Transfert de Tâches en Apprentissage par Renforcement: Apprentissage Auto-Supervisé de Politiques Généralistes

Louis Bagot
louis.bagot@univ-lyon1.fr

Présentation à MAFTEC + SMAA + ACE
22 Janvier 2026

Objectifs de cette présentation

- Présenter les avancées en Transfert de Tâches en RL
- Tisser des liens avec les autres branches du RL / Deep Learning

Objectifs de cette présentation

- Présenter les avancées en Transfert de Tâches en RL
- Tisser des liens avec les autres branches du RL / Deep Learning
  - Exploration et Motivation Intrinsèque
  - Apprentissage de compétences et options, RL hiérarchique
  - RL Multi-objectif

Objectifs de cette présentation

- Présenter les avancées en Transfert de Tâches en RL
- Tisser des liens avec les autres branches du RL / Deep Learning
  - Exploration et Motivation Intrinsèque
  - Apprentissage de compétences et options, RL hiérarchique
  - RL Multi-objectif
  - RL Inversé et Clonage de Politique
  - Interprétabilité et Safe RL

Objectifs de cette présentation

- Présenter les avancées en Transfert de Tâches en RL
- Tisser des liens avec les autres branches du RL / Deep Learning
  - Exploration et Motivation Intrinsèque
  - Apprentissage de compétences et options, RL hiérarchique
  - RL Multi-objectif
  - RL Inversé et Clonage de Politique
  - Interprétabilité et Safe RL
  - Apprentissage auto-supervisé

Objectifs de cette présentation

- Présenter les avancées en Transfert de Tâches en RL
- Tisser des liens avec les autres branches du RL / Deep Learning
  - 🔗 Exploration et Motivation Intrinsèque
  - 🔗 Apprentissage de compétences et options, RL hiérarchique
  - 🔗 RL Multi-objectif
  - 🔗 RL Inversé et Clonage de Politique
  - 🔗 Interprétabilité et Safe RL
  - 🔗 Apprentissage auto-supervisé
  - → faire naître des idées/collaborations?

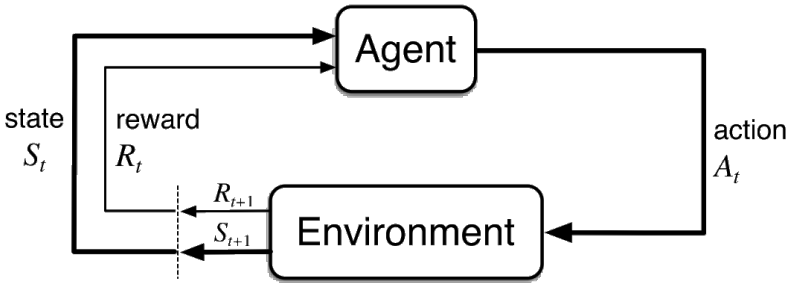
Objectifs de cette présentation

- Présenter les avancées en Transfert de Tâches en RL
- Tisser des liens avec les autres branches du RL / Deep Learning
  - 🔗 Exploration et Motivation Intrinsèque
  - 🔗 Apprentissage de compétences et options, RL hiérarchique
  - 🔗 RL Multi-objectif
  - 🔗 RL Inversé et Clonage de Politique
  - 🔗 Interprétabilité et Safe RL
  - 🔗 Apprentissage auto-supervisé
  - → faire naître des idées/collaborations?
- Présenter un peu mon travail (en thèse à IDLab Université d'Anvers)

Déroulé

1. **Introduction: Problème de Transfert de Tâches en RL**
2. **Approche générale en RL Zéro-Shot**
3. **Familles de Méthodes de Zéro-Shot en RL**
4. **Méthodes découlant de Successor Features**
5. **Conclusion**

Apprentissage par Renforcement

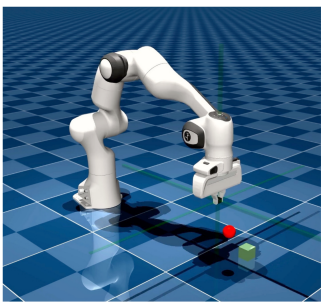


(Sutton et Barto, 2018)

Processus de Décision Markovien (MDP)

MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, r, \gamma, p, \mu_0\}$

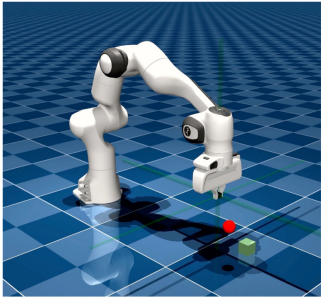
- $\mathcal{S}$: espace d'états
- $\mathcal{A}$: espace d'actions
- $r(s, a)$: fonction de récompense
- $p\left(s' \mid s, a\right)$: fonction de transition
- μ_0: distribution d'états initiaux



Transfert en RL:
Modification de n'importe quel élément du MDP

MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, r, \gamma, p, \mu_0\}$

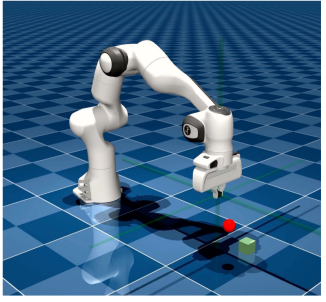
- $\mathcal{S}$: espace d'états
- $\mathcal{A}$: espace d'actions
- $r(s, a)$: fonction de récompense
- $p(s' \mid s, a)$: fonction de transition
- μ_0: distribution d'états initiaux



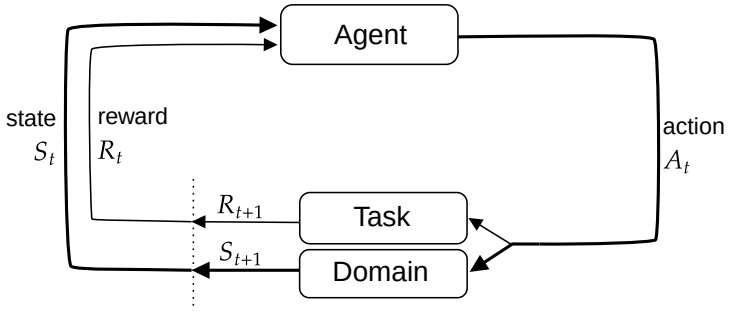
Transfert *de Tâche* en RL: Modification de la fonction de *récompense*

MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, r, \gamma, p, \mu_0\}$

- $\mathcal{S}$: espace d'états
- $\mathcal{A}$: espace d'actions
- $r(s, a)$**: fonction de récompense**
- $p\left(s' \mid s, a\right)$: fonction de transition
- μ_0: distribution d'états initiaux



Tâche et Domaine



Objectif général: maximiser rapidement les récompenses de la nouvelle tâche.

Scénarios de Transfert de Tâches

Scénarios de Transfert de Tâches

- Il y a un lien (ou non) entre les Tâches successives

Scénarios de Transfert de Tâches

- Il y a un lien (ou non) entre les Tâches successives
- La Tâche change doucement (ou subitement)

Scénarios de Transfert de Tâches

- Il y a un lien (ou non) entre les Tâches successives
- La Tâche change doucement (ou subitement)
- L'Agent est au courant (ou non) du changement

Scénarios de Transfert de Tâches

- Il y a un lien (ou non) entre les Tâches successives
- La Tâche change doucement (ou subitement)
- L'Agent est au courant (ou non) du changement
- L'Agent a le droit (ou non) de s'entraîner sur la nouvelle Tâche

Intérêt de cette présentation:
Connaissances sur le Domaine

Intérêt de cette présentation:
Connaissances sur le Domaine

- Il y a un lien (**ou non**) entre les Tâches successives
- La Tâche change doucement (**ou subitement**)
- L'Agent est **au courant** (ou non) du changement
- L'Agent a le droit (**ou non**) de s'entraîner sur la nouvelle Tâche

Scénario de *Zéro-shot* RL

Phase de Pré-entraînement:

- L'Agent a accès à un gros dataset d'interactions s, a, s'; *sans Tâche*
- L'Agent peut s'entraîner dessus pendant longtemps ("offline")

Scénario de *Zéro-shot* RL

Phase de Pré-entraînement:

