Apprentissage par renforcement

Cours 8: Model-based RL, Imitation

Sylvain Lamprier

UE RLD - Master DAC

2021

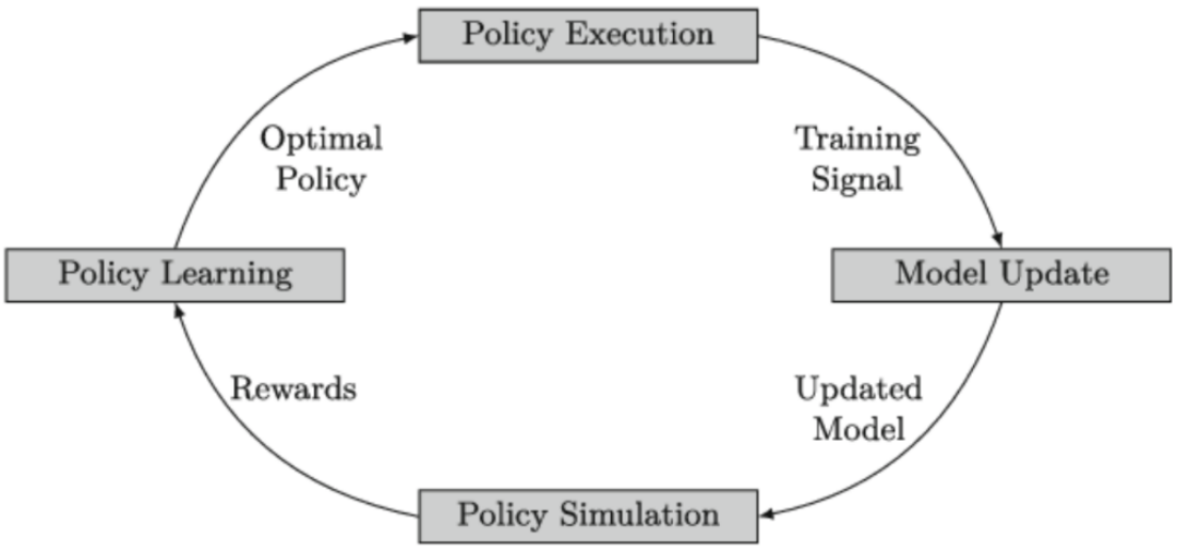
1. Model-based RL
   - Types de modèles
   - Open-loop vs Closed-loop
   - Trois familles d'approches MBRL

2. Imitation Learning
   - Behavior Cloning
   - Hacking
   - Hybridation Imitation Learning / Reinforcement Learning
   - Inverse RL

Model-based RL

Model-based RL consiste en :

- Utiliser des interactions avec l'environnement pour construire un modèle des dynamiques du monde
- Utiliser le modèle du monde pour faciliter l'apprentissage d'une politique efficace (planification)



A flow diagram of model-based RL

Plutôt que de décider d'une politique état $\rightarrow$ action uniquement par l'expérience, on apprend les dynamiques du monde afin de planifier les actions selon les situations.

Model-based RL

Le principal avantage des approche MBRL est l'efficacité ("sample efficiency") :

MBRL a besoin de moins d'interactions avec l'environnement pour établir une politique efficace, si tant est que l'on est capable d'apprendre un bon modèle à partir des observations. Si les dynamiques sont très complexes :

- Sous-apprentissage si modèle trop simpliste : mauvaises prédictions même sur les transitions observées
- Sur-apprentissage si modèle trop complexe et appris sur trop peu de données : faible capacité de généralisation

D'autre part, il est nécessaire de considérer l'aspect stochastique des dynamiques :

- Aleatoric uncertainty : capacité du modèle à émettre des distributions de probabilités qui capturent bien la stochasticité des transitions
- Epsitemic uncertainty : capacité du modèle à évaluer son incertitude sur les futurs possibles

Model-based RL : Types de modèles

Différents types de modèles du monde :

Model	Aleatoric uncertainty	Epistemic uncertainty
Deterministic NN (D)	No	No
Probabilistic NN (P)	Yes	No
Deterministic ensemble NN (DE)	No	Yes
Gaussian process baseline (GP)	Homoscedastic	Yes
Probabilistic ensemble NN (PE)	**Yes**	**Yes**

Table des modèles considérés dans [Chu+18]

Objectifs :

- Modéliser la dynamique $p(s_{t+1}|s_t, a_t)$ selon une fonction $\tilde{f}\, s_t, a_t$ qui retourne le prochain état s_{t+1}
- Modéliser les rewards $r(s, a, s')$ selon une fonction $\tilde{r}(s, a, s')$

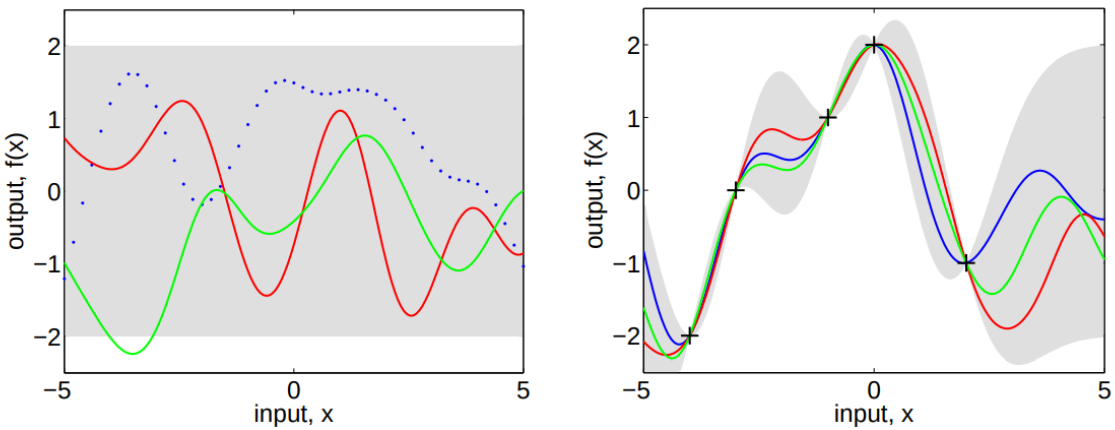
Model-based RL : Types de modèles

Gaussian Process [Ras03] [RW06] :

- Un GP suppose que $p(f(x_1), ..., f(x_N))$ suit une gaussienne multivariée, de moyenne μ et de covariance Σ donnée par $\Sigma_{ij} = k(x_i, x_j)$, où k est un noyau.
- L'idée centrale est de se dire que si x_i et x_j sont similaires au sens du noyau, alors on s'attend à ce que leur valeur associée (selon f) soit également similaire.
- On a alors (avec f les observations et f^* les points non-observés) :

$$\begin{pmatrix} f \\ f_* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu \\ \mu_* \end{pmatrix}, \begin{pmatrix} K & K_* \\ K_*^T & K_{**} \end{pmatrix} \right)$$

$$\Rightarrow \quad f_* | f \sim \mathcal{N}(\mu_* + K_*^T K^{-1}(f - \mu), K_{**} - K_*^T K^{-1} K_*)$$



(a), prior (b), posterior

Figure 2.2: Panel (a) shows three functions drawn at random from a GP prior; the dots indicate values of y actually generated; the two other functions have (less correctly) been drawn as lines by joining a large number of evaluated points. Panel (b) shows three random functions drawn from the posterior, i.e. the prior conditioned on the five noise free observations indicated. In both plots the shaded area represents the pointwise mean plus and minus two times the standard deviation for each input value (corresponding to the 95% confidence region), for the prior and posterior respectively.

Model-based RL : Types de modèles

Possibilité de GP pour RL (explorée dans PILCO [DR11]) :

- Un GP indépendant par dimension des états x
- Points d'entrée $\tilde{x}_t \in \mathbb{R}^{d_x+d_u}$ qui correspondent à la **concaténation de la représentation de l'état x_t et de l'action u_t**.
- On a $f(\tilde{x}_t) = \Delta_t = x_t - x_{t+1} + \epsilon_t$, avec $\epsilon_t \sim \mathcal{N}(0, \Sigma_\epsilon)$, où $\Sigma_\epsilon = diag([\sigma_{\epsilon 1}, ..., \sigma_{\epsilon D}])$
- **Squared Exponential Kernel** : $k(\tilde{x}, \tilde{x}') = \alpha^2 exp(-\frac{1}{2}(\tilde{x} - \tilde{x}')^T \Lambda^{-1}(\tilde{x} - \tilde{x}')$, avec $\Lambda = diag([l_1, ..., l_D])$ et α^2 des hyper-paramètres optimisés sur les observations par maximisation de l'évidence [Ras03]
- $p(x_t^* | x_{t-1}^*, u_{t-1}^*) = \mathcal{N}(x_t^* | \mu_t^*, \Sigma_t^*)$, avec $\mu_t^* = x_{t-1}^* + \mathbb{E}[\Delta_t^*]$ et $\Sigma_t^* = var[\Delta_t^*]$
- $\mathbb{E}[\Delta_t^*] = k(\tilde{X}, \tilde{x}_{t-1}^*)^T (K + \sigma_\epsilon^2 I)^{-1} y$, avec K la matrice de gram sur les observations et $y = [\Delta_1, ..., \Delta_N]$ le vecteur de toutes les observations
- $var[\Delta_t^*] = k(\tilde{x}_{t-1}^*, \tilde{x}_{t-1}^*) - k(\tilde{X}, \tilde{x}_{t-1}^*)^T (K + \sigma_\epsilon^2 I)^{-1} k(\tilde{X}, \tilde{x}_{t-1}^*)$

Autre exemple similaire dans [KR04]

Model-based RL : Types de modèles

- NN déterministe :
  - On cherche les paramètres θ qui minimisent

$$\sum_{s_t, a_t, s_{t+1}} ||s_{t+1} - f_\theta(s_t, a_t)||^2$$

    avec f_θ un réseau de neurones.

- NN probabiliste :
  - On cherche les paramètres θ qui maximisent

$$\sum_{s_t, a_t, s_{t+1}} \log p(s_{t+1} | f_\theta(s_t, a_t))$$

    avec f_θ un réseau de neurones qui émet par exemple les paramètres μ et Σ d'une gaussienne multivariée
  - Possible de considérer des mixtures de gaussiennes en sortie pour éviter le mode-dropping (peut nécessiter inférence variationnelle ou EM pour estimer les paramètres)

$\Rightarrow$ NN plus puissants que GP mais beaucoup plus longs à apprendre et risque d'over-fitting (besoin de beaucoup plus de données)

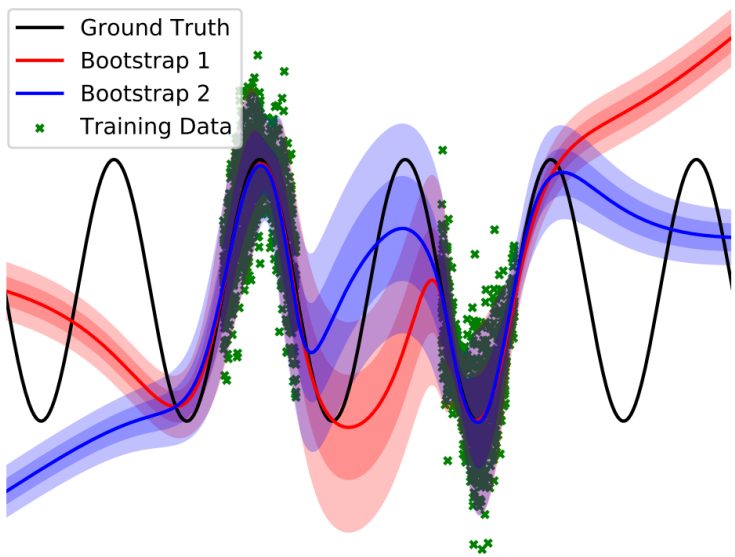
Model-based RL : Types de modèles

Une méthode centrale pour capturer l'incertitude epistémique est de faire de l'inférence bayésienne :

- Déterminer la distribution postérieure de paramètres selon les observations, i.e. les paramètres à chaque couche du NN sont des variables aléatoires avec un prior gaussien par exemple
- Méthodes MCMC ou Inférence variationnelle dans la majorité des cas (voir par exemple [Blu+15])
- Requiert un volume très important de données pour fonctionner

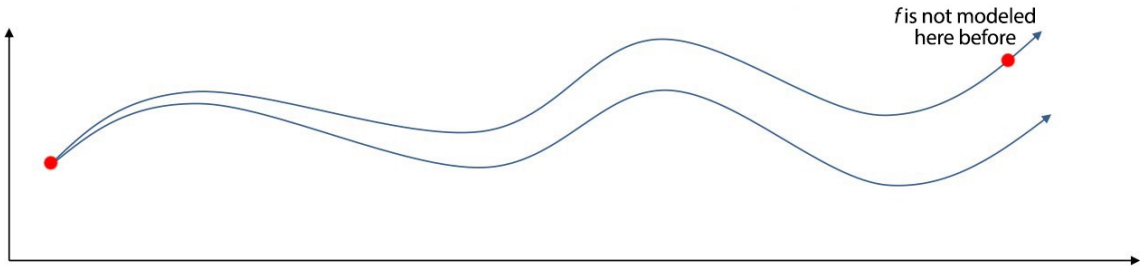
Une alternative est de considérer des ensembles de NN (bagging :Bootstrap AGGregatING) [Kur+18b],[LPB17] :

- Plusieurs NN (deterministes ou probabilistes) appris sur des samplings avec remise de D
- Mixture uniforme de NN pour capturer l'incertitude epistémique
- Distributions globallement d'accord proche des données, beaucoup moins lorsqu'on s'en éloigne (+ d'incertitude)



Model-based RL

- Ok mais comment échantillonner les exemples pour l'apprentissage du modèle ?
  - Si l'espace est grand on risque d'ignorer certaines zones
  - Erreurs s'accumulent au fil du processus :

