renforcement, l’objectif n’est pas tant d’estimer au plus proche une variable que d’en maximiser une. Cela dit, comme les algorithmes d’apprentissage automatique utilisent des modèles statistiques, ils sont aussi sensibles à ces biais, et soumis au dilemme biais-variance.

* Peut-on utiliser l’apprentissage par renforcement pour différencier des corrélations de causalités ? Pour prouver l’existence de causalités ?
* Dans quelle mesure ce biais influence-t-il le résultat de prédictions basées sur l’apprentissage automatique ?
* En cas de correction d’un modèle biaisé, avec quelle inertie les algorithmes apprennent de leurs erreurs ?

### Biais induits par l’humain dans son utilisation de la technologie

Comme nous l’avons vu, la corrélation est une considération statistique, tandis que la causalité appartient au réel. Un algorithme d’apprentissage automatique n’a pas nécessairement besoin de la notion de causalité pour répondre à sa fonction.

C’est après des décisions humaines que la solution finira par utiliser des corrélations comme causalité.

### Comparaison entre types d’apprentissage automatique

Une approche pour isoler dans quelle mesure l’apprentissage par renforcement distingue une corrélation d’une causalité serait de soumettre ses données d’apprentissage à un algorithme d’apprentissage automatique différent.

Pour cela, on pourrait imaginer enregistrer toutes les expériences réalisées par un agent pour apprendre, et utiliser ensuite ces dernières pour un apprentissage supervisé.

Cependant, les résultats d’une régression dépendent énormément de différents paramètres liés à l’environnement. Il est possible, lors du paramétrage de l’environnement, d’ajouter de la connaissance du contexte que l’agent n’aurait pas eu seul, voire d’ajouter des biais de cette façon.

### Implémenter une logique de randomisation

Si un algorithme d’apprentissage par renforcement est soumis à des biais, peut-on appliquer une solution analogue à celle que les humains font sur le réel ?

Peut-on développer un tel algorithme, ou forcer un algorithme existant à le faire ?

Quel impact cela aura-t-il sur ses résultats ?

L’ajout d’une randomisation devrait ralentir l’agent dans son analyse et ralentir sa progression dans les situations non biaisées.

L’agent devrait également être ralenti dans une situation biaisée si les algorithmes actuels sont suffisants pour appréhender la différence entre corrélation et causalité pour les mêmes raisons.

Cette expérimentation pourrait même être affinée en la combinant à une situation de causalité impliquant des paramètres que l’agent ne peut pas tous observer.

### Vrac

* On peut imaginer ajouter des corrélations sans causalité dans l’environnement, et regarder dans quelle mesure elles perturberont l’apprentissage de l’agent **–> Commun à toutes les approches**
* Randomisation, n’est-elle pas en quelque sorte déjà faite par l’algo lors de l’exploration ?
* Randomisation : passer de temps en temps en random policy ou en collect policy pour réduire l’inertie ? **–> trop proche de l’implémentation**
* Versus l’efficacité si on pré-entraine l’agent avec des données ayant déjà été observées puis qu’on le laisse se renforcer avec randomisation (apprentissage supervisé et par renforcement) **–> trop proche de l’implémentation**

# Revue de littérature

## L’inférence causale en statistiques

Des approches existent et parviennent à reconnaître statiquement les corrélations les plus susceptibles d’être des causalités à partir de données “à plat” :

https://www.mitpressjournals.org/doi/full/10.1162/NECO\_a\_00820

D’après cette méta-analyse, il y a beaucoup d’approches dans beaucoup de champs de recherche, mais pas pour l’heure d’outil commun répandu entre les disciplines.

https://www.nature.com/articles/s41576-018-0020-3/

Résultats prometteurs en 2020

https://projecteuclid.org/euclid.ba/1580461461

A DEVELOPPER

### Approche graphique

Les relations de causalité entre des phénomènes peut être modélisée par un graphe orienté acyclique entre les variables qui mesurent ces phénomènes.

https://www.nature.com/articles/s41390-018-0071-3

#### Exemple d’utilisation de l’apprentissage par renforcement pour créer le graphe de causalité

<https://arxiv.org/abs/1906.04477>

## Une question triviale dans le cas de l’apprentissage automatique ?

La question de confusion entre corrélation et causalité dans le cas de l’apprentissage automatique peut être considérée comme triviale du point de vue des algorithmes. En effet, ceux-ci ne sont pas basés sur les causalités, qui sont des phénomènes réels, seulement des corrélations, qui sont des liens statistiques. Selon cette considération, c’est au moment où ces algorithmes sont utilisés au sein d’une application que l’intelligence humaine (des développeurs, des analystes) va biaiser la situation.

Cependant, il convient d’analyser ceci de façon spécifique face à différentes approches de l’apprentissage automatique.

### Réduction de biais dans l’apprentissage automatique

Une grande partie des biais peuvent être réduits par une validation croisée :

* Couper l’échantillon en deux de façon aléatoire
* Apprendre à partir d’une partie des données
* Vérifier la qualité de l’apprentissage en regardant dans quelle mesure le modèle se trompe sur les données restantes.