- L'Agent a accès à un gros dataset d'interactions s, a, s'; *sans Tâche*
- L'Agent peut s'entraîner dessus pendant longtemps ("offline")

Phase de Transfert:

- On fournit un dataset de paires état-récompense labélisé (nouvelle Tâche)
- L'Agent doit proposer la meilleure politique possible *sans apprentissage*

Scénario de *Zéro-shot* RL

Phase de Pré-entraînement:

- L'Agent a accès à un gros dataset d'interactions s, a, s'; *sans Tâche*
- L'Agent peut s'entraîner dessus pendant longtemps ("offline")

Phase de Transfert:

- On fournit un dataset de paires état-récompense labélisé (nouvelle Tâche)
- L'Agent doit proposer la meilleure politique possible *sans apprentissage*

$\rightarrow$ il est tout à fait possible de relaxer des contraintes

Scénario de *Zéro-shot* RL

Phase de Pré-entraînement:

- L'Agent a accès à un gros dataset d'interactions s, a, s'; *sans Tâche*
- L'Agent peut s'entraîner dessus pendant longtemps ("offline")

Phase de Transfert:

- On fournit un dataset de paires état-récompense labélisé (nouvelle Tâche)
- L'Agent doit proposer la meilleure politique possible *sans apprentissage*

$\rightarrow$ il est tout à fait possible de relaxer des contraintes

🔗 Similarité avec l'apprentissage auto-supervisé!

Déroulé

Overview

Problèmes d'Exploration en RL:

- Exploration vs Exploitation
- Récompenses éparses/tardives: pas de signal, rien à faire
- Exploration intelligente du domaine

Motivation Intrinsèque à la rescousse

Motivation Intrinsèque: générer de fausses récompenses pour motiver l'exploration (Aubret et al., 2023)

- Nouveauté d'état (Bellemare at al., 2016; Burda et al., 2018)
- Curiosité / Surprise du modèle (Burda et al., 2018)
- Progrès d'apprentissage (Oudeyer and Kaplan, 2009)

Motivation Intrinsèque: au delà de l'Exploration

Motivation Intrinsèque: au delà de l'Exploration

Imaginons l'ensemble des fonctions de récompense

Motivation Intrinsèque: au delà de l'Exploration

Imaginons l'ensemble des fonctions de récompense

$\rightarrow$ Immensité de comportements/représentations apprenables!

- Atteindre des états ou objectifs
- Eviter des états
- Min/maximiser des features

Motivation Intrinsèque: au delà de l'Exploration

Imaginons l'ensemble des fonctions de récompense

$\rightarrow$ Immensité de comportements/représentations apprenables!

- Atteindre des états ou objectifs
- Eviter des états
- Min/maximiser des features

$\rightarrow$ Quelles récompenses sont *utiles*?

Motivation Intrinsèque: au delà de l'Exploration

Imaginons l'ensemble des fonctions de récompense

$\rightarrow$ Immensité de comportements/représentations apprenables!

- Atteindre des états ou objectifs
- Eviter des états
- Min/maximiser des features

$\rightarrow$ Quelles récompenses sont *utiles*?

🔗 Concept similaire aux *Tâches prétexte* en auto-supervisé!

De la Motivation Intrinsèque à l'Apprentissage de Compétences

De la Motivation Intrinsèque à l'Apprentissage de Compétences

- Est-ce que je peux utiliser plusieurs récompenses intrinsèques?
  - *off-policy*
  - infinité de comportements apprenables depuis s, a, s'!

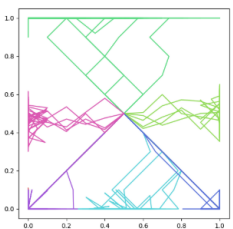
De la Motivation Intrinsèque à l'Apprentissage de Compétences

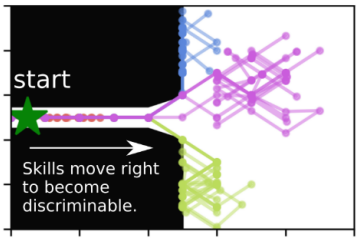
- Est-ce que je peux utiliser plusieurs récompenses intrinsèques?
  - *off-policy*
  - infinité de comportements apprenables depuis s, a, s'!
- Quelle *cohésion* entre les récompenses/politiques?

  $\rightarrow$ éviter la redondance

DIAYN: Apprentissage de compétences via la diversité

Idée simple “proof of concept”: maximiser la *diversité* (Eysenbach et al. 2018)

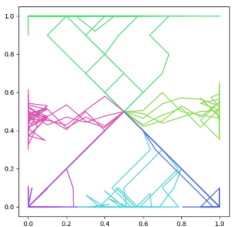


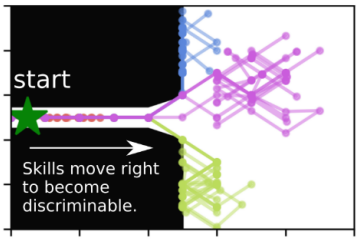


$\rightarrow$ Notion d’ensemble de *compétences* acquises dans le Domaine!

DIAYN: Apprentissage de compétences via la diversité

Idée simple “proof of concept”: maximiser la *diversité* (Eysenbach et al. 2018)





$\rightarrow$ Notion d’ensemble de *compétences* acquises dans le Domaine!

Chaque compétence π_k résout une récompense intrinsèque
$\rightarrow$ peut-on les mettre en commun dans $\pi\left(\cdot, k\right)$?

Overview

Politiques conditionnées par les Tâches

Zéro-Shot RL: je veux résoudre des Tâches données par une fct récompense r.

Politiques conditionnées par les Tâches

Zéro-Shot RL: je veux résoudre des Tâches données par une fct récompense r.

Idée principale:

1. Trouver une procédure d'*encodage* et *décodage* de r vers $\boldsymbol{z_r} \in \mathbb{R}^d$

Politiques conditionnées par les Tâches

Zéro-Shot RL: je veux résoudre des Tâches données par une fct récompense r.

Idée principale:

1. Trouver une procédure d'*encodage* et *décodage* de r vers $\boldsymbol{z_r} \in \mathbb{R}^d$
2. Pré-training: tirer aléatoirement des $\boldsymbol{z}$ et entraîner $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z}\right)$

Politiques conditionnées par les Tâches

Zéro-Shot RL: je veux résoudre des Tâches données par une fct récompense r.

Idée principale:

1. Trouver une procédure d'*encodage* et *décodage* de r vers $\boldsymbol{z}_{\boldsymbol{r}} \in \mathbb{R}^d$
2. Pré-training: tirer aléatoirement des $\boldsymbol{z}$ et entraîner $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z}\right)$
   - tirer un $\boldsymbol{z}$ dans la boule unité (ou exploiter a priori)
   - générer $\hat{r}_{\boldsymbol{z}}$ à partir de $\boldsymbol{z}$ avec le décodage

     $\rightarrow$ 🔗 Motivation Intrinsèque!
   - RL: maximiser la récompense (interaction $\left(s, a, s', \hat{r}_{\boldsymbol{z}}\right)$)

Politiques conditionnées par les Tâches

Zéro-Shot RL: je veux résoudre des Tâches données par une fct récompense r.

Idée principale:

1. Trouver une procédure d'*encodage* et *décodage* de r vers $\boldsymbol{z_r} \in \mathbb{R}^d$
2. Pré-training: tirer aléatoirement des $\boldsymbol{z}$ et entraîner $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z}\right)$
3. Transfert: compresser r en $\boldsymbol{z_r}$ et utiliser $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z_r}\right)$

Liens avec d'autres branches du RL

$$\pi\left(a \mid s, \boldsymbol{z}\right)$$

- Chaque $\hat{r}_{\boldsymbol{z}}$ est une récompense intrinsèque
- Chaque choix de $\boldsymbol{z}$ génère une compétence distincte
- Hiérarchique: on peut contrôler l'agent avec $\boldsymbol{z}$!

Est-ce raisonnable?

1. Trouver une procédure d'*encodage* et *décodage* de r vers $\boldsymbol{z_r} \in \mathbb{R}^d$
2. Pré-training: tirer aléatoirement des $\boldsymbol{z}$ et entraîner $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z}\right)$
3. Transfert: compresser r en $\boldsymbol{z_r}$ et utiliser $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z_r}\right)$

Est-ce raisonnable?