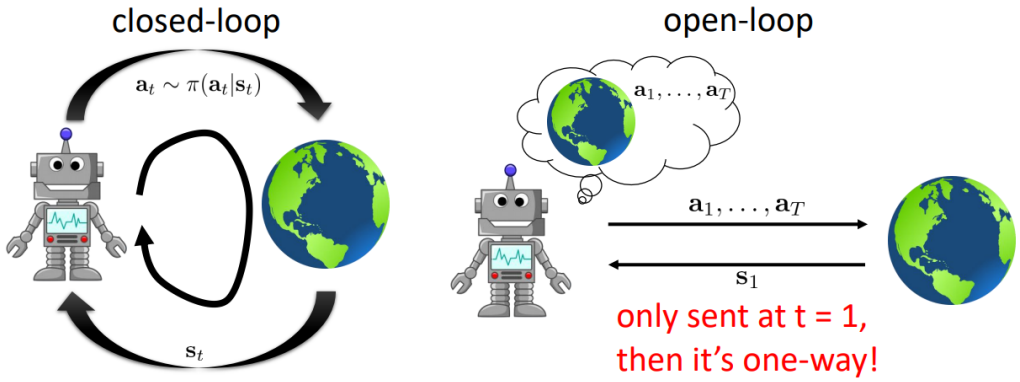


- Possibilité de ré-échantillonner après planning :

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions
4. execute those actions and add the resulting data $\{(\mathbf{x}, \mathbf{u}, \mathbf{x}')_j\}$ to $\mathcal{D}$

- Ok mais besoin de plannifier toutes les actions (open-loop)
  - Difficulté de prédiction d'une séquence complète efficace
  - On risque d'apprendre des dynamiques dans des zones inutiles

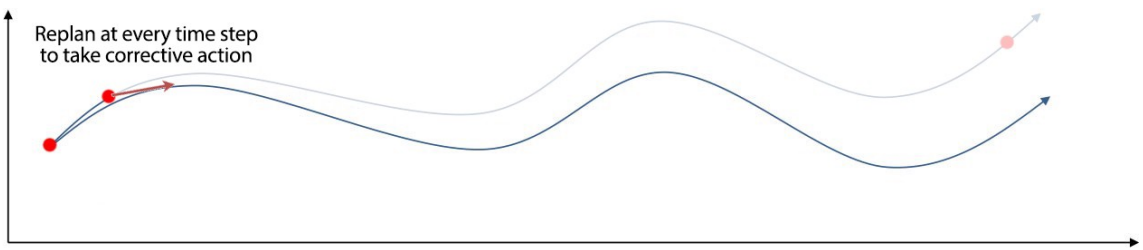
Model-based RL : MPC



$\Rightarrow$ Model Predictive Control (MPC) permet de transformer un problème de plannification open-loop en un problème closed-loop plus simple

every N steps

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions
4. execute the first planned action, observe resulting state $\mathbf{s}'$ (MPC)
5. append $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to dataset $\mathcal{D}$



Model-based RL

Trois grandes familles de MBRL [Wan+19] :

- Approches type Dyna [Sut91]
  - Approches Model-Free sur simulations de l'environnement
- Approches Policy Search via Backpropagation [LK13]
  - Retro-propagation du reward sur la séquence d'états actions
- Approches de Shooting [Chu+18]
  - Sampling aléatoire de séquences et MPC

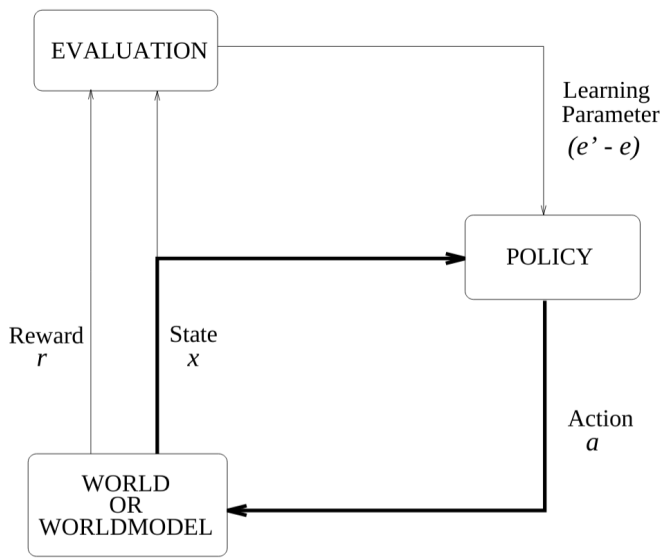
1. Model-based RL
   - Types de modèles
   - Open-loop vs Closed-loop
   - Trois familles d'approches MBRL
     - Approches Dyna
     - Policy Search via Backprop
     - Approches de Shooting

2. Imitation Learning
   - Behavior Cloning
   - Hacking
   - Hybridation Imitation Learning / Reinforcement Learning
   - Inverse RL

Model-based RL : Approches Dyna

Approches Dyna :

- Utilisation des approches Model-Free sur simulations de l'environnement
- 2 étapes d'apprentissage à chaque itération :
  - Apprentissage des dynamiques et des rewards en interragissant avec l'environnement selon la politique courante
  - Mise à jour de la politique en interragissant avec le modèle plutôt qu'avec l'environnement



Model-based RL : Approches Dyna

- Dyna-Q [Sut91] :
  - A chaque étape du Q-Learning
    - Choix et émission de a_t
    - Observation du quadruplet (s_t, a_t, s_{t+1}, r_t)
    - Mise à jour de Q selon le quadruplet observé

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a' \in \mathcal{A}(s_{t+1})} Q(s_{t+1}, a') - Q(s_t, a_t))$$

    - Mise à jour du modèle de MDP

$$\hat{R}(s_t, a_t, s_{t+1}) \leftarrow \hat{R}(s_t, a_t, s_{t+1}) + \alpha_R(r_t - \hat{R}(s_t, a_t, s_{t+1}))$$

$$\hat{P}(s_{t+1}|s_t, a_t) \leftarrow \hat{P}(s_{t+1}|s_t, a_t) + \alpha_R(1 - \hat{P}(s_{t+1}|s_t, a_t))$$

$$\forall s' \neq s_{t+1} : \hat{P}(s'|s_t, a_t) \leftarrow \hat{P}(s'|s_t, a_t) + \alpha_R(0 - \hat{P}(s'|s_t, a_t))$$

    - Puis sample de k couples etat-action (éventuellement selon une stratégie) et pour chaque couple (s^i, a^i) :

$$Q(s^i, a^i) \leftarrow Q(s^i, a^i) + \alpha(\sum_{s' \in \mathcal{S}} \hat{P}(s'|s^i, a^i)(\hat{R}(s^i, a^i, s') + \gamma \max_{a' \in \mathcal{A}(s')} Q(s', a')) - Q(s^i, a^i))$$

Model-based RL : Approches Dyna

Model-Ensemble Trust-Region Policy Optimization (ME-TRPO) [Kur+18a]

- Ensemble de k modèles du monde
- Apprentissage de la politique selon TRPO avec trajectoires samplées de ces différents modèles

$\Rightarrow$ Ici la contrainte de KL de TRPO trouve encore plus d'intérêt qu'en Model-Free car les trajectoires simulées correspondent aux distributions de la politique **et** des modèles de dynamique précédents (éviter de sortir des zones connues)

Algorithm 2 Model Ensemble Trust Region Policy Optimization (ME-TRPO)

Initialize a policy π_θ and all models $\hat{f}_{\phi_1}, \hat{f}_{\phi_2}, ..., \hat{f}_{\phi_K}$.
Initialize an empty dataset $\mathcal{D}$.
repeat
  Collect samples from the real system f using π_θ and add them to D.
  Train all models using $\mathcal{D}$.
  repeat ▷ Optimize π_θ using all models.
    Collect fictitious samples from $\{\hat{f}_{\phi_i}\}_{i=1}^{K}$ using π_θ.
    Update the policy using TRPO on the fictitious samples.
    Estimate the performances $\hat{\eta}(\theta; \phi_i)$ for $i = 1, ..., K$.
  until the performances stop improving.
until the policy performs well in real environment f.

* Model-based RL : Approches Dyna

Stochastic Lower Bound Optimization (SLBO) [Luo+18]

Algorithm 2 Stochastic Lower Bound Optimization (SLBO)

Initialize model network parameters ϕ and policy network parameters θ
Initialize dataset $\mathcal{D} \leftarrow \emptyset$
for n_{outer} iterations **do**
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{$ collect n_{collect} samples from real environment using π_θ with noises $\}$
  for n_{inner} iterations **do** ▷ optimize (6.2) with stochastic alternating updates
    for n_{model} iterations **do**
      optimize (6.1) over ϕ with sampled data from $\mathcal{D}$ by one step of Adam
    for n_{policy} iterations **do**
      $\mathcal{D}' \leftarrow \{$ collect n_{trpo} samples using $\widehat{M}_\phi$ as dynamics $\}$
      optimize π_θ by running TRPO on $\mathcal{D}'$

avec :

$$\mathcal{L}_\phi^{(H)}((s_{t:t+h}, a_{t:t+h}); \phi) = \frac{1}{H}\sum_{i=1}^{H} \|(\hat{s}_{t+i} - \hat{s}_{t+i-1}) - (s_{t+i} - s_{t+i-1})\|_2. \tag{6.1}$$

$$\max_{\phi,\theta} \; V^{\pi_\theta, \text{sg}(\widehat{M}_\phi)} - \lambda \mathop{\mathbb{E}}_{(s_{t:t+h}, a_{t:t+h}) \sim \pi_k, M^\star} \left[\mathcal{L}_\phi^{(H)}((s_{t:t+h}, a_{t:t+h}); \phi)\right] \tag{6.2}$$

$\Rightarrow$ Loss inspirée par analyse théorique d'une borne inférieure (relaxée ensuite) :

$$V^{\pi, M^*} \geq V^{\pi, \widehat{M}} - D_{\pi_{ref}, \delta}(\widehat{M}, \pi), \quad \forall \pi \text{ s.t. } d(\pi, \pi_{\text{ref}}) \leq \delta$$

avec $D_{\pi_{ref}, \delta}(\widehat{M}, \pi)$ une mesure de divergence (corrélée à $\mathcal{L}_\phi^{(H)}$) entre valeurs selon le modèle $\widehat{M}_\phi$ et le vrai monde pour toute politique dans le voisinage de $\pi_{ref} = \pi_k$

$\Rightarrow$ Apprentissage multi-steps de la dynamique (moyenne des différences entre transitions prédites et transitions sur H pas de temps)

* Model-based RL : Approches Dyna

Model-Based Meta-Policy-Optimzation (MB-MPO) [Cla+18]

- Basé sur Meta-Learning MAML [FAL17], considérant K modèles comme K différentes tâches
- Recherche de paramètres θ à partir desquels il est possible de s'adapter aux différents modèles en un seul pas de gradient
  - $\Rightarrow$ permet d'atteindre des politiques à **forte plasticité** structurelle (capacité à changer facilement selon des déplacements des paramètres minimes) dans les zones à **forte incertitude**
- Bénéfices : régularisation, robustesse des modèles, adaptabilité à de nouvelles dynamiques

Algorithm 1 MB-MPO

Require: Inner and outer step size α, β

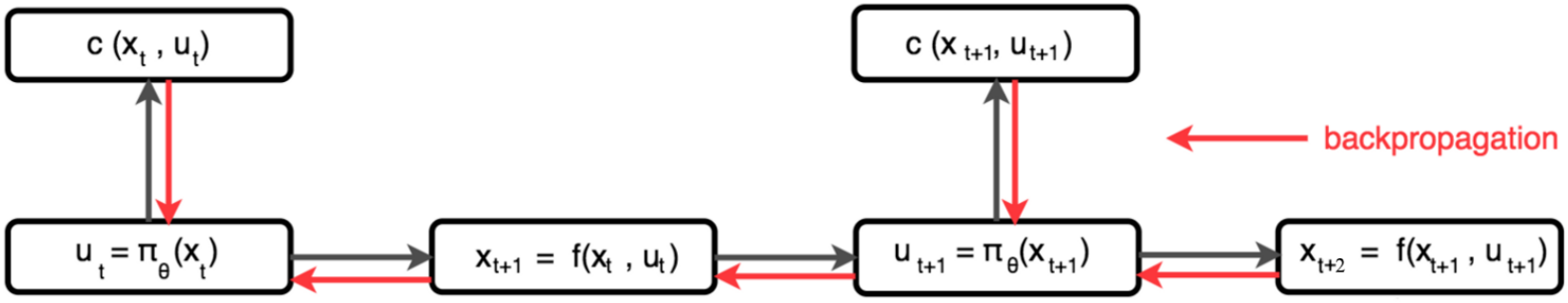
1: Initialize the policy $\pi_{\boldsymbol{\theta}}$, the models $\hat{f}_{\boldsymbol{\phi}_1}, \hat{f}_{\boldsymbol{\phi}_2}, ..., \hat{f}_{\boldsymbol{\phi}_K}$ and $\mathcal{D} \leftarrow \emptyset$
2: **repeat**
3: Sample trajectories from the real environment with the adapted policies $\pi_{\boldsymbol{\theta}'_1}, ..., \pi_{\boldsymbol{\theta}'_K}$. Add them to $\mathcal{D}$.
4: Train all models using $\mathcal{D}$.
5: **for all** models $\hat{f}_{\boldsymbol{\phi}_k}$ **do**
6: Sample imaginary trajectories $\mathcal{T}_k$ from $\hat{f}_{\boldsymbol{\phi}_k}$ using $\pi_{\boldsymbol{\theta}}$
7: Compute adapted parameters $\boldsymbol{\theta}'_k = \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} J_k(\boldsymbol{\theta})$ using trajectories $\mathcal{T}_k$
8: Sample imaginary trajectories $\mathcal{T}'_k$ from $\hat{f}_{\boldsymbol{\phi}_k}$ using the adapted policy $\pi_{\boldsymbol{\theta}'_k}$
9: **end for**
10: Update $\boldsymbol{\theta} \rightarrow \boldsymbol{\theta} - \beta \frac{1}{K} \sum_k \nabla_{\boldsymbol{\theta}} J_k(\boldsymbol{\theta}'_k)$ using the trajectories $\mathcal{T}'_k$
11: **until** the policy performs well in the real environment
12: **return** Optimal pre-update parameters $\boldsymbol{\theta}^*$