Une autre part des biais dépend directement de la qualité des données en entrée. Si les données sont elles-mêmes non représentatives de la situation, une validation croisée ne résoudra pas le problème.

Le cas de la causalité

Dans le cas où confusion entre corrélation et causalité il y a, ça n’est pas au niveau des algorithmes d’apprentissage automatique, qui n’étudient que les corrélations. C’est lors de l’utilisation de ces algorithmes qu’il peut y avoir confusion.

Puisque les modèles consistent à mesurer les corrélations entre toutes les variables afin d’en estimer une à partir des autres, il suffit que certaines variables soient corrélées avec celle à estimer pour biaiser les résultats. Pour combattre ce biais, il faut qu’un être humain analyse le contexte pour déterminer s’il y a causalité entre les variables.

### Dans le cas de l’apprentissage par renforcement

Si le cas de la régression semblait trivial, c’est peut-être parce que l’algorithme n’a pas l’occasion d’interagir avec son environnement pour tester ce qui est une corrélation et ce qui est une causalité.

Cependant, peut-on utiliser des algorithmes d’apprentissage par renforcement pour trouver/prouver des relations de causalité entre des variables ?

Pour les variables de ses actions, on peut se dire que oui : il suffit de le laisser agir pour qu’il fasse ou non varier les autres observations et la récompense.

<https://arxiv.org/abs/1905.10958>

**UTILISER ET CITER**

Cependant, si la récompense augmente et que les observations évoluent de façon linéaire au fil du temps, cela ne prouve pas forcément que les actions font s’améliorer la récompense. Il est possible que la seule relation de causalité soit non pas entre les actions d’un côté, la récompense et les observations de l’autre, mais entre le temps et ces dernières.

On peut d’ailleurs différencier trois types de variables qui construisent la réalité avec laquelle interagit un agent :

* Les données intrinsèques à l’environnement (une partie des observations)
* Les entrées (les actions de l’agent sur l’environnement)
* Les sorties issues des actions sur l’environnement (la récompense et une partie des observations).

Au sein des librairies que nous utiliserons, ces variables sont découpées de la sorte :

* Les observations (qu’elles dépendent des actions de l’agent ou non)
* Les actions (les décisions prises par l’agent)
* La récompense (ce que l’agent doit maximiser).

Ainsi, contrairement au cas d’une régression, il est déterminé dès le départ sur quoi l’agent peut agir. L’agent ne peut pas vérifier l’existence de causalité entre deux variables qui ne dépendent pas de lui.

On pourrait se dire que l’algorithme explorera l’environnement à sa disposition pour maximiser sa récompense indépendamment de toute notion de toute notion de causalité. Cependant, il reste possible d’expérimenter si et dans quelle mesure l’algorithme tombe dans des biais.

### Exemples réels de logiciels biaisés

Dans le cas de l’application de sélection de CV d’Amazon qui défavorisait les profils féminins, l’erreur vient du fait qu’on ait utilisé des données biaisées (des recrutements ayant eu lieu durant 10 années).

Derrière l’expression “l’application défavorise les profils féminins”, on pourrait comprendre “l’algorithme suppose qu’il y a une causalité entre le genre d’un individu et ses compétences”. Or, il s’agit davantage de “l’algorithme observe une corrélation entre le genre des individus et leur proportion au sein des recrutements à laquelle il lui a été demandé de correspondre”.

Le problème n’est donc pas intrinsèque à l’algorithme mais dû aux données en entrée, et à la décision de partir de ces données pour obtenir ce résultat. Il s’agit d’un biais présent dans les données d’apprentissage, ainsi que d’une erreur humaine de confusion entre corrélation et causalité, les concepteurs ayant supposé que les choix de recrutement passés étaient des décisions sinon optimales, au moins de bons exemples vers lesquels tendre, alors que ces choix étaient marqués d’erreurs humaines.

## 

## Exemples de biais possibles dans l’apprentissage par renforcement

### Biais induits au sein des algorithmes

https://arxiv.org/pdf/1907.02908.pdf

Quand un agent est développé pour répondre à un besoin spécifique, on peut être tenté de le paramétrer via des connaissances préexistantes afin d’améliorer ses résultats. Cela peut causer des erreurs supplémentaires, en plus de rendre l’algorithme moins généralisable.

#### Exemple de ce problème dans notre expérimentation

Si notre agent doit acheter des marchandises puis les vendre, avec comme récompense la marge sur coûts variables.

Pour lui éviter d’essayer des cas triviaux et a priori contreproductifs, on serait tenté de le paramétrer de telle sorte qu’il ne fixe jamais de prix de vente inférieur au prix d’achat. Ce qui serait une bonne idée sans compter que :

* Un produit d’appel peut être vendu à perte et pourtant améliorer le revenu global
* Si l’agent doit gérer ses stocks, il peut arriver qu’il doive vendre à perte pour déstocker (ex: péremption)
* Vendre ou acheter à perte peut parfois être une obligation légale
* Exemple : EDF qui achète à un prix plancher l’électricité issue d’énergies renouvelables sur le marché à terme de l’électricité.

### Coïncidence

Cas trivial mais néanmoins possible. On serait tenté de se dire qu’il s’agit là d’une question habituelle de quantité de données insuffisante.