1. Trouver une procédure d'*encodage* et *décodage* de r vers $\boldsymbol{z_r} \in \mathbb{R}^d$
2. Pré-training: tirer aléatoirement des $\boldsymbol{z}$ et entraîner $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z}\right)$
3. Transfert: compresser r en $\boldsymbol{z_r}$ et utiliser $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z_r}\right)$

- la procédure encodage/décodage perd nécessairement de l'information
  - impossible de parfaitement résumer r en général
  - on doit faire des hypothèses sur l'ensemble des fonctions récompenses

Est-ce raisonnable?

1. Trouver une procédure d'*encodage* et *décodage* de r vers $\boldsymbol{z_r} \in \mathbb{R}^d$
2. Pré-training: tirer aléatoirement des $\boldsymbol{z}$ et entraîner $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z}\right)$
3. Transfert: compresser r en $\boldsymbol{z_r}$ et utiliser $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z_r}\right)$

- la procédure encodage/décodage perd nécessairement de l'information
- le réseau de neurones $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z}\right)$ doit emmagasiner une quantité monstrueuse de données et $\boldsymbol{z}$ (croît avec d)
  $\rightarrow$ gros réseau, entraîné longtemps, sur beaucoup de données

Est-ce raisonnable?

1. Trouver une procédure d'*encodage* et *décodage* de r vers $\boldsymbol{z_r} \in \mathbb{R}^d$
2. Pré-training: tirer aléatoirement des $\boldsymbol{z}$ et entraîner $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z}\right)$
3. Transfert: compresser r en $\boldsymbol{z_r}$ et utiliser $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z_r}\right)$

- la procédure encodage/décodage perd nécessairement de l'information
- le réseau de neurones $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z}\right)$ doit emmagasiner une quantité monstrueuse de données et $\boldsymbol{z}$ (croît avec d)
  $\rightarrow$ gros réseau, entraîné longtemps, sur beaucoup de données
- généralisation sur tout l'espace de $\boldsymbol{z}$ compliquée (hors miracles)

Est-ce raisonnable?

1. Trouver une procédure d'*encodage* et *décodage* de r vers $\boldsymbol{z_r} \in \mathbb{R}^d$
2. Pré-training: tirer aléatoirement des $\boldsymbol{z}$ et entraîner $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z}\right)$
3. Transfert: compresser r en $\boldsymbol{z_r}$ et utiliser $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z_r}\right)$

- la procédure encodage/décodage perd nécessairement de l'information
- le réseau de neurones $\pi_{\boldsymbol{\theta}}\left(a \mid s, \boldsymbol{z}\right)$ doit emmagasiner une quantité monstrueuse de données et $\boldsymbol{z}$ (croît avec d)
  $\rightarrow$ gros réseau, entraîné longtemps, sur beaucoup de données
- généralisation sur tout l'espace de $\boldsymbol{z}$ compliquée (hors miracles)
- redondance: énormément de $\boldsymbol{z}$ ont les mêmes $\pi\left(a \mid s, \boldsymbol{z}\right)$

Degré de liberté de l'approche

1. Trouver une procédure d'*encodage* et *décodage* de r vers $\boldsymbol{z_r} \in \mathbb{R}^d$
2. Pré-training: tirer aléatoirement des $\boldsymbol{z}$ et entraîner $\pi\left(a \mid s, \boldsymbol{z}\right)$
3. Transfert: compresser r en $\boldsymbol{z_r}$ et utiliser $\pi\left(a \mid s, \boldsymbol{z_r}\right)$

Degré de liberté de l'approche

1. **Trouver une procédure d'*encodage* et *décodage* de r vers $\boldsymbol{z_r} \in \mathbb{R}^d$**
2. Pré-training: tirer aléatoirement des $\boldsymbol{z}$ et entraîner $\pi\left(a \mid s, \boldsymbol{z}\right)$
3. Transfert: compresser r en $\boldsymbol{z_r}$ et utiliser $\pi\left(a \mid s, \boldsymbol{z_r}\right)$

Degré de liberté de l'approche

1. Trouver une procédure d'*encodage* et *décodage* de r vers $\boldsymbol{z_r} \in \mathbb{R}^d$
2. Pré-training: tirer aléatoirement des $\boldsymbol{z}$ et entraîner $\pi\left(a \mid s, \boldsymbol{z}\right)$
3. Transfert: compresser r en $\boldsymbol{z_r}$ et utiliser $\pi\left(a \mid s, \boldsymbol{z_r}\right)$

$\rightarrow$ Question critique: **quelle procédure d'encodage/décodage choisir?**

Déroulé

Overview

Cas spécial du Conditionnement par Objectif (*goal*)

Si on choisit $\boldsymbol{z} = g \in \mathcal{S}$ (ou espace latent d'états), $\pi\left(a \mid s, g\right)$

- π cherche simplement à *atteindre g*

Cas spécial du Conditionnement par Objectif (*goal*)

Si on choisit $\boldsymbol{z} = g \in \mathcal{S}$ (ou espace latent d'états), $\pi\left(a \mid s, g\right)$

- π cherche simplement à *atteindre* g
- Récompense sparse mais pas un problème: (Andrychowicz et al. 2017)
  - pour chaque transition $\left(s, a, s'\right)$, on peut prétendre que $g = s'$

Cas spécial du Conditionnement par Objectif (*goal*)

Si on choisit $\boldsymbol{z} = g \in \mathcal{S}$ (ou espace latent d'états), $\pi\left(a \mid s, g\right)$

- π cherche simplement à *atteindre g*
- Récompense sparse mais pas un problème: (Andrychowicz et al. 2017)
  - pour chaque transition (s, a, s'), on peut prétendre que $g = s'$

  $\rightarrow$ toute transition peut être une réussite!
  - même idée que "toute transition peut être vue du pdv de toute récompense"

Cas spécial du Conditionnement par Objectif (*goal*)

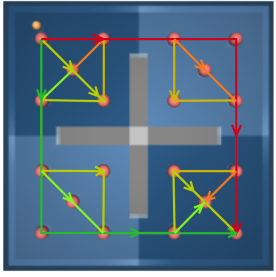
Si on choisit $\boldsymbol{z} = g \in \mathcal{S}$ (ou espace latent d'états), $\pi\left(a \mid s, g\right)$

- π cherche simplement à *atteindre g*
- Pas vraiment du zero-shot
  - "encodage": $g = \arg\max_s r(s)$
  - "décodage": $\hat{r}(s) = \mathbb{1}_{s=g}$ (ou pareil -1)

Cas spécial du Conditionnement par Objectif (*goal*)

Si on choisit $\boldsymbol{z} = g \in \mathcal{S}$ (ou espace latent d'états), $\pi\left(a \mid s, g\right)$

- π cherche simplement à *atteindre g*
- Pas vraiment du zero-shot
  - "encodage": $g = \arg\max_s r(s)$
  - "décodage": $\hat{r}(s) = \mathbb{1}_{s=g}$ (ou pareil -1)
  - plein de trajectoires peuvent mener au même état objectif



Cas spécial du Conditionnement par Objectif (*goal*)

Si on choisit $\boldsymbol{z} = g \in \mathcal{S}$ (ou espace latent d'états), $\pi\left(a \mid s, g\right)$

- π cherche simplement à *atteindre* g
- Pas vraiment du zero-shot
  - "encodage": $g = \arg\max_s r(s)$
  - "décodage": $\hat{r}(s) = \mathbb{1}_{s=g}$ (ou pareil -1)
- formulation récente en contrastif (Eysenbach et al., 2022, Wang et al., 2025)
  🔗 lien avec l'auto-supervisé!