1. Model-based RL
   - Types de modèles
   - Open-loop vs Closed-loop
   - Trois familles d'approches MBRL
     - Approches Dyna
     - Policy Search via Backprop
     - Approches de Shooting

2. Imitation Learning
   - Behavior Cloning
   - Hacking
   - Hybridation Imitation Learning / Reinforcement Learning
   - Inverse RL

Model-based RL : Policy Search via Backprop

Et si on propageait le gradient sur toute la séquence connaissant le modèle du monde ?



avec c une fonction de coût et x un état et u une action (notations issues du domaine du contrôle optimal)

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. backpropagate through $f(\mathbf{s}, \mathbf{a})$ into the policy to optimize $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$
4. run $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$, appending the visited tuples $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to $\mathcal{D}$

Model-based RL : Backprop through time

PILCO [DR11] :

- Apprentissage des dynamiques par GP
- Minimisation à chaque étape du coût (distance à une zone cible) :
  $c(\mathbf{x}) = 1 - \exp\left(-\|\mathbf{x} - \mathbf{x}_{\text{target}}\|^2 / \sigma_c^2\right)$
- Backpropagation des gradients selon la chain-rule des posterior-predictive à chaque étape pour trouver la meilleure politique $\pi(x_t)$

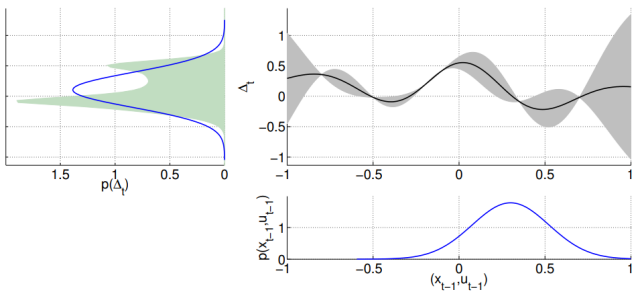


Figure 2. GP prediction at an uncertain input. The input distribution $p(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ is assumed Gaussian (lower right panel). When propagating it through the GP model (upper right panel), we obtain the shaded distribution $p(\Delta_t)$, upper left panel. We approximate $p(\Delta_t)$ by a Gaussian with the exact mean and variance (upper left panel).

Algorithm 1 PILCO

1: **init:** Sample controller parameters $\theta \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
Apply random control signals and record data.
2: **repeat**
3: Learn probabilistic (GP) dynamics model,
4: **repeat**
5: Approximate inference for policy evaluation (get $J^\pi(\theta)$),
6: Update parameters θ (e.g., CG).
7: **until** convergence; **return** θ^*
8: Set $\pi^* \leftarrow \pi(\theta^*)$.
9: Apply π^* to system (single trial/episode) and record data.
10: **until** task learned

Très sample efficient : les données sont utilisées efficacement par rétro-propagation des feedback dans la chaîne complète

Cependant, les états consécutifs d'une trajectoire sont fortement corrélés. Back-propager dans ces conditions conduit souvent à des gradients qui s'effondrent ou qui explosent.

Model-based RL : Contrôle optimal

Autre possibilité : s'appuyer sur la théorie du contrôle optimal

$\Rightarrow$ Linear-Quadratic Regulator : Trouver la séquence de contrôle $\mathbf{u}_1, \ldots, \mathbf{u}_T$ qui minimise :

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T} c\left(\mathbf{x}_1, \mathbf{u}_1\right) + c\left(f\left(\mathbf{x}_1, \mathbf{u}_1\right), \mathbf{u}_2\right) + \cdots + c\left(f(f(\ldots)\ldots), \mathbf{u}_T\right)$$

Selon un modèle de dynamiques linéaire connu pour chaque t :

$$f\left(\mathbf{x}_t, \mathbf{u}_t\right) = \mathbf{F}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \mathbf{f}_t$$

Et un coût quadratique :

$$c\left(\mathbf{x}_t, \mathbf{u}_t\right) = \frac{1}{2} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{C}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{c}_t$$

LQR définit une solution analytique pour ce problème

Model-based RL : Contrôle optimal

LQR définit une solution analytique pour ce problème

Backward recursion

for $t = T$ to 1:

$$\mathbf{Q}_t = \mathbf{C}_t + \mathbf{F}_t^T \mathbf{V}_{t+1} \mathbf{F}_t$$

$$\mathbf{q}_t = \mathbf{c}_t + \mathbf{F}_t^T \mathbf{V}_{t+1} \mathbf{f}_t + \mathbf{F}_t^T \mathbf{v}_{t+1}$$

$$Q(\mathbf{x}_t, \mathbf{u}_t) = \text{const} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{Q}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{q}_t$$

$$\mathbf{u}_t \leftarrow \arg\min_{\mathbf{u}_t} Q(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{K}_t \mathbf{x}_t + \mathbf{k}_t$$

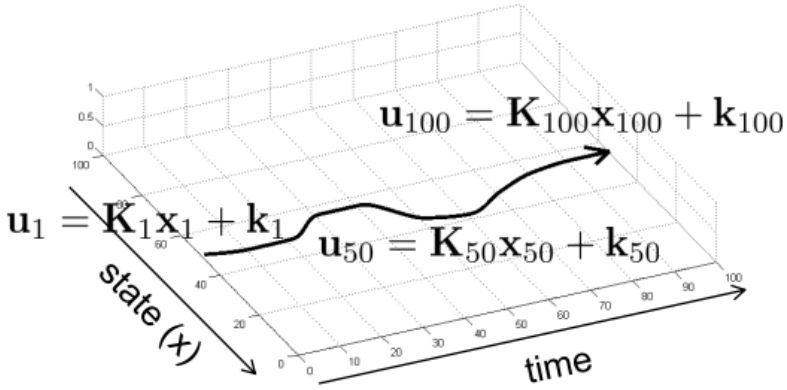
$$\mathbf{K}_t = -\mathbf{Q}_{\mathbf{u}_t, \mathbf{u}_t}^{-1} \mathbf{Q}_{\mathbf{u}_t, \mathbf{x}_t}$$

$$\mathbf{k}_t = -\mathbf{Q}_{\mathbf{u}_t, \mathbf{u}_t}^{-1} \mathbf{q}_{\mathbf{u}_t}$$

$$\mathbf{V}_t = \mathbf{Q}_{\mathbf{x}_t, \mathbf{x}_t} + \mathbf{Q}_{\mathbf{x}_t, \mathbf{u}_t} \mathbf{K}_t + \mathbf{K}_t^T \mathbf{Q}_{\mathbf{u}_t, \mathbf{x}_t} + \mathbf{K}_t^T \mathbf{Q}_{\mathbf{u}_t, \mathbf{u}_t} \mathbf{K}_t$$

$$\mathbf{v}_t = \mathbf{q}_{\mathbf{x}_t} + \mathbf{Q}_{\mathbf{x}_t, \mathbf{u}_t} \mathbf{k}_t + \mathbf{K}_t^T \mathbf{Q}_{\mathbf{u}_t} + \mathbf{K}_t^T \mathbf{Q}_{\mathbf{u}_t, \mathbf{u}_t} \mathbf{k}_t$$

$$V(\mathbf{x}_t) = \text{const} + \frac{1}{2} \mathbf{x}_t^T \mathbf{V}_t \mathbf{x}_t + \mathbf{x}_t^T \mathbf{v}_t$$



we know $\mathbf{x}_1$!

Forward recursion

for $t = 1$ to T:

$$\mathbf{u}_t = \mathbf{K}_t \mathbf{x}_t + \mathbf{k}_t$$

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$$

Avec $Q_{x,u}$ la dérivée seconde de Q selon x et u

Model-based RL : Contrôle optimal

Version stochastique LQG selon dynamique linéaire-gaussienne :

$$p\left(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t\right) = \mathcal{N}\left(\mathbf{F}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \mathbf{f}_t, \Sigma_t\right)$$

☝ Ne modifie pas la solution du contrôle optimal !

$\Rightarrow$ Contrôle toujours selon $\mathbf{u}_t = \mathbf{K}_t \mathbf{x}_t + \mathbf{k}_t$ (on peut ignorer Σ_t dans les calculs car n'intervient pas dans la minimisation de l'espérance).

Par contre :

👎 Comment faire si f pas linéaire ?

👎 Comment faire si c pas quadratique ?

Model-based RL : Contrôle optimal

Une solution est apportée par iLQG (Iterative Linear Quadratic-Gaussian)

Expansions de Taylor de f et c selon une trajectoire nominale $\bar{\tau} = (\bar{x_1}, \bar{u_1}, ..., \bar{x_T})$:

$$f(\mathbf{x}_t, \mathbf{u}_t) \approx f(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t, \mathbf{u}_t} f(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t) \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix}$$

$$c(\mathbf{x}_t, \mathbf{u}_t) \approx c(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t, \mathbf{u}_t} c(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t) \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix}^T \nabla^2_{\mathbf{x}_t, \mathbf{u}_t} c(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t) \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix}$$

avec $\delta x_t = \mathbf{x}_t - \bar{\mathbf{x}}_t$ et $\delta u_t = \mathbf{u}_t - \bar{\mathbf{u}}_t$

Approximation locale linéaire-quadratique d'un système non linéaire

$\Rightarrow$ À chaque itération de l'optimisation : Recherche de la meilleure adaptation du contrôle $\delta u_t^*(\delta x_t)$ à chaque étape de la trajectoire selon des modifications d'états δx_t par rapport à la trajectoire de référence et le coût à minimiser

Model-based RL : Contrôle optimal

Iterative Linear Quadratic-Gaussian (iLQG) [TL05] :

- Programmation dynamique : $V(x,i) = \min_u [c(x,u) + V(f(x,u), i+1)]$, avec f qui prédit l'état à $i+1$ après avoir appliqué u à l'état x.

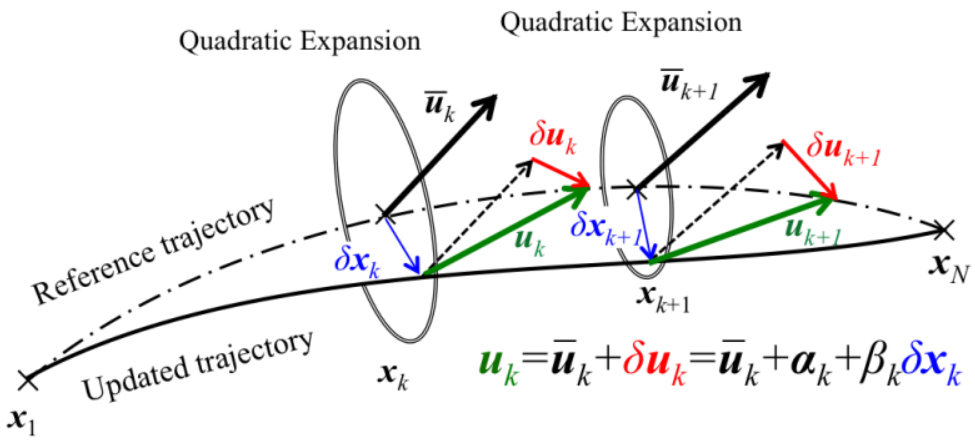
$\Rightarrow$ Optimisation basée sur Differential Dynamic Programming [Jac68] :

$\Rightarrow$ À chaque itération on cherche à améliorer une trajectoire de référence $(\bar{x_1}, \bar{u_1}, ..., \bar{x_N})$

1. Backward pass : Recherche de déplacements locaux $\delta u^*(\delta x)$ qui minimisent pour chaque étape i une approximation quadratique de $c(\bar{x}+\delta x, \bar{u}+\delta u) + V(f(\bar{x}+\delta x, \bar{u}+\delta u), i+1)$ en partant de la fin.
2. Forward pass : Application des corrections $u_i = \bar{u_i} + \delta u_i^*(\delta x)$ à chaque étape i de la trajectoire en partant du début, afin d'ajuster la politique en prenant en compte des déviations légères de la trajectoire.