Cependant, le fait est que l’étude des petits jeux de données est un champ de recherche à part entière aussi bien en statistiques que dans l’apprentissage automatique. Il apparait que tous les algorithmes n’apprennent pas aussi bien selon les volumétries.

Dans quelle mesure les informaticiens sont-ils formés aux conditions de volumétrie des différents algorithmes et de leurs différentes implémentations ? Ces conditions sont-elles suffisamment documentées ?

### Données d’apprentissage non représentatives

#### Peut-on tromper l’agent s’il ne peut déterminer l’importance de ses actions dans la récompense finale ?

Si 80% de sa récompense est basée sur 20% de ses actions, l’algorithme mettra plus de temps à estimer l’importance respective de chaque variable.

On peut maximiser ce biais :

* En fournissant une récompense et/ou des observations agrégées à une granularité trop épaisse
* Nous arrivons alors à une coïncidence et un biais de surapprentissage.

#### Données non représentatives : l’avantage de l’apprentissage par renforcement sur l’apprentissage supervisé

https://app.wandb.ai/stacey/aprl/reports/Adversarial-Policies-in-Multi-Agent-Settings–VmlldzoxMDEyNzE

Résumé du protocole de cette publication :

* On prend deux agents, A et B, et un jeu compétitif.
* A apprend à jouer à partir de données de véritables joueurs.
* Puis B apprend à jouer contre A.

Il en résulte que la meilleure manière pour B de gagner consiste à ne pas jouer. En effet, A n’a appris à jouer que contre des personnes qui savent jouer. B faisant des choses inattendues, A perd tout seul. L’apprentissage par renforcement gagne sur le long terme face à un programme exclusivement formé sur des données qui ne recouvrent pas assez de cas.

Conclusion : l’apprentissage par renforcement a moins de difficultés face à des données partielles en entrée.

### Biais de confirmation

Un principe de raisonnement parfois considéré comme fondamental de la philosophie des sciences peut aussi être la source de biais.

Blumer, Ehrenfeucht, Haussler & Warmuth, 1987 selon https://arxiv.org/pdf/cmp-lg/9612001.pdf mais je ne trouve pas la publication originale.

https://psycnet.apa.org/fulltext/2017-54956-007.html

Les algorithmes y sont-ils sensibles ? Causalité au début qui décroit avec le temps, mais l’algorithme continue dans le sens initial ?

Testable, mais consiste surtout à mesurer l’inertie de l’algorithme à un changement du poids de ses variables.

## Conclusion

En conclusion, si de nombreuses approches existent sur la question de l’inférence causale, mais aucune ne fait consensus dans le domaine de l’apprentissage automatique. Le cas particulier de l’apprentissage par renforcement semble intéressant, expérimentable en plus d’être relativement peu étudié. C’est pourquoi nous expérimenterons ce dernier.

<https://arxiv.org/abs/1905.10958>

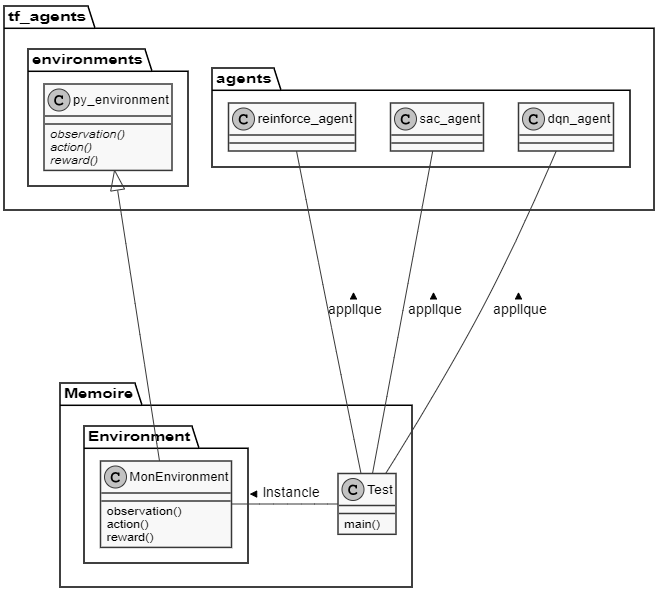
# Description des expérimentations

Les expérimentations ont consisté à faire interagir un agent suivant plusieurs algorithmes d’apprentissage par renforcement pour apprendre face à un environnement biaisé.

Pour créer des situations biaisées, on préfèrera utiliser une librairie permettant de créer un environnement.

J’ai choisi Tensorflow, car la documentation semble claire et bien fournie, et que la librairie implémente de nombreux algorithmes de différentes catégories.