Overview

Utiliser un réseau pour faire l'encodage/décodage

Version la plus brute: *Functional Reward Encodings* (Frans et al., 2024)

- réseau encodeur $\xi_{\boldsymbol{\theta}}\left(\{s_i, r(s_i)\}_i\right) = \boldsymbol{z_r}$
- réseau décodeur $\xi_{\boldsymbol{w}}^{-1}\left(\{\boldsymbol{z_r}, s_i\}_i\right) = \{\hat{r}\left(s_i\right)\}_i$

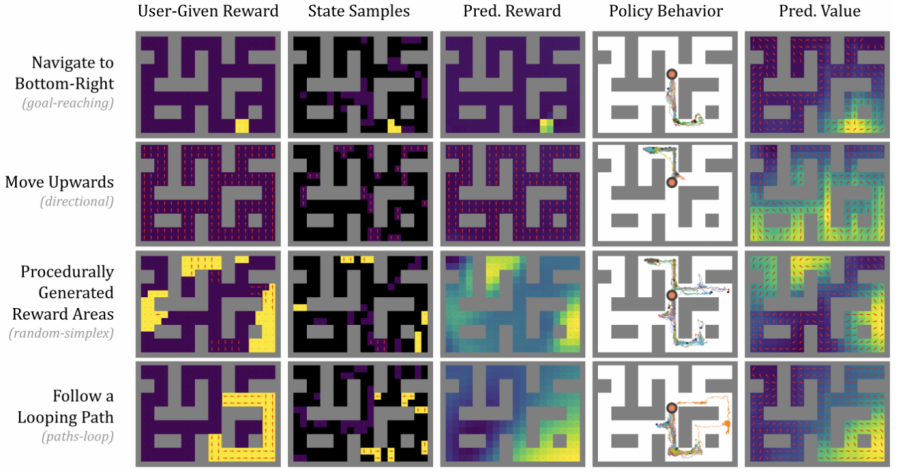
Utiliser un réseau pour faire l'encodage/décodage

Version la plus brute: *Functional Reward Encodings* (Frans et al., 2024)

- réseau encodeur $\xi_{\boldsymbol{\theta}}\left(\{s_i, r(s_i)\}_i\right) = \boldsymbol{z_r}$
- réseau décodeur $\xi_{\boldsymbol{w}}^{-1}\left(\{\boldsymbol{z_r}, s_i\}_i\right) = \{\hat{r}\left(s_i\right)\}_i$

- setup le plus flexible
- boîte noire absolue
- pas évident de tirer aléatoirement des $\boldsymbol{z}$ pour l'entraînement

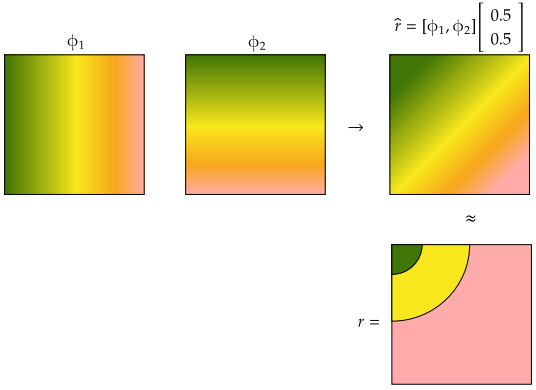
Functional Reward Encodings: visualisation



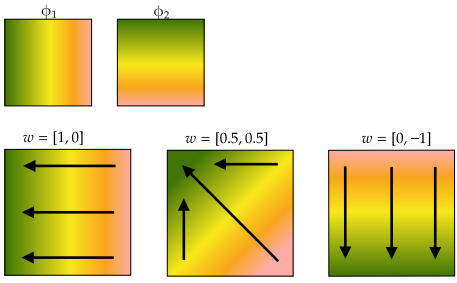
Overview

Successor Features: Intuition (Barreto et al., 2017, 2018)

Approximation linéaire de la récompense avec *features* $\{\phi_i\}_i$: $r \approx \boldsymbol{\phi} \cdot \mathbf{z_r}$

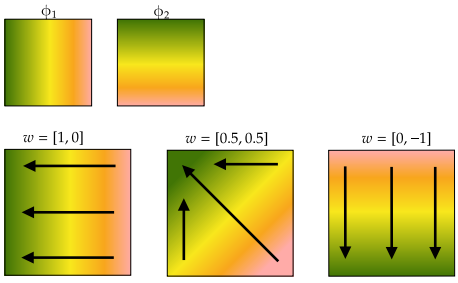


Conditionnement sur le vecteur d'approx. linéaire



Observation: pour une base ϕ donnée, ***z* encode la récompense**.

Conditionnement sur le vecteur d'approx. linéaire



Observation: pour une base ϕ donnée, ***z* encode la récompense**.
$\rightarrow$ Entraîner $\pi(a \mid s, \boldsymbol{z})$.

Successor Features comme procédure encodage/décodage

Pour une base des récompenses $\boldsymbol{\phi} : \mathcal{S} \to \mathbb{R}^d$,

- Encodage: moindre carrés $\xi\left(\{s_i, r(s_i)\}_i\right) = \boldsymbol{z_r} = \arg\min_{\boldsymbol{z}} \|r - \boldsymbol{\phi} \cdot \boldsymbol{z}\|_2$
- Décodage: approx. lin. $\xi^{-1}\left(s, \boldsymbol{z_r}\right) = \boldsymbol{\phi}\left(s\right) \cdot \boldsymbol{z_r}$

Successor Features comme procédure encodage/décodage

Pour une base des récompenses $\boldsymbol{\phi} : \mathcal{S} \to \mathbb{R}^d$,

- Encodage: moindre carrés $\xi\left(\{s_i, r(s_i)\}_i\right) = \boldsymbol{z_r} = \arg\min_{\boldsymbol{z}} \|r - \boldsymbol{\phi} \cdot \boldsymbol{z}\|_2$
- Décodage: approx. lin. $\xi^{-1}\left(s, \boldsymbol{z_r}\right) = \boldsymbol{\phi}\left(s\right) \cdot \boldsymbol{z_r}$
- Chaque z_i correspond à la **contribution du feature** ϕ_i
- $\pi\left(a \mid s, \boldsymbol{z}\right)$ apprend à maximiser les comb. lin. de features

🔗 Parallèles avec le RL Multi-Objectif

En RL Multi Objectif:

- Ensemble de récompenses $\boldsymbol{r} = [r_0, r_1, \dots r_d]$
- Version linéaire: maximiser une comb. lin. d'objectifs $\boldsymbol{r} \cdot \boldsymbol{z}$

🔗 Parallèles avec le RL Multi-Objectif

En RL Multi Objectif:

- Ensemble de récompenses $\boldsymbol{r} = [r_0, r_1, \ldots r_d]$
- Version linéaire: maximiser une comb. lin. d'objectifs $\boldsymbol{r} \cdot \boldsymbol{z}$

$\rightarrow$ Même setup, mais en SF on choisit l'ensemble de récompenses ϕ (eq. $\boldsymbol{r}$) (récompenses intrinsèques)

🔗 Parallèles avec le RL Multi-Objectif

En RL Multi Objectif:

- Ensemble de récompenses $\boldsymbol{r} = [r_0, r_1, \dots r_d]$
- Version linéaire: maximiser une comb. lin. d'objectifs $\boldsymbol{r} \cdot \boldsymbol{z}$

$\rightarrow$ Même setup, mais en SF on choisit l'ensemble de récompenses $\boldsymbol{\phi}$ (eq. $\boldsymbol{r}$) (récompenses intrinsèques)

Même notion de fonction de valeur vectorielle:

- Multi-Objectif: $\boldsymbol{V}^{\pi} = \mathbb{E}_{\pi}[\sum_t \boldsymbol{r}(S_t)]$
- Définition de *Successor* Features: $\boldsymbol{\psi}^{\pi} = \mathbb{E}_{\pi}[\sum_t \boldsymbol{\phi}(S_t)]$
- Et pour une comb. lin. $\boldsymbol{z}$, $v_z^{\pi} = \boldsymbol{V}^{\pi} \cdot \boldsymbol{z}$ (idem $\boldsymbol{\psi}$)

Successor Features comme procédure encodage/décodage

Pour une base des récompenses $\boldsymbol{\phi} : \mathcal{S} \to \mathbb{R}^d$,

- Chaque z_i correspond à la **contribution du feature** ϕ_i
- $\pi\left(a \mid s, \boldsymbol{z}\right)$ apprend à maximiser les comb. lin. de features