* Model-based RL : Contrôle optimal

Differential Dynamic Programming [Jac68] [Oza+15]



- À chaque étape on cherche : $\delta u^*(\delta x) = \arg\min_{\delta u} Q(\delta x, \delta u)$, avec $Q(\delta x, \delta u)$ une fonction de perturbations autour de $(\bar{x}, \bar{u})$:

$$Q(\delta \mathbf{x}, \delta \mathbf{u}) = c(\bar{\mathbf{x}} + \delta \mathbf{x}, \bar{\mathbf{u}} + \delta \mathbf{u}, i) - c(\bar{\mathbf{x}}, \bar{\mathbf{u}}, i) + V(\mathbf{f}(\bar{\mathbf{x}} + \delta \mathbf{x}, \bar{\mathbf{u}} + \delta \mathbf{u}), i+1) - V(\mathbf{f}(\bar{\mathbf{x}}, \bar{\mathbf{u}}), i+1)$$

Expansion du second ordre :

$$\approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta \mathbf{x} \\ \delta \mathbf{u} \end{bmatrix}^\top \begin{bmatrix} 0 & Q_\mathbf{x}^\top & Q_\mathbf{u}^\top \\ Q_\mathbf{x} & Q_{\mathbf{xx}} & Q_{\mathbf{xu}} \\ Q_\mathbf{u} & Q_{\mathbf{ux}} & Q_{\mathbf{uu}} \end{bmatrix} \begin{bmatrix} 1 \\ \delta \mathbf{x} \\ \delta \mathbf{u} \end{bmatrix}$$

Avec :

$$Q_\mathbf{x} = c_\mathbf{x} + \mathbf{f}_\mathbf{x}^\top V'_\mathbf{x} \qquad (a)$$

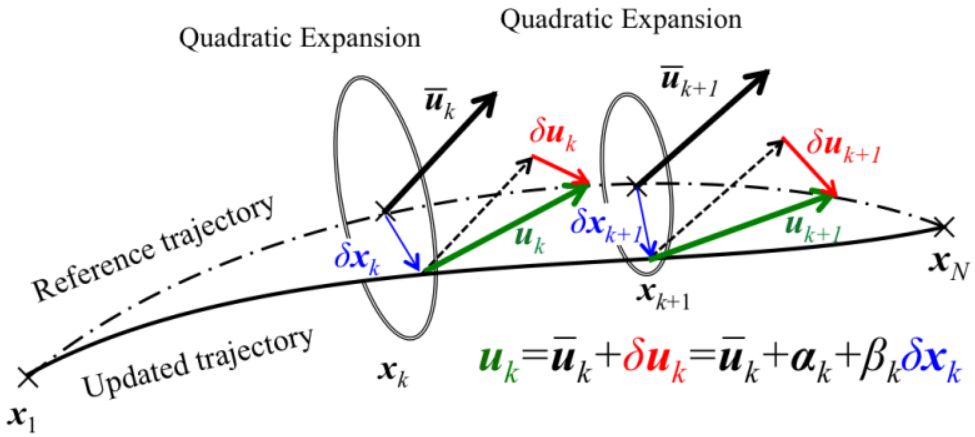
$$Q_\mathbf{u} = c_\mathbf{u} + \mathbf{f}_\mathbf{u}^\top V'_\mathbf{x} \qquad (b) \qquad Q_{\mathbf{xx}} = c_{\mathbf{xx}} + \mathbf{f}_\mathbf{x}^\top V'_{\mathbf{xx}} \mathbf{f}_\mathbf{x} + V'_\mathbf{x} \cdot \mathbf{f}_{\mathbf{xx}} \qquad (c)$$

$$Q_{\mathbf{uu}} = c_{\mathbf{uu}} + \mathbf{f}_\mathbf{u}^\top V'_{\mathbf{xx}} \mathbf{f}_\mathbf{u} + V'_\mathbf{x} \cdot \mathbf{f}_{\mathbf{uu}} \qquad (d) \qquad Q_{\mathbf{ux}} = c_{\mathbf{ux}} + \mathbf{f}_\mathbf{u}^\top V'_{\mathbf{xx}} \mathbf{f}_\mathbf{x} + V'_\mathbf{x} \cdot \mathbf{f}_{\mathbf{ux}} \qquad (e)$$

(avec V' un raccourci pour V à l'étape $i+1$ et V_x la dérivée de V par rapport à x)

* Model-based RL : Contrôle optimal

Differential Dynamic Programming [Jac68] [Oza+15]



- À chaque étape on cherche : $\delta u^*(\delta x) = \arg\min_{\delta u} Q(\delta x, \delta u)$, avec $Q(\delta x, \delta u)$ une fonction de perturbations autour de $(\bar{x}, \bar{u})$:

$$Q(\delta \mathbf{x}, \delta \mathbf{u}) = c(\bar{\mathbf{x}} + \delta \mathbf{x}, \bar{\mathbf{u}} + \delta \mathbf{u}, i) - c(\bar{\mathbf{x}}, \bar{\mathbf{u}}, i) + V(\mathbf{f}(\bar{\mathbf{x}} + \delta \mathbf{x}, \bar{\mathbf{u}} + \delta \mathbf{u}), i+1) - V(\mathbf{f}(\bar{\mathbf{x}}, \bar{\mathbf{u}}), i+1)$$

Expansion du second ordre :

$$\approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta \mathbf{x} \\ \delta \mathbf{u} \end{bmatrix}^\top \begin{bmatrix} 0 & Q_{\mathbf{x}}^\top & Q_{\mathbf{u}}^\top \\ Q_{\mathbf{x}} & Q_{\mathbf{xx}} & Q_{\mathbf{xu}} \\ Q_{\mathbf{u}} & Q_{\mathbf{ux}} & Q_{\mathbf{uu}} \end{bmatrix} \begin{bmatrix} 1 \\ \delta \mathbf{x} \\ \delta \mathbf{u} \end{bmatrix}$$

$\Rightarrow$ En annulant la dérivée de Q par rapport au déplacement δu, on obtient :

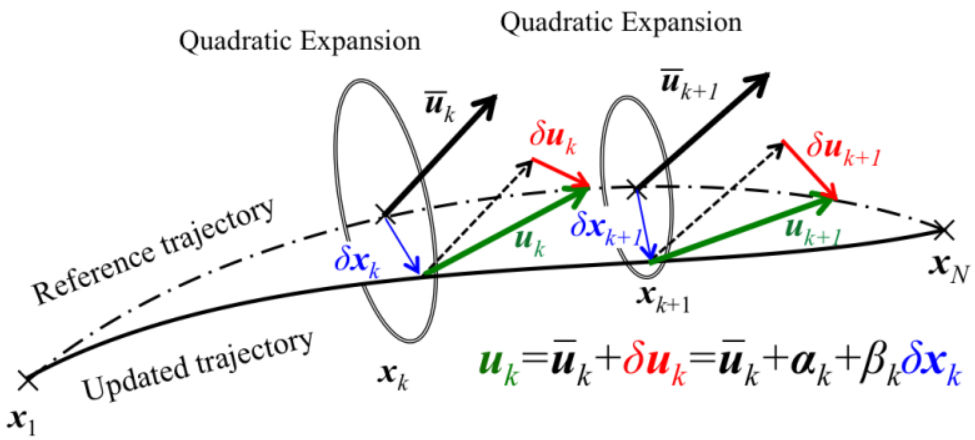
$$\delta u^*(\delta x) = \alpha + \beta \delta x, \text{ avec : } \alpha = -Q_{uu}^{-1} Q_u \text{ et } \beta = -Q_{uu}^{-1} Q_{xu}^\top$$

$\Rightarrow$ En pluggant ce résultat dans l'expansion, on a : $Q(\delta x) \approx \Delta V + V_x \delta x + \frac{1}{2} \delta_x^\top V_{xx} \delta_x$
avec : $\Delta V = -\frac{1}{2} Q_u Q_{uu}^{-1} Q_u$, $V_x = Q_x - Q_u Q_{uu}^{-1} Q_{ux}$ et $V_{xx} = Q_{xx} - Q_{xu} Q_{uu}^{-1} Q_{ux}$

$\Rightarrow$ Ce qui permet de calculer l'expansion quadratique en $i-1$ (backward pass)

* Model-based RL : Contrôle optimal

Differential Dynamic Programming [Jac68] [Oza+15]



- À chaque étape on cherche : $\delta u^*(\delta x) = \arg\min_{\delta u} Q(\delta x, \delta u)$, avec $Q(\delta x, \delta u)$ une fonction de perturbations autour de $(\bar{x}, \bar{u})$:

$$Q(\delta \mathbf{x}, \delta \mathbf{u}) = c(\bar{\mathbf{x}} + \delta \mathbf{x}, \bar{\mathbf{u}} + \delta \mathbf{u}, i) - c(\bar{\mathbf{x}}, \bar{\mathbf{u}}, i) + V(\mathbf{f}(\bar{\mathbf{x}} + \delta \mathbf{x}, \bar{\mathbf{u}} + \delta \mathbf{u}), i+1) - V(\mathbf{f}(\bar{\mathbf{x}}, \bar{\mathbf{u}}), i+1)$$

Expansion du second ordre :

$$\approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta \mathbf{x} \\ \delta \mathbf{u} \end{bmatrix}^\top \begin{bmatrix} 0 & Q_{\mathbf{x}}^\top & Q_{\mathbf{u}}^\top \\ Q_{\mathbf{x}} & Q_{\mathbf{xx}} & Q_{\mathbf{xu}} \\ Q_{\mathbf{u}} & Q_{\mathbf{ux}} & Q_{\mathbf{uu}} \end{bmatrix} \begin{bmatrix} 1 \\ \delta \mathbf{x} \\ \delta \mathbf{u} \end{bmatrix}$$

$\Rightarrow$ À partir de $i = 1$, on applique alors (forward pass) :

$$u_i \leftarrow \bar{u}_i + \alpha_i + \beta_i (x_i - \bar{x}_i)$$
$$x_{i+1} \leftarrow f(x_i, u_i)$$

$\Rightarrow$ Nouvelle trajectoire à partir de laquelle on peut ré-appliquer la passe de backward et ainsi de suite jusqu'à convergence

Model-based RL : Contrôle optimal

Finalement ILQG :

- **Contrôle local** basé sur DDP, en supprimant les dérivées secondes selon f (ajout de beaucoup de complexité pour peu de bénéfice)

Pour résumer iLQG (optimisation identique à iLQR ci dessous) :

until convergence:

$$\mathbf{F}_t = \nabla_{\mathbf{x}_t, \mathbf{u}_t} f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)$$

$$\mathbf{c}_t = \nabla_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)$$

$$\mathbf{C}_t = \nabla^2_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)$$

Run LQR backward pass on state $\delta\mathbf{x}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$ and action $\delta\mathbf{u}_t = \mathbf{u}_t - \hat{\mathbf{u}}_t$

Run forward pass with real nonlinear dynamics and $\mathbf{u}_t = \mathbf{K}_t(\mathbf{x}_t - \hat{\mathbf{x}}_t) + \mathbf{k}_t + \hat{\mathbf{u}}_t$

Update $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{u}}_t$ based on states and actions in forward pass

$\Rightarrow$ Proche d'une méthode de Newton (DDP = méthode de Newton pour le contrôle optimal)

$\Rightarrow$ Ajout d'une line search d'un pas α à chaque itération afin de rester suffisamment proche de la trajectoire $\bar{\tau}$: $u_t = \bar{u}_t + \alpha k_t + K_t(\delta x_t)$

- [TET12] propose une version avec MPC pour optimiser à chaque étape selon les états fournis par le système

Model-based RL : Contrôleurs locaux vs globaux

Contrôleurs locaux (tels que DDP, iLQR, iLQG) :

- 👍 Peuvent être très efficaces pour des tâches de contrôle précises avec des points de départ et des buts fixes (e.g. déplacer un bras de robot d'un point A à un point B)
- 👎 Mais on ne peut pas espérer qu'ils généralisent à des situations nouvelles (mêmes variant assez légèrement des cadres appris)
- 👎 De plus, ils sont limités à des modèles de dynamique et de coût continus et dérivables partout (du moins dans leur version classique)

Contrôleurs globaux (tels que des modèles neuronaux) :

- 👍 Possèdent une bonne capacité de généralisation et ne nécessitent pas de re-planifications permanentes
- 👎 Mais souffrent du manque de données

Model-based RL : Guided Policy Search

Guided Policy Search (GPS) [LK13] :

- Objectif : Allier les avantages des deux types de méthodes
- Méthodes locales = Professeurs qui montrent des décisions sur des problèmes simplifiés (points de départ et objectifs fixes)
- Méthodes globales = Élèves qui cherchent mimer le comportement des locales tout en ayant une meilleure capacité de généralisation

1. Construction de contrôleurs stochastiques locaux à partir des solutions de DDP/iLQG :

$$u_t \sim \mathcal{N}(\bar{u}_t + k_t + K_t(x_t - \bar{x}_t), Q^{-1}_{u_t,u_t})$$

   $\Rightarrow$ [LK13] montre que cette distribution $q(\tau)$ minimise la divergence KL avec une distribution de maximum d'entropie $p(\tau) \propto exp(-c(\tau))$ selon le coût c (sampling dans les zones de faible coût autour de la trajectoire $\bar{\tau}$)

2. Apprentissage d'une politique globale π_θ qui se sert de ces distributions comme contrôleurs off-policy via importance sampling pondéré :

$$\Phi(\theta) = \sum_{t=1}^{T} \left[\frac{1}{Z_t(\theta)} \sum_{i=1}^{m} \frac{\pi_\theta\left(\tau_{i,1:t}\right)}{q\left(\tau_{i,1:t}\right)} r\left(\mathbf{x}_t^i, \mathbf{u}_t^i\right) + w_r \log Z_t(\theta) \right] \text{ avec } Z_t(\theta) = \frac{\pi_\theta\left(\tau_{i,1:t}\right)}{q\left(\tau_{i,1:t}\right)}$$

   $\Rightarrow$ [LK13] introduit le terme $w_r \log Z_t(\theta)$ qui vise à contraindre π_θ à rester proches des distributions guides $q\left(\tau_{i,1:t}\right)$, avec w_r un poids adaptatif