Diagramme de classes simplifié :



Le code est quant à lui sur le repository suivant : https://github.com/OdelinT/Memoire

## L’agent

Les algorithmes suivants sont présents dans TF :

* Behavioral\_cloning : https://arxiv.org/abs/1805.01954
* DDPG : Deep Deterministic Policy Gradient https://arxiv.org/pdf/1509.02971.pdf
* DQN : Deep Q Network https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf
* Categorical\_DQN : un type d’agent DQN
* PPO : Proximal Policy Optimization https://arxiv.org/abs/1707.06347
* Reinforce : https://people.cs.umass.edu/~barto/courses/cs687/williams92simple.pdf
* SAC : Soft Actor Critic https://arxiv.org/abs/1801.01290
* TD3 : Twin Delayed Deep Deterministic policy gradient https://arxiv.org/pdf/1802.09477.pdf

Lors de la mise en place du projet de test, il s’est avéré que l’intégralité des implémentations de ces algorithmes dans la librairie Tensorflow étaient contraignantes, nécessitant des conditions spécifiques non documentées. Par exemple, l’agent DQN ne s’applique qu’à des environnements dont les actions sont des dimensions spécifiques.

La majorité de ces problèmes se sont révélés dans des messages d’erreur lors de l’application des agents à l’environnement. Pour éviter de perdre du temps à chercher à différencier ce qui est une erreur de ma part de ce qui est une contrainte non documentée, je me suis concentré dans les expérimentations sur l’agent SAC, qui est le premier à avoir donné des résultats satisfaisants. Finalement, ce dernier a lui aussi été la source de quelques difficultés, comme vous le verrez plus tard.

**EXPLIQUER LES POLICES**

## L’environnement

On peut imaginer des prix mis à jour en temps réel par l’agent, l’objectif de l’agent étant de trouver le prix maximisant le résultat net. L’environnement répondrait, pour chaque offre, une demande (un nombre d’achats).

Cas réels qui correspondraient : prix dans un centre commercial connecté, sur un site d’e-commerce, sur un marché à terme en temps réel (financier, de l’électricité, du blé), etc.

Dans TF, on peut créer plusieurs types d’environnement. Tous prennent en compte des paramètres similaires. Dans notre exemple, l’environnement de base à partir duquel seront construits les autres aura les paramètres suivants :

* Le temps est linéaire et discret
* Action : pour chaque lieu et/ou produit, un prix de vente
* Observation : pour chaque lieu et/ou produit, une demande
* Récompense : la somme, pour chaque lieu et/ou produit, du prix de vente auquel on soustrait le prix d’achat.

## Les biais à implémenter

La classe de l’environnement basique sera dupliquée en plusieurs versions, chacune ayant pour but de tester un biais ou une situation spécifique.

La graine des paramètres aléatoires sera la même dans tous les environnements, et on créera un test unitaire pour vérifier que les paramètres générés dans les différents environnements seront bien identiques. Cela permettra d’écarter la possibilité que certains environnements soient, au moment de l’exécution des tests, plus favorables que les autres.

On pourrait utiliser un système d’héritage, mais les classes sont assez courtes et les paramètres à faire évoluer ne sont présents que dans deux méthodes à redéfinir. Il est donc bien plus simple et lisible de dupliquer les classes sans lien d’héritage entre elles.

### Trop paramétrer l’environnement

Comme mentionné plus tôt, on peut être tenté de contraindre notre agent dans ses actions et lui éviter d’essayer des actions qui nous paraissent contreproductives.

https://arxiv.org/pdf/1907.02908.pdf

Dans notre cas, et afin d’obtenir des résultats plus rapidement, on peut en effet afin d’obtenir des résultats plus rapidement interdire à notre environnement de vendre à un prix inférieur à son coût unitaire.

|  |
| --- |
| self.\_action\_spec = array\_spec.BoundedArraySpec(  shape=(1,), dtype=np.float32, minimum=1, maximum=100, name='action')  self.\_action\_spec = array\_spec.BoundedArraySpec(  shape=(1,), dtype=np.float32, minimum=1, maximum=100, name='action') |

Ici, l’action correspond au prix auquel on vend un produit, exprimé en un coefficient multiplicateur du coût unitaire de ce produit.

Avec certains algorithmes, on obtient les premiers résultats positifs dès les premières itérations si le prix minimal est le coût unitaire, au bout de plusieurs milliers d’itérations si le prix minimal est de 0.

Cependant, ce genre d’approche peut empêcher l’agent d’être optimal dans certains cas :

* Péremption d’un produit
* Un produit d’appel peut être vendu à perte afin de permettre de vendre plus au final (essence à la station-service d’un hypermarché, par exemple)
* Spéculation (il faut parfois accepter de vendre avec un déficit pour pouvoir réinvestir sur un produit qui offre de meilleures perspectives)

Pour le cas d’un produit d’appel, nous ne verrons pas ici de mesure du manque à gagner possible, car celui-ci ne peut dépendre que de cas réels très spécifiques.

Le cas de la péremption comprend beaucoup de paramètres, et en établir une simulation réaliste risque d’être trop complexe. On fera donc ici une approximation de la possible différence de résulat entre un agent pouvant vendre à perte et un autre qui ne le peut pas.

### Variable importante invisible

Un paramètre inconnu est créé, et influence les résultats. Ensuite, on modifie ce paramètre et on observe l’inertie de l’agent en comparant ses résultats à ceux qu’il aurait obtenu sur un environnement qui aurait été dès le départ à l’étape finale.