Successor Features comme procédure encodage/décodage

Pour une base des récompenses $\phi : \mathcal{S} \to \mathbb{R}^d$,

- Chaque z_i correspond à la **contribution du feature** ϕ_i
- $\pi\left(a \mid s, \boldsymbol{z}\right)$ apprend à maximiser les comb. lin. de features

- Beaucoup plus interprétable
- Beaucoup moins flexible

Successor Features comme procédure encodage/décodage

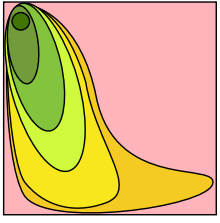
Pour une base des récompenses $\phi : \mathcal{S} \to \mathbb{R}^d$,

- Chaque z_i correspond à la **contribution du feature** ϕ_i
- $\pi\left(a \mid s, \boldsymbol{z}\right)$ apprend à maximiser les comb. lin. de features

- Beaucoup plus interprétable
- Beaucoup moins flexible
- ? **Quel choix de features/base linéaire ϕ?**

Quel choix de ϕ pour approximer r linéairement?

$$r \approx \boldsymbol{\phi} \cdot \boldsymbol{w}$$



$$\approx \begin{matrix} \boxed{?} \times w_1 \\ + \\ \boxed{?} \times w_2 \\ + \\ \boxed{?} \times w_3 \\ + \\ \boxed{?} \times w_4 \\ + \\ \boxed{?} \times w_5 \\ \vdots \end{matrix} = \boxed{?}$$

Déroulé

Rappel du problème

Rappel du problème

- Objectif: transfert zéro-shot
  - $\rightarrow$ extraire des connaissances du domaine pour résoudre toute Tâche

Rappel du problème

- Objectif: transfert zéro-shot
  - $\rightarrow$ extraire des connaissances du domaine pour résoudre toute Tâche

$\downarrow$

- Politiques conditionnées par la Tâche $\pi\left(a \mid s, \boldsymbol{z}\right)$
  - $\rightarrow$ besoin d'une procédure d'encodage/décodage de r vers $\boldsymbol{z}_r$

Rappel du problème

- Objectif: transfert zéro-shot
  - $\rightarrow$ extraire des connaissances du domaine pour résoudre toute Tâche

$\downarrow$

- Politiques conditionnées par la Tâche $\pi\left(a \mid s, \boldsymbol{z}\right)$
  - $\rightarrow$ besoin d'une procédure d'encodage/décodage de r vers $\boldsymbol{z_r}$

$\downarrow$

- Successor Features: approximation linéaire de récompense $r \approx \boldsymbol{\phi} \cdot \boldsymbol{z_r}$
  - $\rightarrow$ $\boldsymbol{z_r}$ encode l'importance de chaque feature ϕ_i dans r

Rappel du problème

- Objectif: transfert zéro-shot
  - $\rightarrow$ extraire des connaissances du domaine pour résoudre toute Tâche

$\downarrow$

- Politiques conditionnées par la Tâche $\pi\left(a \mid s, \boldsymbol{z}\right)$
  - $\rightarrow$ besoin d'une procédure d'encodage/décodage de r vers $\boldsymbol{z_r}$

$\downarrow$

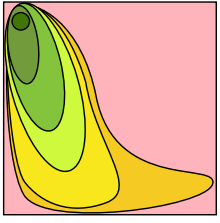
- Successor Features: approximation linéaire de récompense $r \approx \phi \cdot \boldsymbol{z_r}$
  - $\rightarrow$ $\boldsymbol{z_r}$ encode l'importance de chaque feature ϕ_i dans r

$\downarrow$

- Quel choix de features/base linéaire ϕ?

Quel choix de ϕ pour approximer r linéairement?

$$r \approx \boldsymbol{\phi} \cdot \boldsymbol{w}$$



$\approx$

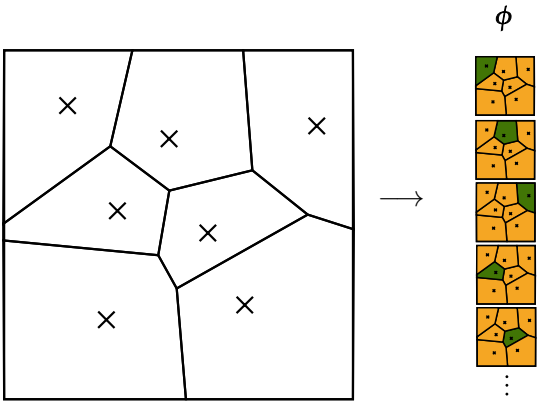
$$\begin{bmatrix} ? \\ ? \\ ? \\ ? \\ ? \\ \vdots \end{bmatrix} \begin{matrix} \times w_1 \\ + \\ \times w_2 \\ + \\ \times w_3 \\ + \\ \times w_4 \\ + \\ \times w_5 \end{matrix}$$

$=$

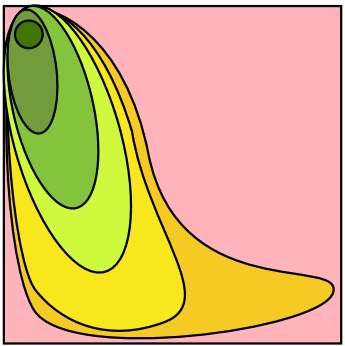
$?$

Overview

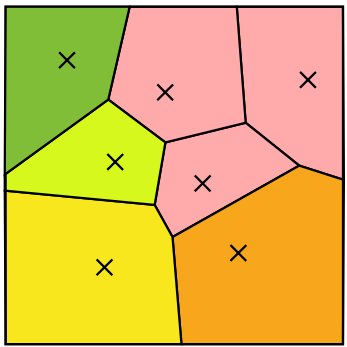
Occupation de cluster comme features



Approximation de récompense via clusters



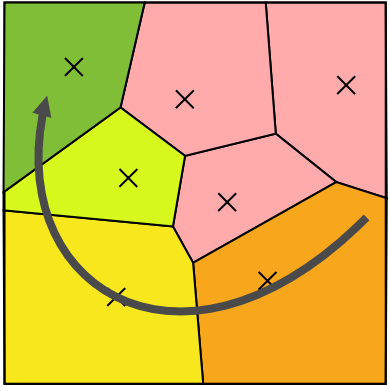
true reward r



reward approximation

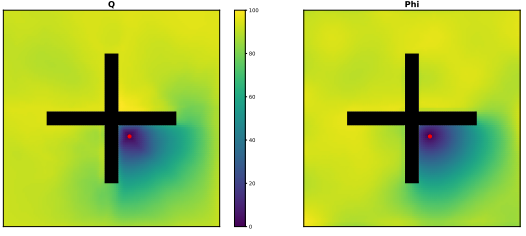
Politique conditionnée par les valeurs des clusters

$\pi\left(a \mid s, \boldsymbol{z}\right)$ conditionnée sur $\boldsymbol{z}$ avec z_i = récompense moyenne du cluster i