Model-based RL : Guided Policy Search

Algorithm 1 Guided Policy Search

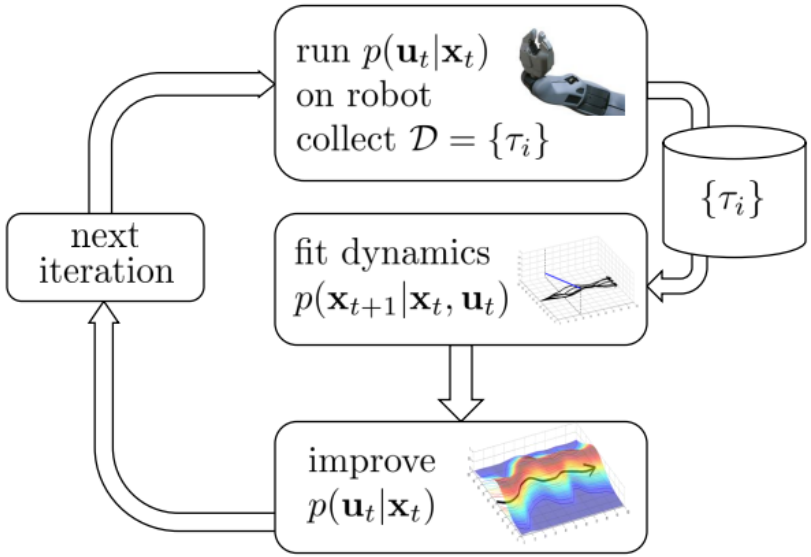
Generate DDP solutions $\pi_{\mathcal{G}_1}, \ldots, \pi_{\mathcal{G}_n}$
Sample $\zeta_1, \ldots, \zeta_m$ from $q(\zeta) = \frac{1}{n} \sum_i \pi_{\mathcal{G}_i}(\zeta)$
Initialize $\theta^\star \leftarrow \arg\max_\theta \sum_i \log \pi_{\theta^\star}(\zeta_i)$
Build initial sample set $\mathcal{S}$ from $\pi_{\mathcal{G}_1}, \ldots, \pi_{\mathcal{G}_n}, \pi_{\theta^\star}$
for iteration $k = 1$ to K **do**
    Choose current sample set $\mathcal{S}_k \subset \mathcal{S}$
    Optimize $\theta_k \leftarrow \arg\max_\theta \Phi_{\mathcal{S}_k}(\theta)$
    Append samples from π_{θ_k} to $\mathcal{S}_k$ and $\mathcal{S}$
    Optionally generate adaptive guiding samples
    Estimate the values of π_{θ_k} and $\pi_{\theta^\star}$ using $\mathcal{S}_k$
    if π_{θ_k} is better than $\pi_{\theta^\star}$ **then**
        Set $\theta^\star \leftarrow \theta_k$
        Decrease w_r
    else
        Increase w_r
        Optionally, resample from $\pi_{\theta^\star}$
    end if
end for
Return the best policy $\pi_{\theta^\star}$

$\Rightarrow$ De manière optionnelle : adapte les samples guides selon π_θ en considérant des contrôleurs DDP appris avec le coût $\bar{c}(x_t, u_t) = c(x_t, u_t) - \log \pi_\theta(u_t|x_t)$.

Model-based RL : GPS avec dynamiques inconnues

Mais si on ne connaît pas le modèle f ?

- Interactions avec l'environnement réel pour apprendre f (modèle linéaire, GP ou GMM) en jouant des samples de la prochaine action plannifiée u_t



$\Rightarrow$ Généralement modèle linéaire gaussien local permettant d'approcher $\frac{df}{d\mathbf{x}_t}$ et $\frac{df}{d\mathbf{u}_t}$ dont on a besoin dans iLQG :

$$p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}(f(\mathbf{x}_t, \mathbf{u}_t), \Sigma)$$
$$f(\mathbf{x}_t, \mathbf{u}_t) \approx \mathbf{A}_t \mathbf{x}_t + \mathbf{B}_t \mathbf{u}_t$$
$$\mathbf{A}_t = \frac{df}{d\mathbf{x}_t} \quad \mathbf{B}_t = \frac{df}{d\mathbf{u}_t}$$

$\Rightarrow$ Possibilité d'utiliser un modèle global appris (NN, GP, GMM) comme prior

Model-based RL : GPS avec dynamiques inconnues

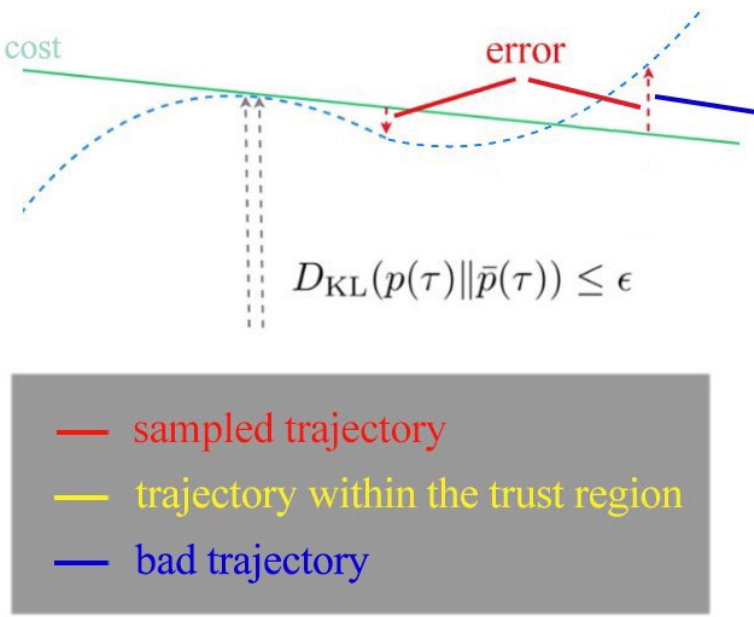
Lorsque l'on apprend des modèles locaux à partir de données, un risque important lorsque l'on emploie iLQG est :

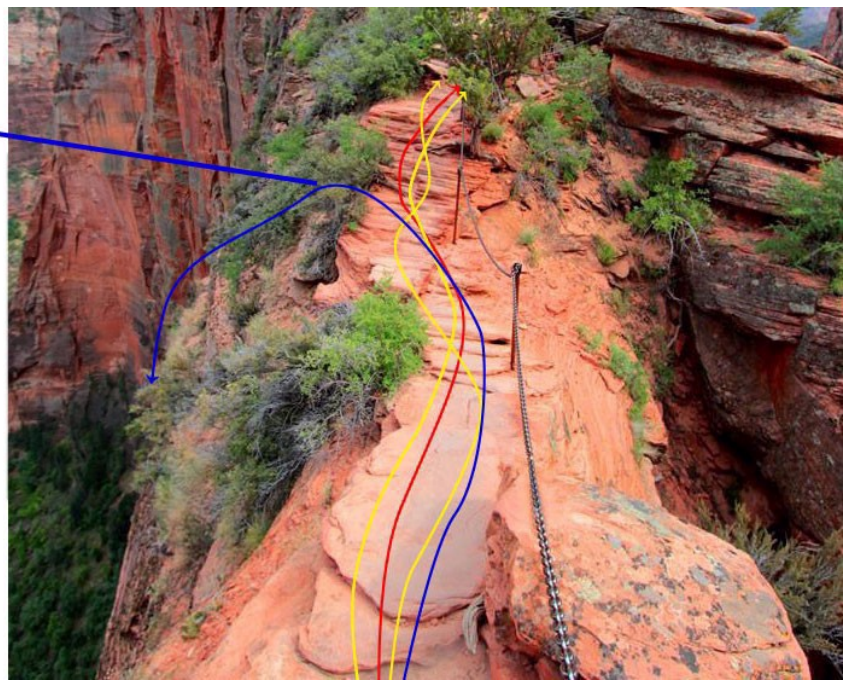
- de modifier trop fortement la politique u

$\Rightarrow$ ce qui nous amènerait à des zones où le modèle local ne s'applique plus

On peut alors considérer l'objectif suivant :

$$\min_p \sum_{t=1}^{T} \mathbb{E}_{p(x_t,u_t)} c(x_t, u_t) \text{ s.t. } D_{KL}(p(\tau)||\bar{p}(\tau)) \leq \epsilon$$





Model-based RL : GPS avec dynamiques inconnues

On peut alors considérer l'objectif suivant :

$$\min_p \sum_{t=1}^{T} \mathbb{E}_{p(x_t,u_t)} c(x_t, u_t) \text{ s.t. } D_{KL}(p(\tau)||\bar{p}(\tau)) \leq \epsilon$$

Une méthode pour optimiser cet objectif sous contrainte est de considérer une descente de gradient duale, impliquant un coefficient de Lagrange λ :

$$\max_\lambda \min_p \mathcal{L}(p, \lambda) = \sum_{t=1}^{T} \mathbb{E}_{p(x_t,u_t)} c(x_t, u_t) + \lambda(D_{KL}(p(\tau)||\bar{p}(\tau)) - \epsilon)$$

Descente de gradient duale

= Méthode de Lagrange pour les cas où l'on ne peut pas exprimer la solution analytique de la minimisation selon x dans le problème de maximisation.

Problème $\max_\lambda \min_x L(x, \lambda) = f(x) + \lambda C(x)$:

1. Choix d'un λ initial
2. Résoudre $x^* \leftarrow \arg\min_x L(x, \lambda)$
3. Calcul de $\frac{\partial g}{\partial \lambda} = \frac{\partial \min_x L(x,\lambda)}{\partial \lambda}$
4. $\lambda \leftarrow \lambda + \alpha \frac{\partial g}{\partial \lambda}$
5. Retour étape 2.

Model-based RL : GPS avec dynamiques inconnues

$\Rightarrow$ Pour un λ fixé, notre problème de minimisation revient à :

$$
\begin{aligned}
p^* &= \arg\min_p \mathcal{L}(p, \lambda) \\
&= \arg\min_p \sum_{t=1}^{T} \mathbb{E}_{p(x_t,u_t)}[c(x_t, u_t) - \lambda \log \bar{p}(u_t|x_t) - \lambda H(p(u_t|x_t)] - \lambda\epsilon \\
&= \arg\min_p \sum_{t=1}^{T} \mathbb{E}_{p(x_t,u_t)}[\frac{1}{\lambda} c(x_t, u_t) - \log \bar{p}(u_t|x_t) - H(p(u_t|x_t)] - \epsilon
\end{aligned}
$$

Étant donné que $\bar{p}(u_t|x_t)$ ne dépend pas du contrôleur que l'on optimise :

- On peut considérer $-\log \bar{p}(u_t|x_t)$ comme un coût supplémentaire
- On a alors : $\bar{c}(t) = \frac{1}{\lambda} c(x_t, u_t) - \log \bar{p}(u_t|x_t)$

$\Rightarrow$ C'est justement ce que les méthodes type iLQG optimisent lorsqu'on utilise $p(u_t|x_t) = \mathcal{N}(\bar{u}_t + \alpha_t + \beta_t(x_t - \bar{x}_t), Q^{-1}_{u_t,u_t})$ avec ce coût modifié [LA14] ! On peut montrer par ailleurs que dans ce cas on obtient une distribution $p(\tau) \propto exp(-\frac{1}{\lambda} c(\tau) + \log \bar{p}(\tau))$.

$\Rightarrow$ Pour l'étape de maximisation selon λ, on prend :

$$\lambda \leftarrow \lambda + \alpha(D_{KL}(p(\tau)||\bar{p}(\tau)) - \epsilon)$$

Si la contrainte de KL est violée, cela mène à accroître λ, ce qui tend à être plus conservatif pour les prochaines étapes.

Model-based RL : GPS avec dynamiques inconnues

GPS avec dynamiques inconnues [LA14] :

Algorithm 1 Guided policy search with unknown dynamics

1: **for** iteration $k = 1$ to K **do**
2: Generate samples $\{\tau_i^j\}$ from each linear-Gaussian controller $p_i(\tau)$ by performing rollouts
3: Fit the dynamics $p_i(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ to the samples $\{\tau_i^j\}$
4: Minimize $\sum_{i,t} \lambda_{i,t} D_{\mathrm{KL}}(p_i(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t) \| p_i(\mathbf{x}_t, \mathbf{u}_t))$ with respect to θ using samples $\{\tau_i^j\}$
5: Update $p_i(\mathbf{u}_t|\mathbf{x}_t)$ using the algorithm in Section 3 and the supplementary appendix
6: Increment dual variables $\lambda_{i,t}$ by $\alpha D_{\mathrm{KL}}(p_i(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t) \| p_i(\mathbf{x}_t, \mathbf{u}_t))$
7: **end for**
8: **return** optimized policy parameters θ

* Model-based RL : GPS avec dynamiques inconnues

Variante de Guided Policy Search (GPS) via mirror descent dans [ML16] :

Algorithm 1 Generic guided policy search method

1: **for** iteration $k \in \{1, \ldots, K\}$ **do**
2: C-step: improve each $p_i(\mathbf{u}_t|\mathbf{x}_t)$ based on surrogate cost $\tilde{\ell}_i(\mathbf{x}_t, \mathbf{u}_t)$, return samples $\mathcal{D}_i$
3: S-step: train $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$ with supervised learning on the dataset $\mathcal{D} = \cup_i \mathcal{D}_i$
4: Modify $\tilde{\ell}_i(\mathbf{x}_t, \mathbf{u}_t)$ to enforce agreement between $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$ and each $p(\mathbf{u}_t|\mathbf{x}_t)$
5: **end for**

Algorithm 3 Mirror descent guided policy search (MDGPS): unknown nonlinear dynamics

1: **for** iteration $k \in \{1, \ldots, K\}$ **do**
2: Generate samples $\mathcal{D}_i = \{\tau_{i,j}\}$ by running either p_i or $\pi_{\theta i}$
3: Fit linear-Gaussian dynamics $p_i(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ using samples in $\mathcal{D}_i$
4: Fit linearized global policy $\bar{\pi}_{\theta i}(\mathbf{u}_t|\mathbf{x}_t)$ using samples in $\mathcal{D}_i$
5: C-step: $p_i \leftarrow \arg\min_{p_i} E_{p_i(\tau)}[\sum_{t=1}^{T} \ell(\mathbf{x}_t, \mathbf{u}_t)]$ such that $D_{\mathrm{KL}}(p_i(\tau) \| \bar{\pi}_{\theta i}(\tau)) \leq \epsilon$
6: S-step: $\pi_\theta \leftarrow \arg\min_\theta \sum_{t,i,j} D_{\mathrm{KL}}(\pi_\theta(\mathbf{u}_t|\mathbf{x}_{t,i,j}) \| p_i(\mathbf{u}_t|\mathbf{x}_{t,i,j}))$ (via supervised learning)
7: Adjust ϵ (see Section 4.2)
8: **end for**