Exemples de variables invisibles :

* La taille des magasins. Si on imagine que l’expérience était sur les magasins carrefour city et qu’elle inclut par la suite également les carrefour market, d’une taille en moyenne différente. Toutes les quantités varient.
* La flexibilité de la demande selon le prix. On peut imaginer qu’au fil du temps ce paramètre influe plus ou moins les décisions d’achat.
* Dans l’environnement de base, on a pour seule observation la quantité vendue. Le fait que les actions soient le prix de vente de chaque produit exprimé en un coefficient multiplicateur du coût, et la récompense la marge sur coûts variables, rendent extrêmement difficile de déterminer l’importance relative de chaque produit dans le résultat final.

On testera donc en ajoutant tous ces paramètres dans les observations fournies par notre environnement à notre agent.

Par la suite, on fera évoluer ces paramètres, et on mesurera l’inertie des agents.

#### Difficulté d’implémentation

Un bug dans la librairie Tensorflow perturbe cette expérience.

Lors de la création de l’environnement, on doit spécifier les dimensions des observations (retournées par l’environnement) ainsi que des actions (à effectuer sur l’environnement) de la manière suivante :

|  |
| --- |
| self.\_action\_spec = array\_spec.BoundedArraySpec(  shape=(10,), dtype=np.float32, minimum=1, maximum=100, name='action') self.\_observation\_spec = array\_spec.ArraySpec(  shape = (6, 10),dtype=np.float32,name = 'observation') |

La spécification signale que les actions doivent être de dimension 10 (un prix pour chaque produit) et que les observations retournées seront de dimension 6\*10.

En effet, pour cette expérience, les observations ne seront plus seulement la quantité vendue de chaque produit, mais également 5 autres tableaux de dimension 10 représentant respectivement :

* Le coût des produits
* Leur taux de marge usuel
* Leur taux d’achat habituel
* Leur prix
* La flexibilité de la demande liée au prix

L’environnement est valide (conforme à ses spécifications, et qu’il fonctionne si on le manipule) selon la méthode prévue à cet effet par la librairie Tensorflow :

|  |
| --- |
| utils.validate\_py\_environment(self.BetterObservations\_env, episodes=5) |

Mais les actions suggérées par l’algorithme SAC sont sont conformes aux spécifications des observations et non des actions. Ce problème n’apparaît que maintenant étant donné que jusqu’à présent ces spécifications étaient les mêmes.

Solutions possibles :

* Changer de librairie
  + Trop long
* Changer d’algorithme
  + Très long, SAC était le seul à donner des résultats satisfaisants rapidement
* Mettre à jour la librairie concernée
  + Passer de Tensorflow 2.2.0 à 2.3.0 (la seule mise à jour possible) apporte trop de changements pour que le reste du code fonctionne, sans garantie que le problème soit résolu.
  + Le problème risque d’être le même en utilisant la version nightly (build quotidien).
* Redimensionner les actions reçues
  + **C’est cette solution que nous retiendrons pour l’instant (15/08)**

Les 6 lignes des actions suggérées par l’agent seront additionnées pour n’en former qu’une seule. Pour ces raisons, l’expérience pourra fournir des résultats, mais il sera très difficile de pouvoir extrapoler ces derniers dans la mesure où les actions de l’agent ne seront pas appliquées telles qu’elles sont censées.

|  |
| --- |
| if action.shape != self.productsCosts.shape:  action = np.sum(action, axis=0) |

### Inertie face au changement de poids de variables

Est-ce que le modèle se complexifie pour prendre en compte les anciennes données en plus des récentes ?

Quels paramètres pour modifier le taux d’apprentissage et le poids des variables au fil du temps permettent de limiter ce problème ?

Les différents paramètres évolueront dans des sens différents (favorables ou défavorables au résultat), afin d’une part de ne pas trop complexifier les comparaisons, et d’autre part pour qu’exploiter les environnements évolués nécessite des “tactiques” différentes de l’environnement initial.

Nuance avec l’expérience précédente : contrairement à l’expérience précédente, qui consistait à comparer des résultats issus d’observations complètes de résultats issus d’observations partielles, il s’agit ici de comparer les résultats avec ou sans changement des paramètres qui ne font pas partie des observations.

### Créer une corrélation sans causalité et observer l’inertie de l’agent

Pour cela, on peut forcer arbitrairement les actions de l’agent dans un premier temps à aller dans une certaine direction.

L’environnement sera paramétré pour retourner de meilleurs résultats au début des tests. Cela créera une corrélation sans causalité entre la direction des variables et le résultat.

En mesurant l’inertie de l’agent pour rétablir ses variables vers quelque chose qui cause effectivement le résultat, on devrait déterminer dans quelle mesure celui-ci confond corrélation et causalité.

### Inciter au biais du rasoir d’Ockham

On devrait pouvoir inciter un agent à mettre en évidence un biais du rasoir d’Ockham https://arxiv.org/pdf/cmp-lg/9612001.pdf

https://ieeexplore.ieee.org/abstract/document/4640797/

Pour créer la situation :

* Créer deux variables corrélées, l’une expliquant beaucoup les observations, l’autre moins, pour que l’agent se concentre sur la première
* Inverser l’importance de ces variables au fil du temps

Il devrait ensuite suffire de comparer les résultats avec des tests directement sur la valeur finale de la variable, tout comme dans l’expérience C.