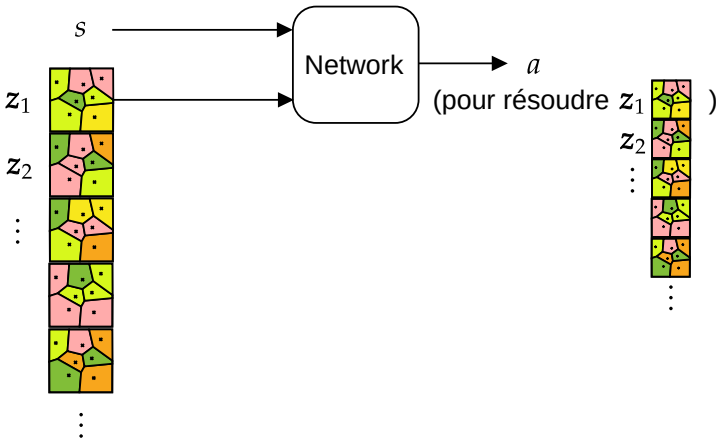


Notion de distance pour les clusters: Distance temporelle minimale

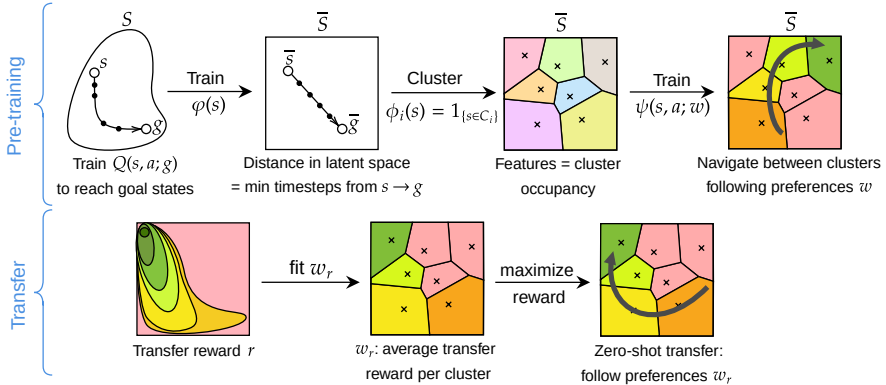
1. Entraîner un agent conditionné par l'objectif $\pi\left(a \mid s, g\right)$
2. Récompense prétexte $= -\mathbb{1}_{s \neq g}$
   $\rightarrow$ fonction de valeur = distance temporelle minimale
3. Apprendre un espace latent où dist. eudlidienne = dist. temporelle



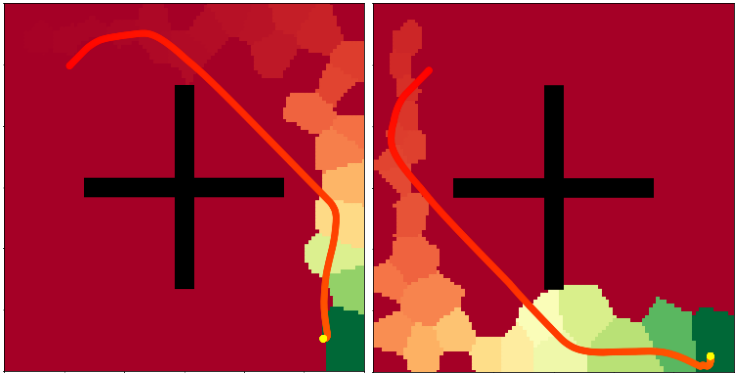
Entraînement du réseau: génération de *z* prétexte



Méthode complète: RL Zéro-shot avec features de clusters



Contrôle de l'agent via *z*

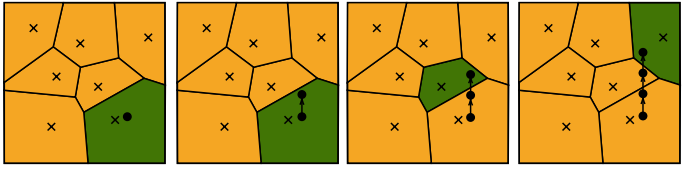


background: ce que l'agent prend en entrée
(vecteur 100 dimensions pour 100 clusters)

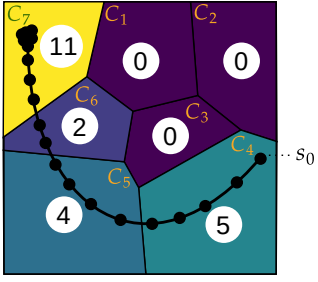
Successor Clusters: prédiction de visites futures de clusters

On a établi que $\psi^{\pi}(s_0) = \mathbb{E}_{\pi}[\sum_t \phi(S_t) \mid s_0]$.

$$\psi^{\pi}(s_0) = \phi(s_0) + \phi(s_1) + \phi(s_2) + \phi(s_3) + \dots$$

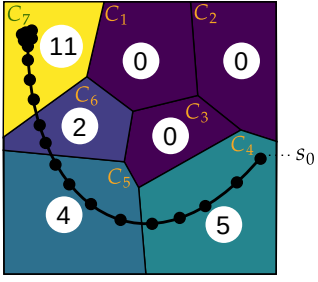


Successor Clusters sur tout une trajectoire



$$\psi^{\pi}(s_0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 5 \\ 4 \\ 2 \\ 11 \end{bmatrix}$$

Successor Clusters sur tout une trajectoire

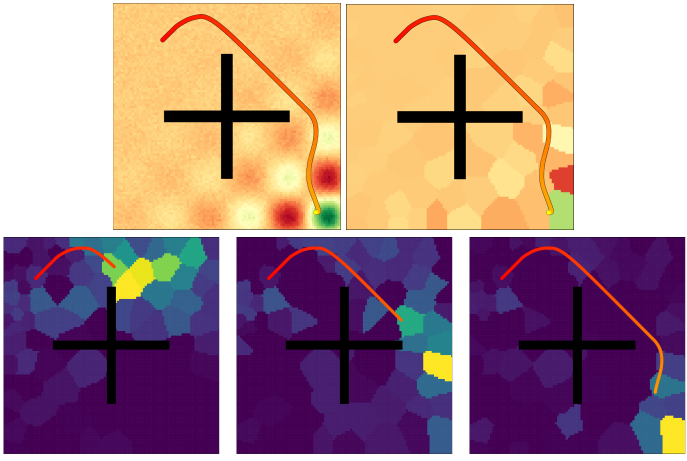


$$\psi^{\pi}(s_0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 5 \\ 4 \\ 2 \\ 11 \end{bmatrix}$$

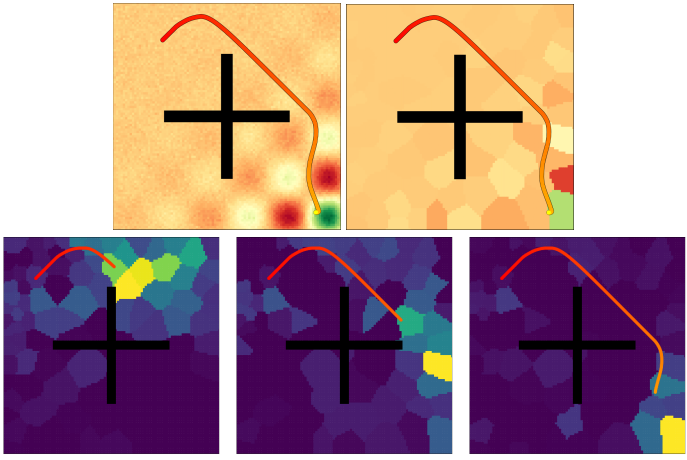
En général, $\psi(s_0, a_0, \boldsymbol{z}) = \mathbb{E}_{\pi(\cdot|\boldsymbol{z})}[\sum_t \phi(S_t) \mid s_0, a_0]$

(USFA, Borsa et al., 2018)

Successor Clusters: Exemple Zero-shot



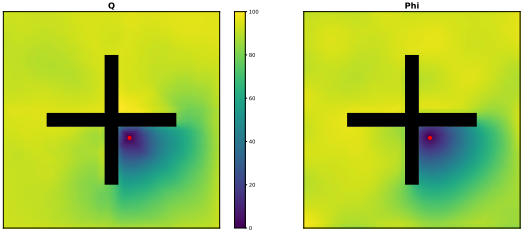
Successor Clusters: Exemple Zero-shot



🔗 Interprétabilité: l'agent prédit sa trajectoire future.