$\Rightarrow$ Ici on cherche à satisfaire des contraintes de proximité à chaque itération plutôt qu'à convergence (meilleures garanties)

Model-based RL : SVG

Stochastic Value Gradients (SVG) [Hee+15] :

$\Rightarrow$ Plutôt que de back-propager sur des trajectoires entières, utilise une critique $V_\theta(s)$ dérivable selon l'entrée

- Soit la politique $\pi(s, \eta; \theta)$ retournant une action continue $a = \pi(s; \theta) + \eta$ avec $\eta \sim \rho(\eta)$
- Soit le modèle de dynamiques $f(s, a, \xi)$ retournant l'état continu $s' = f(s, a) + \xi$ avec $\epsilon \sim \rho(\xi)$
- Optimise :

$$V(\mathbf{s}) = \mathbb{E}_{\rho(\eta)} \left[r(\mathbf{s}, \pi(\mathbf{s}, \eta; \theta)) + \gamma \mathbb{E}_{\rho(\xi)} \left[V'(f(\mathbf{s}, \pi(\mathbf{s}, \eta; \theta), \xi)) \right] \right]$$

- Pour toute transition (s, a, s'), avec $\hat{f}$ le modèle de dynamiques, on a les gradients (où les sub-scripts indiquent "dérivé par") :

  Gradient par rapport à s :

$$V_{\mathrm{s}} = \mathbb{E}_{p(\mathbf{a}|\mathbf{s})} \mathbb{E}_{p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \mathbb{E}_{p(\eta,\xi|\mathbf{s},\mathbf{a},\mathbf{s}')} \left[r_{\mathbf{s}} + r_{\mathbf{a}} \pi_s + \gamma V'_{\mathbf{s}'} \left(\hat{\mathbf{f}}_{\mathbf{s}} + \hat{\mathbf{f}}_{\mathbf{a}} \pi_{\mathbf{s}} \right) \right]$$

  Gradient par rapport à θ :

$$V_\theta = \mathbb{E}_{p(\mathbf{a}|\mathbf{s})} \mathbb{E}_{p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \mathbb{E}_{p(\eta,\xi|\mathbf{s},\mathbf{a},\mathbf{s}')} \left[r_{\mathbf{a}} \pi_\theta + \gamma \left(V'_{\mathbf{s}} \hat{\mathbf{f}}_{\mathbf{a}} \pi_\theta + V'_\theta \right) \right]$$

Model-based RL : SVG

Stochastic Value Gradients (SVG) [Hee+15] :

Algorithm 2 SVG(1) with Replay

Given empty experience database $\mathcal{D}$
for $t = 0$ **to** ∞ **do**
  Apply control $\pi(\mathbf{s}, \eta; \theta)$, $\eta \sim \rho(\eta)$
  Observe $r, \mathbf{s}'$
  Insert $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')$ into $\mathcal{D}$
  // Model and critic updates
  Train generative model $\hat{\mathbf{f}}$ using $\mathcal{D}$
  Train value function $\hat{V}$ using $\mathcal{D}$ (Alg. 4)
  // Policy update
  Sample $(\mathbf{s}^k, \mathbf{a}^k, r^k, \mathbf{s}^{k+1})$ from $\mathcal{D}$ $(k \leq t)$
  $w = \frac{p(\mathbf{a}^k|\mathbf{s}^k;\theta^t)}{p(\mathbf{a}^k|\mathbf{s}^k;\theta^k)}$
  Infer $\xi^k|(\mathbf{s}^k, \mathbf{a}^k, \mathbf{s}^{k+1})$ and $\eta^k|(\mathbf{s}^k, \mathbf{a}^k)$
  $v_\theta = w(r_\mathbf{a} + \gamma \hat{V}'_{\mathbf{s}'} \hat{\mathbf{f}}_\mathbf{a}) \pi_\theta \big|_{\eta^k, \xi^k}$
  Apply gradient-based update using v_θ
end for

$\Rightarrow$ Inférence de η et ξ selon $p(\eta, \xi|s, a, s')$ (via re-parametrization) uniquement pour les cas de modèles plus complexes que ceux avec bruits additifs définis plus haut

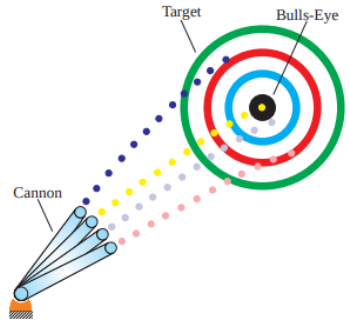
1. Model-based RL
   - Types de modèles
   - Open-loop vs Closed-loop
   - Trois familles d'approches MBRL
     - Approches Dyna
     - Policy Search via Backprop
     - Approches de Shooting

2. Imitation Learning
   - Behavior Cloning
   - Hacking
   - Hybridation Imitation Learning / Reinforcement Learning
   - Inverse RL

Model-based RL : Shooting Algorithms

Méthodes de Shooting [Rao09]

- Modèles centrés sur MPC
- Lorsque les dynamiques sont complexes et/ou les rewards non-convexes, méthodes backprop through time difficiles à mettre en place



- Plutôt que d'optimiser directement $\arg\max_{\mathbf{a}_t,\ldots,\mathbf{a}_{t+H-1}} \sum_{t'=t}^{t+H-1} \gamma^{t'-t} r\left(\mathbf{s}_{t'}, \mathbf{a}_{t'}\right)$:
  - $\Rightarrow$ On sample des séquences aléatoirement (en respectant le modèle de dynamique)
  - $\Rightarrow$ Puis on prend la première action de la meilleure des séquences échantillonnées

Model-based RL : Shooting Algorithms

Model-Based Mode-Free (MB-MF) [Nag+17] : Initialisation de méthodes Model-Free avec des politiques apprises en Model-Based

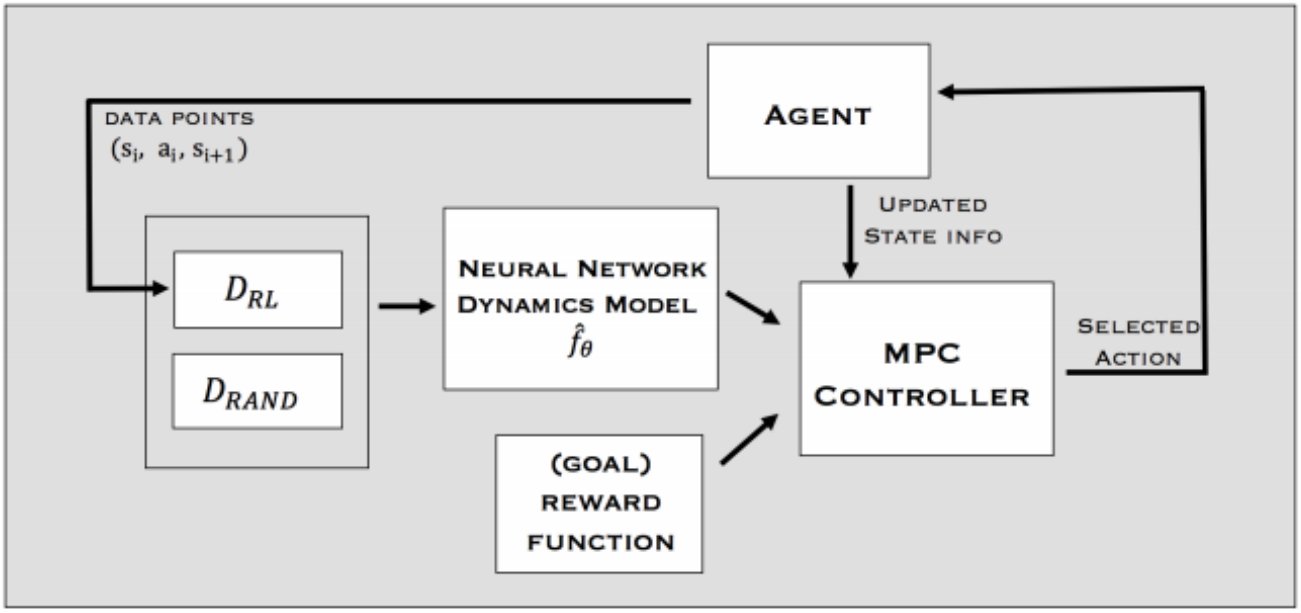


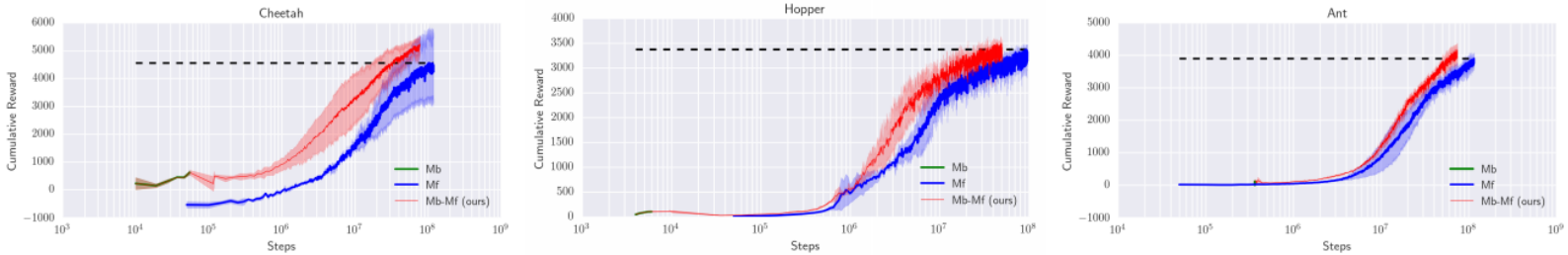
Fig. 2: Illustration of Algorithm 1. On the first iteration, random actions are performed and used to initialize $\mathcal{D}_{\text{RAND}}$. On all following iterations, this iterative procedure is used to train the dynamics model, run the MPC controller for action selection, aggregate data, and retrain the model.

Model-based RL : Shooting Algorithms

Algorithm 1 Model-based Reinforcement Learning

1: gather dataset $\mathcal{D}_{\text{RAND}}$ of random trajectories
2: initialize empty dataset $\mathcal{D}_{\text{RL}}$, and randomly initialize $\hat{f}_\theta$
3: **for** iter=1 **to** max_iter **do**
4: train $\hat{f}_\theta(\mathbf{s}, \mathbf{a})$ by performing gradient descent, using $\mathcal{D}_{\text{RAND}}$ and $\mathcal{D}_{\text{RL}}$
5: **for** $t = 1$ **to** T **do**
6: get agent's current state $\mathbf{s}_t$
7: use $\hat{f}_\theta$ to estimate optimal action sequence $\mathbf{A}_t^{(H)}$
8: execute first action $\mathbf{a}_t$ from selected action sequence $\mathbf{A}_t^{(H)}$
9: add $(\mathbf{s}_t, \mathbf{a}_t)$ to $\mathcal{D}_{\text{RL}}$
10: **end for**
11: **end for**

Random Shooting de taille H

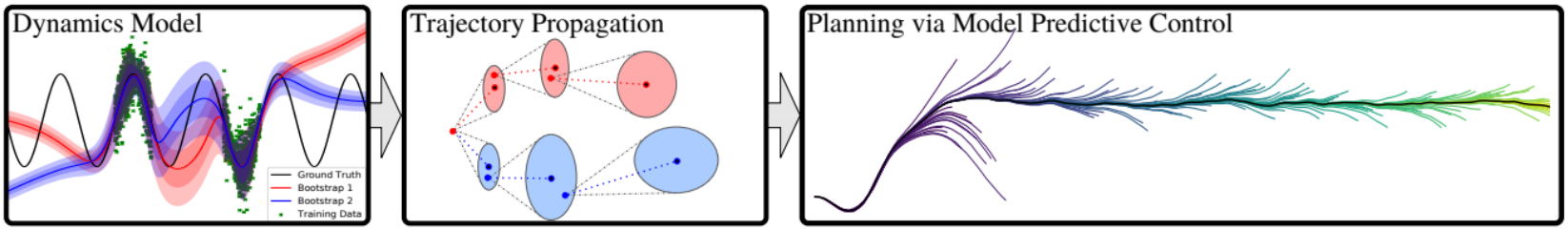


Plots show the mean and standard deviation over multiple runs and compare our model-based approach, a model-free approach (TRPO [3]), and our hybrid model-based plus model-free approach. Our combined approach shows a $3-5\times$ improvement in sample efficiency for all shown agents. Note that the x-axis uses a logarithmic scale.

Model-based RL : Shooting Algorithms

Probabilistic Ensembles with Trajectory Sampling (PETS) [Chu+18]

- Observations :
  - GP apprennent très vite mais limités à des dynamiques simples, NN très bonne capacité mais mauvaises prédictions si appris sur peu de données
  - La prise en compte de l'incertitude peut permettre de largement diminuer ce problème de sur-apprentissage des NN
- 3 Composantes :
  - Dynamics Model : Ensemble de NN probabilistes
  - Trajectory Sampling (TS) : Cross-Entropy Method [Bot+13] pour sampler actions puis application des NNs pour y faire correspondre des états
  - Planning with MPC : sélection de la meilleure séquence puis application de sa première action. Opérations répétées jusqu'à atteindre l'horizon de la tâche.