**PLUS REDIGER**

# Analyse des résultats

## Trop paramétrer l’environnement

### Avantage du sur-paramétrage en termes de vitesse d’apprentissage

Voilà un tableau des résultats obtenus après au fur et à mesure d’un apprentissage sur 10 000 étapes.

|  |  |  |
| --- | --- | --- |
| Etape | Résultats en cas de vente à perte autorisée | Résultats en cas de vente à perte interdite |
| 0 | 0 | 0 |
| 2000 | 2 995 | 4 994 |
| 4000 | 5 621 | 11 849 |
| 6000 | 4 766 | 15 283 |
| 8000 | 4 112 | 18 320 |
| 10000 | 8 165 | 20 326 |

Bien que les résultats puissent changer aléatoirement lors de l’exécution de l’algorithme SAC, on observe en général plus rapidement de bien meilleurs résultats sur un environnement où l’agent ne vendra pas à perte.

En mesurant l’efficacité des agents lors de 100 tests à la fin de 1 000 étapes d’apprentissage, on obtient le tableau suivant :

|  |  |  |
| --- | --- | --- |
|  | Résultats en cas de vente à perte autorisée | Résultats en cas de vente à perte interdite |
|  | 7 400 | 1 380 |
|  | 2 519 | 8 485 |
|  | 8 901 | 16 389 |
|  | 2 175 | 2 676 |
|  | 5 767 | 3 754 |
|  | 8 797 | 13 232 |
|  | 8 815 | 4 335 |
|  | 9 207 | 14 685 |
|  | 3 811 | 10 903 |
|  | 11 554 | 8 848 |

Pour rappel, les environnements ont des paramètres identiques. En effet, la même graine est utilisée pour la génération des nombres aléatoires, et nous disposons d’un test unitaire qui vérifie que ce soit bien le cas. Ainsi, **les données peuvent être interverties en colonne**, faire la moyenne en ligne n’a donc pas de sens.

On peut observer que la moyenne des 10 résultats est plus élevée de 23% lorsque l’environnement ne peut pas vendre à perte (8 469) que lorsqu’il le peut (6 895).

### Inconvénients du sur-paramétrage en termes de résultat

Si, dans la partie précédente, nous avons vu que dans notre environnement fictif, il est néfaste de pouvoir vendre à perte, cela peut être utile voire nécessaire dans le monde réel.

Si on suppose qu’un produit périmé à un prix supérieur ou égal à son coût ne se vendra pas (hypothèse coûteuse en soi) :

Il existe des magasins spécialisés sur les produits périmés. Dans cet exemple, ceux-ci sont vendus avec un taux de réduction de 30% par rapport à un prix en magasin :

https://www.francetvinfo.fr/sante/alimentation/video-le-supermarche-anti-gaspi-qui-vend-des-produits-perimes\_3595515.html

Le taux de marge de la distribution alimentaire, très soumise aux questions de péremption, est selon l’INSEE compris entre 13% et 27% (on retiendra 20%).

https://www.insee.fr/fr/statistiques/:~:text=Pour%20les%20produits%20alimentaires,%20les,de%20produits%20%C3%A0%20l’autre.

Les pertes représentent 3,3% du poids des denrées alimentaires transitant par la distribution (on supposera le même ordre de grandeur en valeur)

https://www.ademe.fr/sites/default/files/assets/documents/pertes-gaspillages-alimentaires-etat-lieux-201605-synt.pdf page 8

Empêcher un agent de vendre à perte des denrées alimentaires peut occasionner un manque à gagner de l’ordre de 3,3% des denrées vendues à 70% de 120% de leur coût.

Soit un manque à gagner de 3,3% \* 70% \* 120% ~= 2,8% de ses coûts, soit (2,8 / 120%) / 2,9 ~= 80% de la marge opérationnelle courante d’une entreprise de grande distribution telle que Carrefour.

https://bfmbusiness.bfmtv.com/entreprise/carrefour-a-renoue-avec-les-benefices-en-2019-apres-deux-annees-dans-le-rouge-1865256.html#:~:text=Quant%20%C3%A0%20la%20marge%20op%C3%A9rationnelle,r%C3%A9sultat%20net%20part%20du%20groupe).

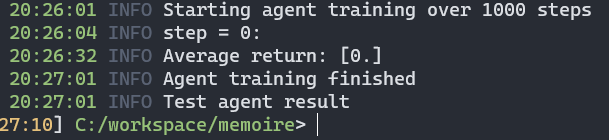
En conclusion, dans une situation où les produits vendus peuvent périmer, et où ceux-ci sont à faible valeur ajoutée, il est théoriquement possible qu’un agent chargé de gérer les prix avec interdiction de vendre à perte ait de moins bons résultats qu’un autre. Cette différence pourrait être d’un ordre de grandeur comparable à celui de la marge opérationnelle de celui qui a le droit de vendre à perte.