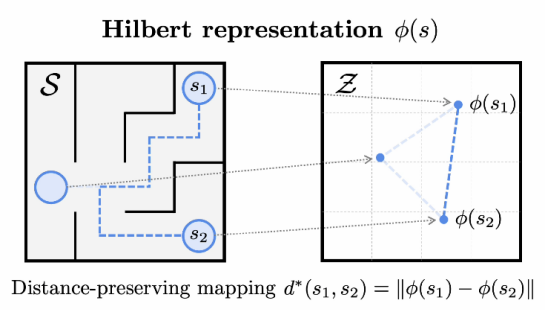
Overview

HILP: Autre utilisation de l'espace
"dist. euclidienne = dist. temporelle"



HILP: Autre utilisation de l'espace "dist. euclidienne = dist. temporelle"

- Quasiment le même espace latent que Successor Clusters



HILP: Autre utilisation de l'espace "dist. euclidienne = dist. temporelle"

- Idée clé: **tirer *z* directement de l'espace latent!**

HILP: Autre utilisation de l'espace "dist. euclidienne = dist. temporelle"

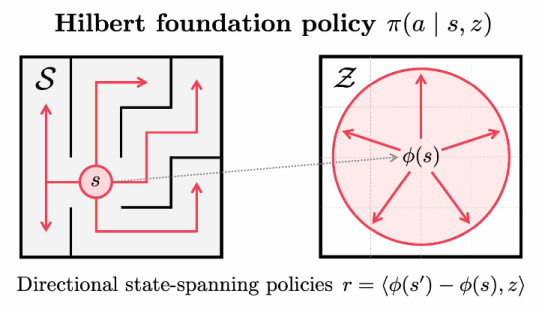
- Idée clé: **tirer *z* directement de l'espace latent!**
- Tout ***z*** (normalisé) correspond à une *direction* dans l'espace latent

HILP: Autre utilisation de l'espace "dist. euclidienne = dist. temporelle"

- Idée clé: **tirer *z* directement de l'espace latent!**
- Tout ***z*** (normalisé) correspond à une *direction* dans l'espace latent

$\rightarrow$ récompense pour suivre cette direction ***z***:

décodeur $r_z(s, a, s') = (\varphi(s') - \varphi(s)) \cdot \boldsymbol{z}$



HILP: Autre utilisation de l'espace "dist. euclidienne = dist. temporelle"

- Idée clé: **tirer z directement de l'espace latent!**
- Tout $\boldsymbol{z}$ (normalisé) correspond à une *direction* dans l'espace latent

$\rightarrow$ récompense pour suivre cette direction $\boldsymbol{z}$:

décodeur $r_z(s, a, s') = (\varphi(s') - \varphi(s)) \cdot \boldsymbol{z}$

- Entraîner la politique $\pi\left(a \mid s, \boldsymbol{z}\right)$ avec $\boldsymbol{z} \in \mathcal{Z} \subset \mathbb{R}^d$, l'espace latent
  - la base orthonormale de $\mathcal{Z}$ définit nos récompenses intrinsèques

HILP: Autre utilisation de l'espace "dist. euclidienne = dist. temporelle"

- Idée clé: **tirer z directement de l'espace latent!**
- Tout $\boldsymbol{z}$ (normalisé) correspond à une *direction* dans l'espace latent

$\rightarrow$ récompense pour suivre cette direction $\boldsymbol{z}$:

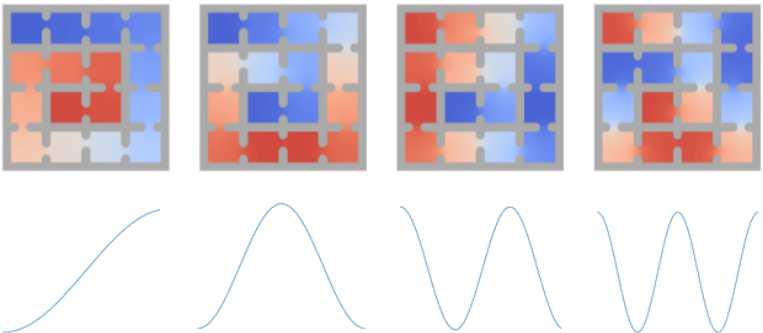
décodeur $r_z(s, a, s') = (\varphi(s') - \varphi(s)) \cdot \boldsymbol{z}$

- Entraîner la politique $\pi\left(a \mid s, \boldsymbol{z}\right)$ avec $\boldsymbol{z} \in \mathcal{Z} \subset \mathbb{R}^d$, l'espace latent
  - la base orthonormale de $\mathcal{Z}$ définit nos récompenses intrinsèques
- On peut faire du conditionnement d'objectif: $\boldsymbol{z}_g(s) = \varphi(g) - \varphi(s)$

  $\rightarrow$ vecteur de Tâche $\boldsymbol{z}$ *dynamique*!

Overview

Fonctions propres du Laplacien (K. Wang et al., 2021)



Equivalent MDP d'une base de Fourier $\longrightarrow$ très adaptées à une combinaison linéaire!

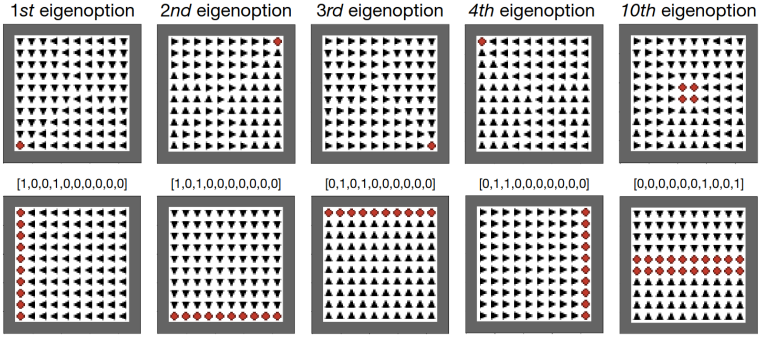
Exploration et Zéro-Shot RL avec le Laplacien

- Pour f_i chaque fonction propre on utilise $\phi = [f_1, f_2 \dots f_d]$
  - les f_i servent de motivation intrinsèque (Machado et al., 2023)

Exploration et Zéro-Shot RL avec le Laplacien

- Pour f_i chaque fonction propre on utilise $\phi = [f_1, f_2 \dots f_d]$
  - les f_i servent de motivation intrinsèque (Machado et al., 2023)
- z_i correspond à la constribution de chaque f_i
- On entraîne $\pi\left(a \mid s, \boldsymbol{z}\right)$.

Visualisation de l'agent "Laplacien"



Autre méthode intéressante: Forward-Backward Representations

(Touati et al., 2021, 2023)

pas le temps désolé!

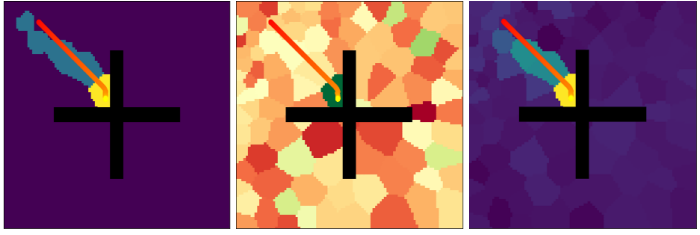
Overview

🔗 Inverse RL / Clonage de Politique

- Pour une trajectoire désirée $s_0, s_1, s_2, \ldots$
- Somme de features désirée $\Gamma = \sum_t \phi(s_t)$
- Optimiser $\boldsymbol{z}$ pour minimiser $\|\Gamma - \psi(s_0, \boldsymbol{z})\|$

🔗 Inverse RL / Clonage de Politique

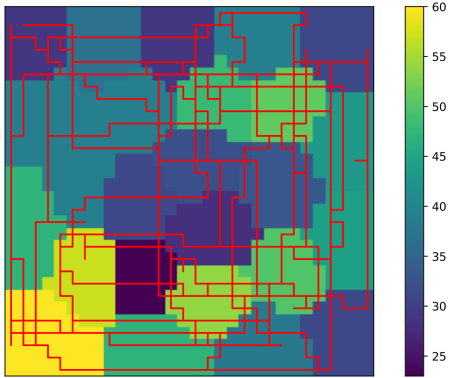
- Pour une trajectoire désirée $S_0, S_1, S_2, \ldots$
- Somme de features désirée $\Gamma = \sum_t \phi(S_t)$
- Optimiser $\boldsymbol{z}$ pour minimiser $\|\Gamma - \psi(S_0, \boldsymbol{z})\|$



trajectoire désirée Γ; $\boldsymbol{z}_\Gamma$ optimisé; politique et visites pour $\boldsymbol{z}_\Gamma$

Générer des politiques d'Exploration

Clusters: compter et punir chaque visite $\rightarrow \boldsymbol{z} = 1/[N(C_1), N(C_2), \ldots, N(C_d)]$



$\rightarrow$ Généralisation à venir...

🔗 Safe RL et Interprétabilité

Entraînement: on peut contrôler les features ϕ_i et $\boldsymbol{z}$

- Ajouter à la main des features/régions d'états dangereux
- Entraîner seulement pour les $\boldsymbol{z}$ qui nous intéressent
  $\rightarrow$ pas besoin de maximiser les features dangereux!