Model-based RL : Shooting Algorithms

Cross-Entropy Method pour sampler les trajectoires dans PETS :

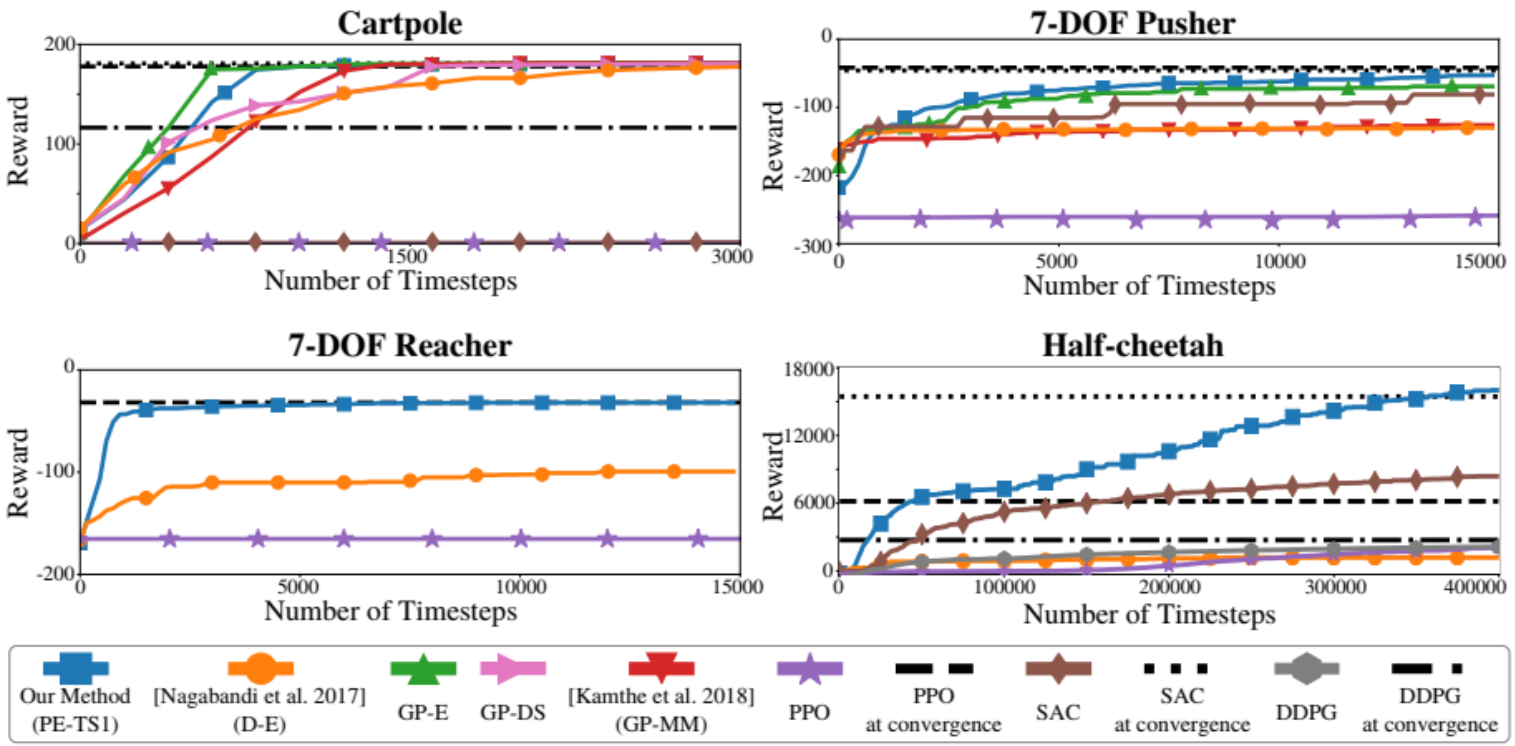
1. Tirage de N séquences de H actions selon une distribution candidate (par exemple normale multivariée $H \times d$ avec d la dimension des actions)
2. Découverte des états associés aux séquences d'actions en appliquant à chaque étape de chaque séquence un des NN tirés aléatoirement pour obtenir s_{t+1} à partir de s_t et a_t
3. Evaluation de la population de trajectoires selon le modèle de récompenses
4. Mise à jour de la distribution candidate en fonction des k meilleures trajectoires

$\Rightarrow$ Permet de guider la procédure de shooting (meilleurs résultats que Random Shooting si dynamiques complexes)

Model-based RL : Shooting Algorithms

Algorithm 1 Our model-based MPC algorithm '*PETS*':

Initialize data $\mathbb{D}$ with a random controller for one trial.
for Trial $k = 1$ to K **do**
  Train a *PE* dynamics model $\widetilde{f}$ given $\mathbb{D}$.
  for Time $t = 0$ to TaskHorizon **do**
    for Actions sampled $\boldsymbol{a}_{t:t+T} \sim$ CEM$(\cdot)$, 1 to NSamples **do**
      Propagate state particles $\boldsymbol{s}_\tau^p$ using *TS* and $\widetilde{f}|\{\mathbb{D}, \boldsymbol{a}_{t:t+T}\}$.
      Evaluate actions as $\sum_{\tau=t}^{t+T} \frac{1}{P} \sum_{p=1}^{P} r(\boldsymbol{s}_\tau^p, \boldsymbol{a}_\tau)$
      Update CEM$(\cdot)$ distribution.
    Execute first action $\boldsymbol{a}_t^*$ (only) from optimal actions $\boldsymbol{a}_{t:t+T}^*$.
    Record outcome: $\mathbb{D} \leftarrow \mathbb{D} \cup \{\boldsymbol{s}_t, \boldsymbol{a}_t^*, \boldsymbol{s}_{t+1}\}$.



Benchmarking Model-Based Reinforcement Learning [Wan+19]

⇒ PETS semble être le plus efficace en début d'apprentissage :

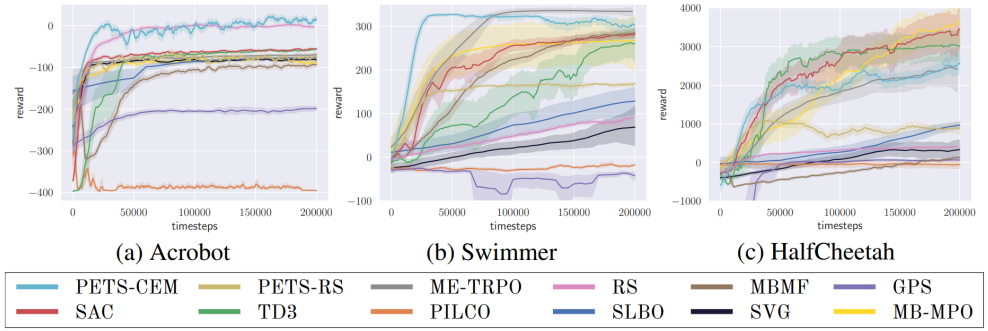


Figure 1: A subset of all 18 performance curve figures of the bench-marked algorithms. All the algorithms are run for 200k time-steps and with 4 random seeds.

⇒ Mais MBRL atteint un plateau loin de l'optimum dans la plupart des cas :

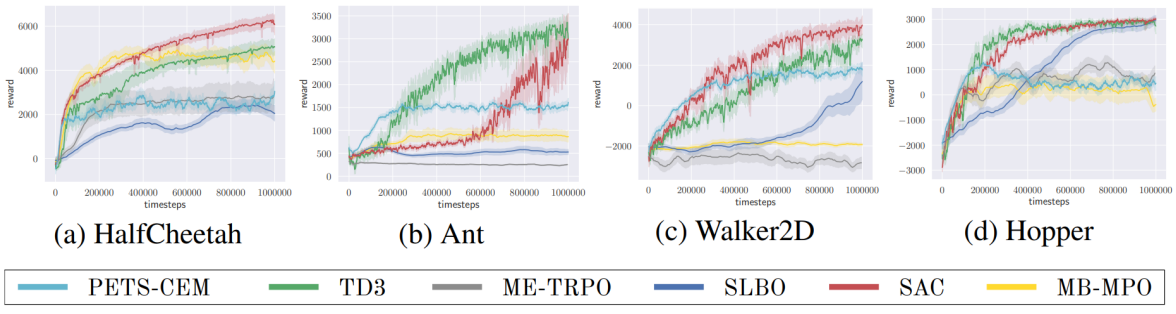


Figure 2: Performance curve for each algorithm trained for 1 million time-steps.

⇒ Pas de gagnant clair sur l'ensemble des tâches expérimentées :

	RS	MB-MF	PETS-CEM	PETS-RS	ME-TRPO	GPS	PILCO	SVG	MB-MPO	SLBO
Mean rank	5.2 / 10	5.5 / 10	4.0 / 10	4.8 / 10	5.7 / 10	7.7 / 10	9.5 / 10	4.8 / 10	4.7 / 10	4.0 / 10
Median rank	5.5 / 10	7 / 10	4 / 10	5 / 10	6 / 10	8.5 / 10	10 / 10	4 / 10	4.5 / 10	3.5 / 10

Table 5: The ranking of the MBRL algorithms in the 18 benchmarking environments

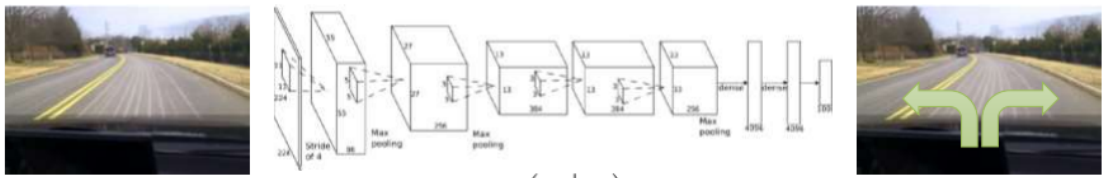
1. Model-based RL
   - Types de modèles
   - Open-loop vs Closed-loop
   - Trois familles d'approches MBRL

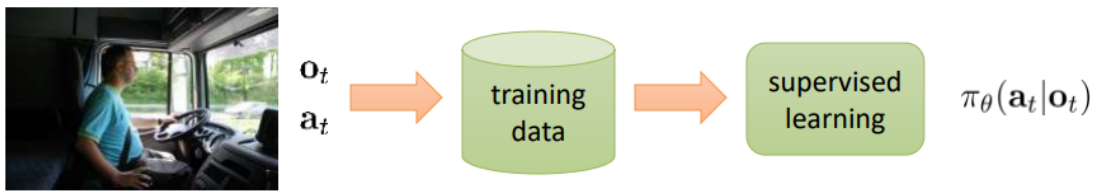
2. Imitation Learning
   - Behavior Cloning
   - Hacking
   - Hybridation Imitation Learning / Reinforcement Learning
   - Inverse RL

Imitation Learning

- Apprentissage par Renforcement
  - Souvent difficile à faire converger (problèmes de stabilité)
  - Problèmes d'exploration
  - Pas toujours la possibilité d'interagir avec l'environnement
- Dans de nombreux problèmes, on peut se servir de l'expertise humaine

  $\Rightarrow$ Ramener le problème de recherche de politique à un problème d'apprentissage supervisé : Behavior Cloning



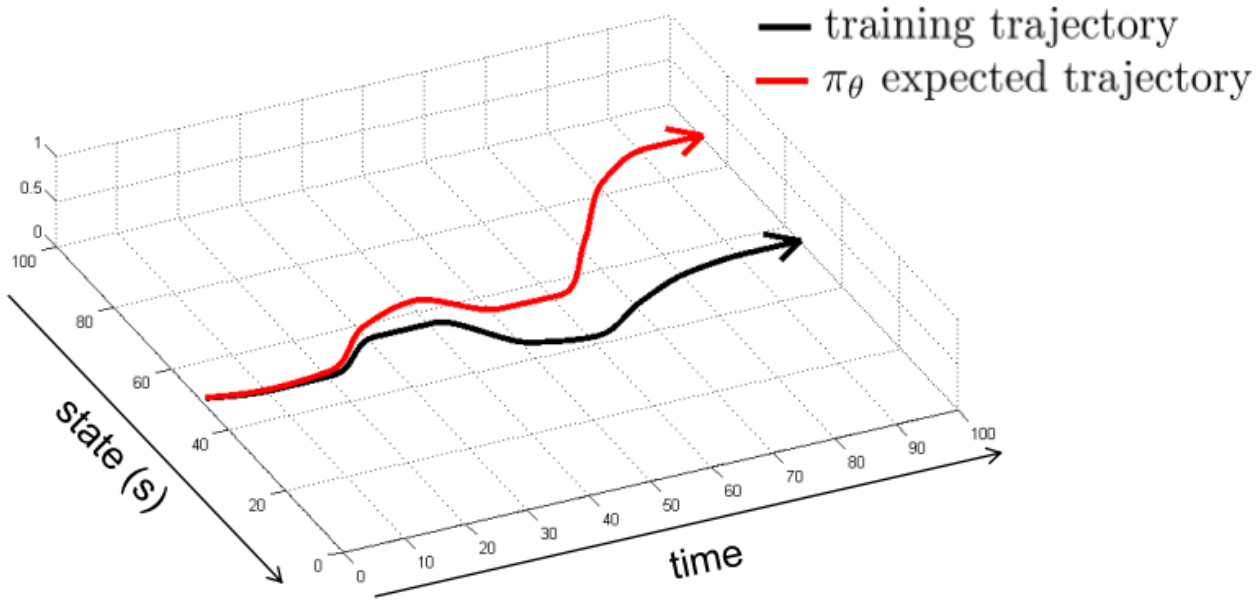


$$p_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T) = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)$$

demonstration $p(\tau)$ — train

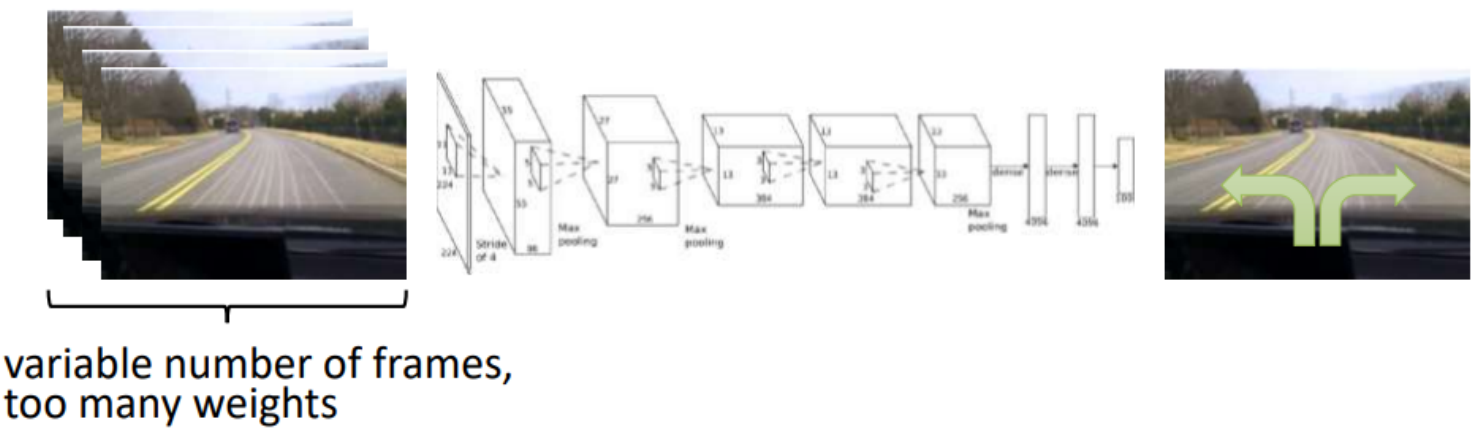
Imitation Learning

- ... Mais bien souvent cela ne fonctionne pas bien
  - Difficile de coller parfaitement au comportement humain
    - Hypothèse de Markov peu réaliste
    - Distributions multimodales, Mode Dropping
  - Manque de données
    - Connaissances limitées à un ensemble d'observations
    - Que faire dans des situations inconnues ?
- Erreurs s'accumulent au fil du processus :