## Variable importante invisible

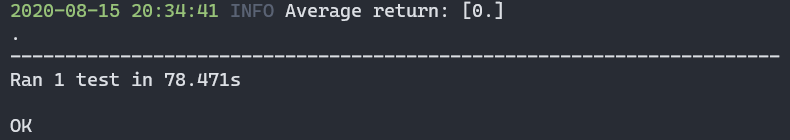
Les difficultés d’implémentations ne se sont finalement pas arrêtées là.

En effet, l’exécution des expériences s’interrompt purement et simplement sans levée d’exception ni message d’erreur, et stoppe même le module de tests unitaires censé gérer ces comportements.

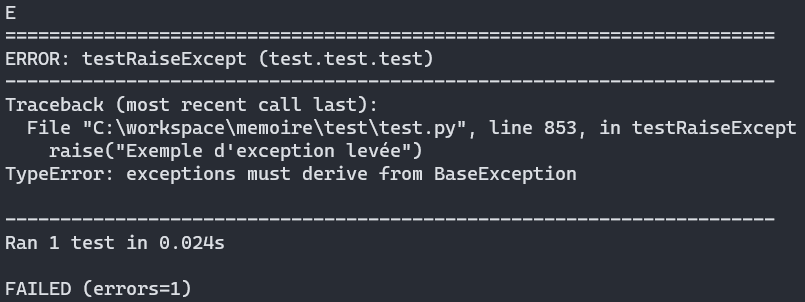
Ce qui se produit :



Comment est censé se terminer un test réussi :



Comment est censé se terminer un test échoué :



**En conclusion, ce cas ne sera pas traité ici 15/08**

## Inertie face au changement de poids de variables

### Résultats

Résultat sur 10 tests d’apprentissage sur 1 000 étapes :

|  |  |  |
| --- | --- | --- |
| Environnement statique | Environnement qui évolue au fil du temps | Environnement qui suit une évolution comparable à l’initialisation puis qui devient statique |
| 2 775 | 536 | 0 |
| 2 501 | 0 | 0 |
| 7 881 | 0 | 0 |
| 2 679 | 0 | 0 |
| 4 323 | 0 | 0 |
| 4 597 | 0 | 0 |
| 2 988 | 0 | 0 |
| 3 144 | 0 | 0 |
| 4 376 | 0 | 0 |
| 8 633 | 0 | 0 |

### 

### Première analyse

C’est réussi, la colonne du milieu a des données au milieu comprises entre les deux autres !

Autre explication : les paramètres devaient ne pas être viables.

### 

### Résolution et nouveaux tests

Après modification de l’évolution des paramètres pour qu’ils restent plus viables :

|  |  |  |
| --- | --- | --- |
| Environnement statique | Environnement qui évolue au fil du temps | Environnement qui suit une évolution comparable à l’initialisation puis qui devient statique |
| 5 157 | 0 | 1 747 |
| 11 084 | 0 | 2 776 |
| 8 247 | 0 | 2 426 |
| 4 975 | 0 | 1 705 |
| 8 943 | 0 | 9 835 |
| 4 987 | 0 | 5 759 |
| 11 351 | 0 | 2 186 |
| 3 705 | 0 | 4 634 |
| 3 325 | 0 | 2 715 |
| 4 463 | 0 | 14 475 |

Moyenne de 6 624 dans le premier cas, 0 dans le deuxième, 4 826 dans le troisième.

### Deuxième analyse

Les résultats pourraient être interprétés tels quels : faire évoluer les paramètres, même progressivement, donne de moins bons résultats que de garder des paramètres constants.

**Cependant**, il est très surprenant de n’obtenir **aucun** résultat sur l’environnement qui évolue au cours du temps après 1 000 épisodes.

Hypothèse : si les variables de l’environnement n’étaient pas réinitialisées entre chaque épisode, et que les évolutions tendent à rendre l’environnement moins viable (ce qui est probable au regard des résultats obtenus en moyenne 37% plus faibles), il n’est pas surprenant qu’au bout de 1000 épisodes \* 30 étapes = 30 000 modifications, il devienne extrêmement difficile d’obtenir un gain.

### Vérification de cette hypothèse

En effet, la méthode de réinitialisation d’environnement ne régénérait pas les variables modifiées à chaque étape dans le cas de l’environnement que nous testions ici.

|  |  |  |
| --- | --- | --- |
| Environnement statique | Environnement qui évolue au fil du temps | Environnement qui suit une évolution comparable à l’initialisation puis qui devient statique |
| 13 808 | 0 | 112 |
| 7 296 | 0 | 12 079 |
| 5 353 | 0 | 3 194 |
| 2 007 | 0 | 2 109 |
| 11 138 | 0 | 1 821 |
| 9 203 | 0 | 973 |
| 10 095 | 0 | 2 602 |
| 4 896 | 0 | 124 |
| 11 008 | 0 | 3 918 |
| 5 688 | 0 | 6 754 |

Nous obtenons à peu près les mêmes résultats.

Une autre hypothèse pouvant expliquer la difficulté d’obtenir des résultats est la suivante :

Dans le 3e environnement, les paramètres évoluent à l’initialisation, c’est-à-dire au moment où la graine est fixée. Son instance d’entrainement et son instance de test doivent donc avoir des paramètres identiques. L’environnement qui évolue au fil du temps voit ses paramètres évoluer petit à petit. Peut-être que lui se fait tester sur une instance avec des paramètres différents.