🔗 Safe RL et Interprétabilité

Entraînement: on peut contrôler les features ϕ_i et $\boldsymbol{z}$

- Ajouter à la main des features/régions d'états dangereux
- Entraîner seulement pour les $\boldsymbol{z}$ qui nous intéressent
  - $\rightarrow$ pas besoin de maximiser les features dangereux!

Transfert/exécution: on peut contrôler les visites prévues ψ_i

- Stopper l'agent en cas de prédiction trop élevée
- Prédiction encore très imparfaite

Déroulé

1. Introduction: Problème de Transfert de Tâches en RL
2. Approche générale en RL Zéro-Shot
3. Familles de Méthodes de Zéro-Shot en RL
4. Méthodes découlant de Successor Features
5. **Conclusion**

Résumé: méthodes abordées

- On veut extraire des politique et représentations du domaine pour le transfert
- Politique conditionnée: $\pi\left(a \mid s, \boldsymbol{z}\right)$
- Dépend d'une méthode d'encodage des tâches $r \longrightarrow \boldsymbol{z}$
  - Conditionnement d'objectifs: $\boldsymbol{z}$ est un état
  - Encodage brut à base de réseau: flexible mais obscur
  - Successor Features: décomposition linéaire

Résumé: liens tissés

- Exploration & Motivation: *ensemble* de récompenses intrinsèques
- Compétences & Hiérarchique: politiques fortes et contrôlables avec $\boldsymbol{z}$
  - problème: pas pensées pour être composées
- Multi-Objectif: même scénario, mais on choisit les récompenses
- RL Inversé & Clonage: on peut extraire le $\boldsymbol{z}$ qui suit une trajectoire
- RL Safe & Interprétabilité: l'agent prédit ses trajectoires
- Apprentissage auto-supervisé: on réfléchit à des tâches prétexte

Limites et futur

Plein de facteurs limitants:

- Objectif de $\pi\left(a \mid s, \boldsymbol{z}\right)$ très compliqué
  - $\rightarrow$ surtout vu les redondances dans $\boldsymbol{z}$
- Pas de compositionalité
  - $\rightarrow$ π essaie de tout résoudre seule!
- Setup de zero-shot trop simplifié
  - pas d'exploration
  - gros dataset & temps d'entraînement
  - transfert idéalisé

Merci pour votre attention!

Appendix: éléments complémentaires

Déroulé

Overview

Successor Features

Approximate reward

$$r \approx \boldsymbol{\phi} \cdot \boldsymbol{w}_r$$

Successor Features are q-values for the features ϕ_i:

$$\boldsymbol{\psi}_\pi(s,a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \boldsymbol{\phi}\left(S_{t+k+1}, A_{t+k+1}\right) \mid S_t = s, A_t = a\right].$$

Successor Features linearly break down the expected return:

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right] \\ &\approx \boldsymbol{\psi}_\pi(s,a) \cdot \boldsymbol{w}_r. \end{aligned}$$

Generating Optimal Policies with USFAs

Compute USFAs with Q learning:

$$\boldsymbol{\psi}(s, a; \boldsymbol{w}_r) \leftarrow \boldsymbol{\phi}(s, a) + \gamma \max \boldsymbol{\psi}(s', a^*; \boldsymbol{w}_r).$$

Obtain optimal policy with

$$\pi\left(s, \boldsymbol{w}_r\right) = \arg\max_a \boldsymbol{\psi}(s, a; \boldsymbol{w}_r) \cdot \boldsymbol{w}_r.$$

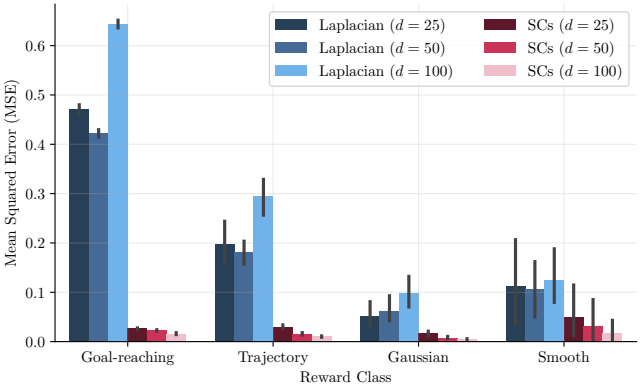
Successor Clusters predict future cluster visits

- Linear reward approximation: $r \approx \boldsymbol{\phi} \cdot \boldsymbol{w}_r$.
- Value function breaks down linearly: $q_\pi \approx \boldsymbol{\psi}_\pi \cdot \boldsymbol{w}_r$
  - $\rightarrow$ with $\psi_{\pi,i}$ the value function for ϕ_i.
    (Dayan, 1993; Barreto et al., 2017, 2018)

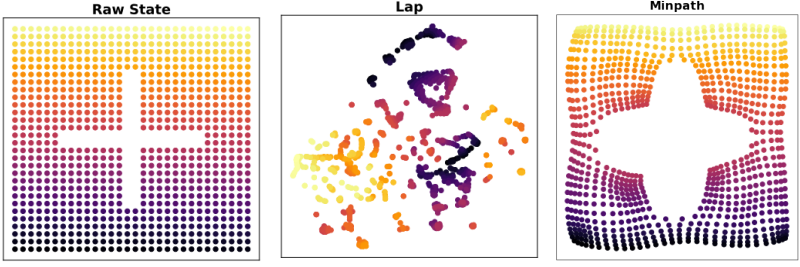
$$\boldsymbol{\psi}^{\pi}(s_0) = \boldsymbol{\phi}(s_0) + \boldsymbol{\phi}(s_1) + \boldsymbol{\phi}(s_2) + \boldsymbol{\phi}(s_3) + \dots$$

Overview

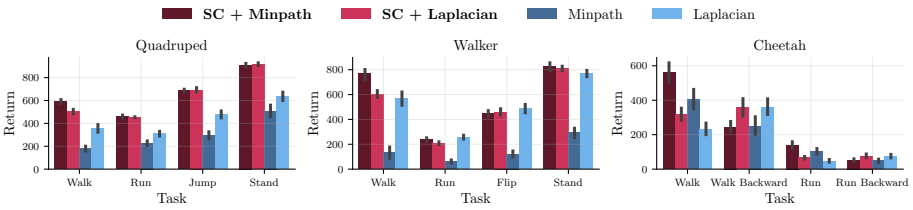
Reward Approximation

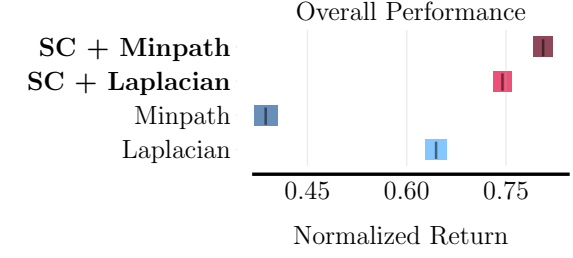


Latent space comparison



Zero-shot RL results



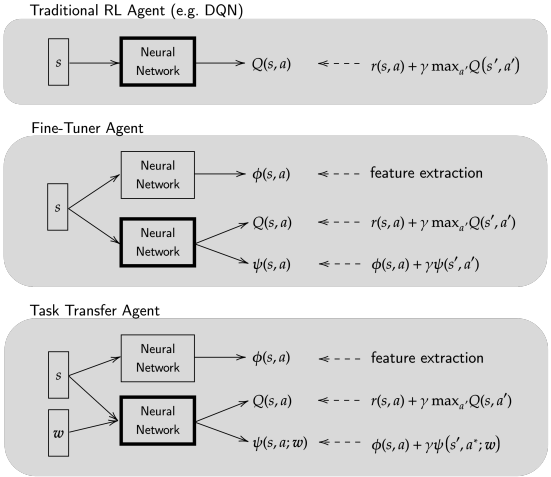


Déroulé

6. Successor Features & Clusters

7. **Future work: concrete propositions**

Between single-task and task-conditioned policies



Proof of concept “fine-tuner” agent