### Vérification de la seconde hypothèse

On fait en sorte que l’environnement qui évolue génère des instances aussi similaires au cours du temps que l’environnement qui évolue à l’initialisation.

Pour cela, on va créer une graine différente pour chaque étape :

|  |
| --- |
| self.seeds = [] for i in range(self.duration):  seeds.append(i) |

Et elles seront appliquées respectivement à chaque étape :

|  |
| --- |
| seed = self.seeds[self.\_state % self.duration] random.seed(seed) np.random.seed(seed) |

|  |  |
| --- | --- |
| Tests **avant** apprentissage | Tests **après** apprentissage |
| 3 638 | 0 |
| 6 661 | 0 |
| 5 676 | 0 |
| 2 761 | 0 |
| 3 110 | 0 |
| 1 836 | 0 |
| 3 493 | 0 |
| 2 958 | 0 |
| 1 291 | 0 |
| 3 304 | 0 |

Ces résultats montrent que bien que l’environnement soit viable, le fait d’apprendre sur un environnement dont les paramètres évoluent au fil du temps est bien plus complexe. En effet, un agent basé sur l’algorithme SAC obtient de meilleurs résultats (en politique d’exploration) avant d’avoir appris qu’après (en politique cupide comme en politique d’exploration). Cela signifie explicitement que l’agent est capable d’obtenir des résultats lorsqu’il apprend, mais pas lorsqu’il cherche à maximiser son résultat à partir de ce qu’il a appris.

### Ralentir l’évolution et remesurer ?

**Seulement si j’ai le temps à la fin, sinon tant pis**

## Inertie face à une corrélation temporaire

L’expérience n’a pu avoir lieu pour les mêmes raisons que l’expérience de l’ajout d’une variable importante invisible qui évolue : l’interruption de l’exécution des tests sans levée d’exception ni affichage de message d’erreur.

## Biais du rasoir d’Ockham

Toutes les expérimentations qui ont nécessité des observations ou des actions à plus d’une dimension avec l’algorithme SAC s’interrompent sans explication. Pour cette raison, nous n’allons malheureusement pas investir de temps sur cette expérience.

# Conclusion

**A REDIGER**

# Sources

* https://scholar.google.com/scholar?hl=fr&as\_sdt=0%2C5&q=reinforcement+learning+causality
* En 2009 : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2713351/
* 2019 : https://arxiv.org/abs/1901.08162
* 2018 : https://ieeexplore.ieee.org/abstract/document/8115277
* ? : https://books.google.fr/books?hl=fr&lr=&id=2qt0DgAAQBAJ&oi=fnd&pg=PA295&dq=reinforcement+learning+causality&ots=aypw5lcR00&sig=Buj0QQOXRdF6\_rFoCpeov9HdVYM&redir\_esc=y#v=onepage&q=reinforcement%20learning%20causality&f=false

# VRAC

## Résumés de publications

http://www.cems.uwe.ac.uk/~irjohnso/coursenotes/uqc832/tr-bias.pdf

* **Solution** : réduction de la variance
* **Problème** : coûteux en ressources (pour l’époque ?)
* **Comment l’utiliser** : La question a déjà été traitée, mais il y a 25 ans, et seulement pour les régressions, et ça coûtait trop de ressources.
* **Peut-être toujours d’actualité pour le renforcement ?** De plus, si on combine une réduction de variance avec un algo plus récent et moins coûteux, ça pourrait être pas mal.

“Randomization is paradoxical, because at first glance it seems to increase variance by deliberately introducing variation into the splits in the decision tree.”

https://arxiv.org/pdf/cmp-lg/9612001.pdf

https://people.csail.mit.edu/malte/pub/papers/2019-iclr-variance.pdf

https://dl.acm.org/doi/10.5555/3305381.3305400

à propos de la baisse de la variance grâce à l’algo DQN

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5722032/

Comment différencier (pas exactement mais souvent) corrélation de causalité en stats “normales”

Randomisation mendelienne

https://people.csail.mit.edu/malte/pub/papers/2019-iclr-variance.pdf

https://dl.acm.org/doi/10.5555/3305381.3305400

https://academic.oup.com/ije/article/48/3/691/5132989

La randomisation mendelienne est sensible au biais de sélection

https://arxiv.org/abs/1908.02983

No entiendo todo, mais il y a peut-être quelque chose à en tirer pour induire un biais de confirmation

https://arxiv.org/pdf/1703.02702.pdf

“Robust Adversarial Reinforcement Learning” -> des choses à en tirer ?

Différents algos de RL

https://spinningup.openai.com/en/latest/spinningup/rl\_intro2.html

## Autres sources

https://fr.wikipedia.org/wiki/Dilemme\_biais-variance

https://fr.wikipedia.org/wiki/Biais\_algorithmique#Biais\_cognitifs

https://www.ibm.com/blogs/ibm-france/2019/09/26/apprentissage-automatique-et-biais/

http://www.prclaudeberaud.fr/?129-erreur-ecologique-erreur-atomiste-lepidemiologie-contextuelle