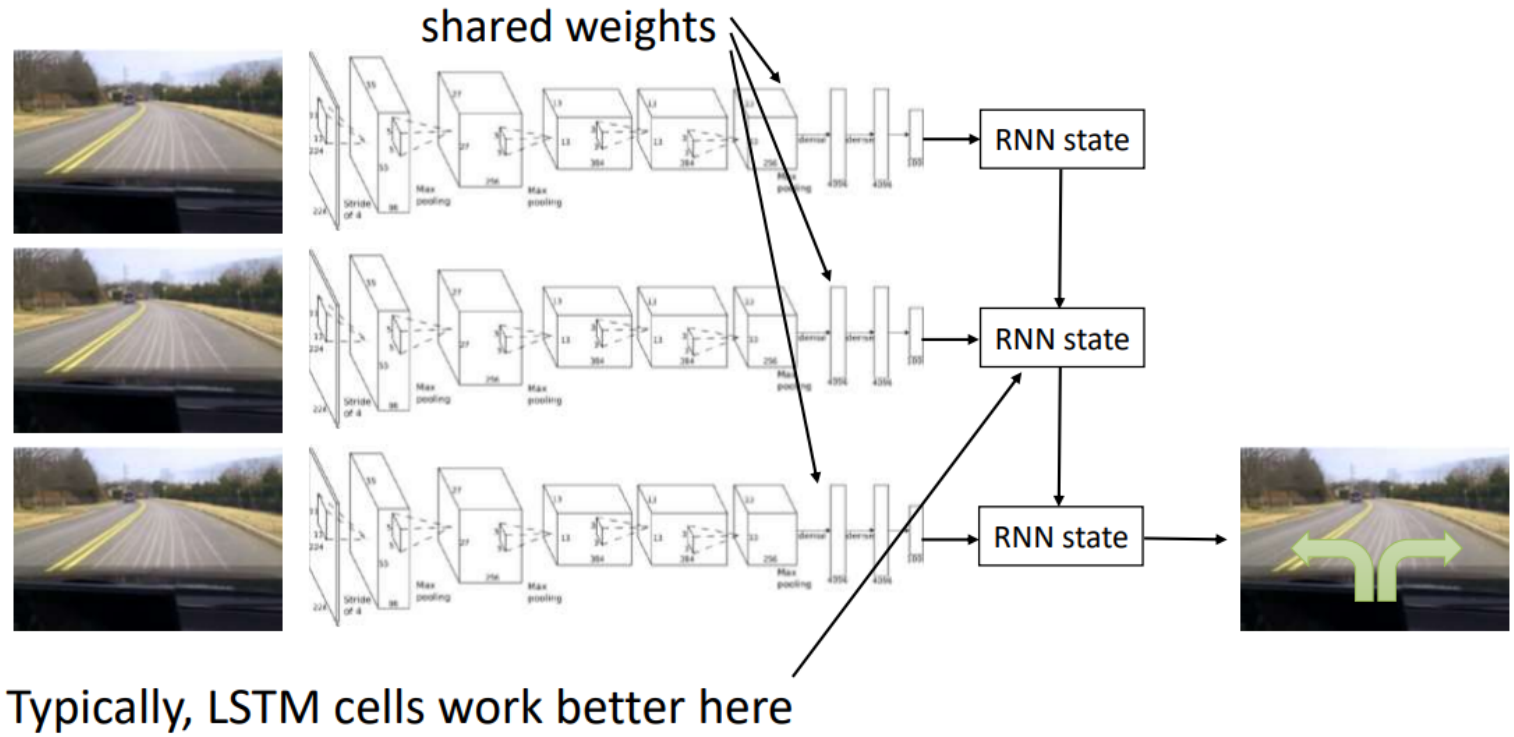
Comportement non-Markovien

- Le fait de s'appuyer sur la dernière observation uniquement peut conduire à ignorer des informations utiles :
  - Exemples : panneau de signalisation limitant la vitesse, véhicule que l'on est en train de doubler, etc.
- Prendre en compte la totalité des observations passées en entrée d'un réseau classique est inenvisageable :



Comportement non-Markovien

Possibilité : Recurrent Neural Networks

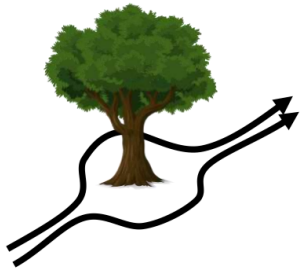


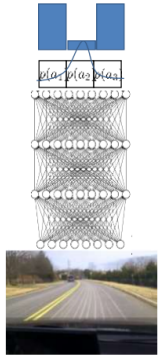
Les états du RNN représentent les états cachés d'un HMM

- Le dernier est censé contenir toute l'information provenant du passé

Distributions multi-modales

Risque d'apprendre un comportement moyen (mode dropping)

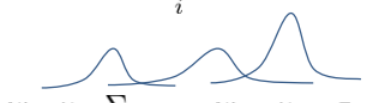




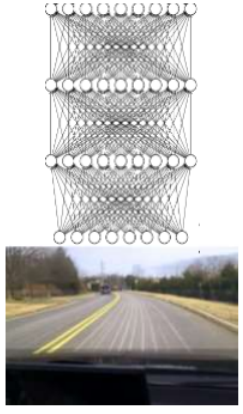
Possibilités :

Modèles de Mixtures

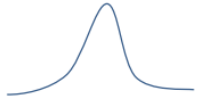
$$\pi(\mathbf{a}|\mathbf{o}) = \sum_i w_i \mathcal{N}(\mu_i, \Sigma_i)$$

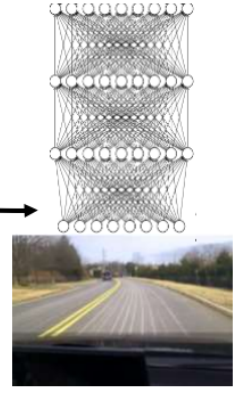


$$w_1, \mu_1, \Sigma_1, \ldots, w_N, \mu_N, \sigma_N$$



Densités Implicites





$$\xi \sim \mathcal{N}(0, \mathbf{I})$$

Imitation Learning

Malgré tout, dans la plupart des problèmes complexes :

- Impossible de coller parfaitement aux actions observées
- Manque de données d'entraînement pour éviter la dérive de la politique

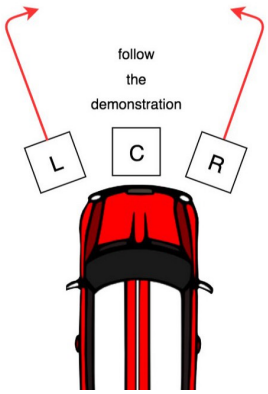
Solutions :

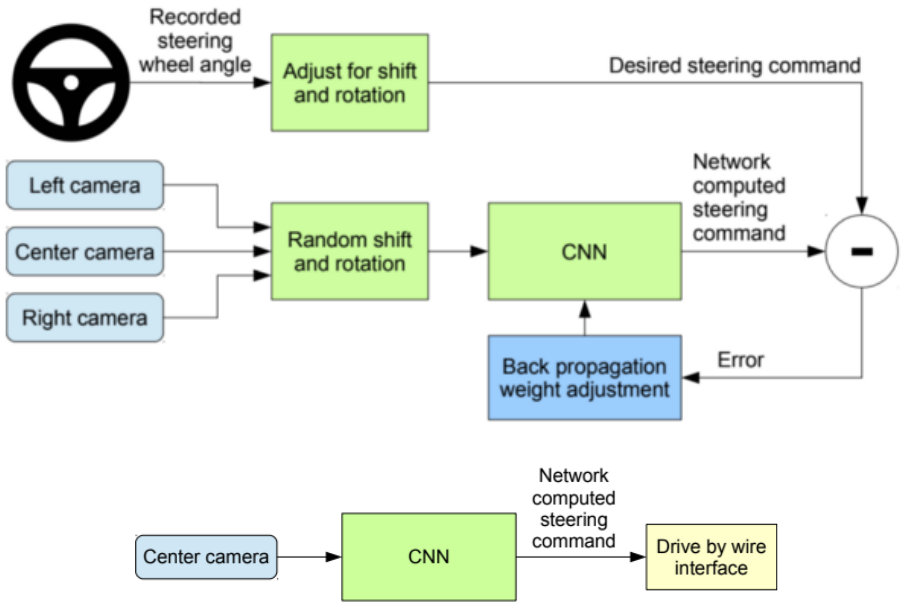
- Hacking
- Interactive Policy Learning
- Hybridation avec RL
- Inverse RL

Hacking Solution

Pour certains problèmes, il est possible [Boj+16] :

- D'utiliser des capteurs annexes pour modifier l'entrée
- De faire comme si l'entrée provenait du capteur standard
- Et d'apprendre une adaptation de l'action en conséquence





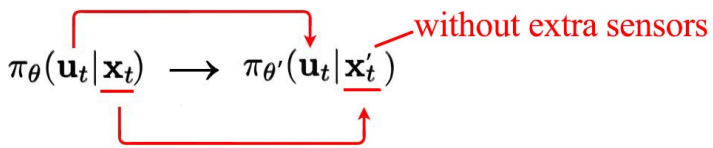
- On tourne aléatoirement la vue selon les caméras droite et gauche
- On adapte l'action d'entraînement en conséquence
- Et on apprend à choisir cette action selon cette entrée

$\Rightarrow$ Augmentation automatique du jeu de données d'entraînement

Hacking Solution

Une autre possibilité lorsque l'on a l'opportunité d'utiliser des capteurs supplémentaires pour l'apprentissage :

1. Apprendre à imiter l'humain selon les entrées de tous les capteurs
2. Apprendre à imiter la politique apprise avec des données moins fournies pour limiter les coûts

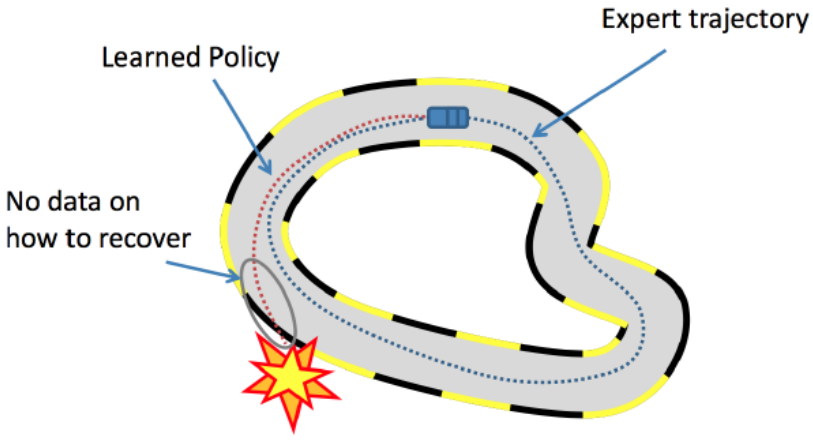


Exemple pour la conduite autonome :

- LIDAR, RADAR et vidéo lors de l'apprentissage
- Uniquement RADAR et vidéo lors de l'utilisation du système de conduite

Interactive Policy Learning

Malgré cela, on peut toujours avoir un problème de déviation dans des zones pour lesquelles on n'a pas de données :



Objective Mismatch avec ce que l'on souhaite optimiser (un coût général d'imitation qui dépendrait des distributions d'états de la politique de l'agent) :

Behavioral Cloning (Supervised Learning):

$$\text{argmin}_\theta \, E_{(s,a^*)\sim P^*} L(a^*, \pi_\theta(s))$$

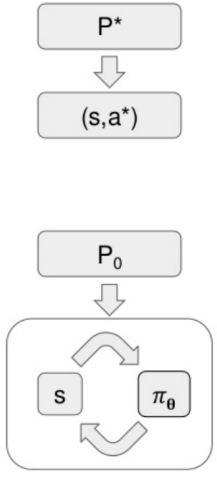
Distribution provided exogenously

Training Loss

(General) Imitation Learning:

$$\text{argmin}_\theta \, E_{s\sim P(s|\theta)} L(\pi^*(s), \pi_\theta(s))$$

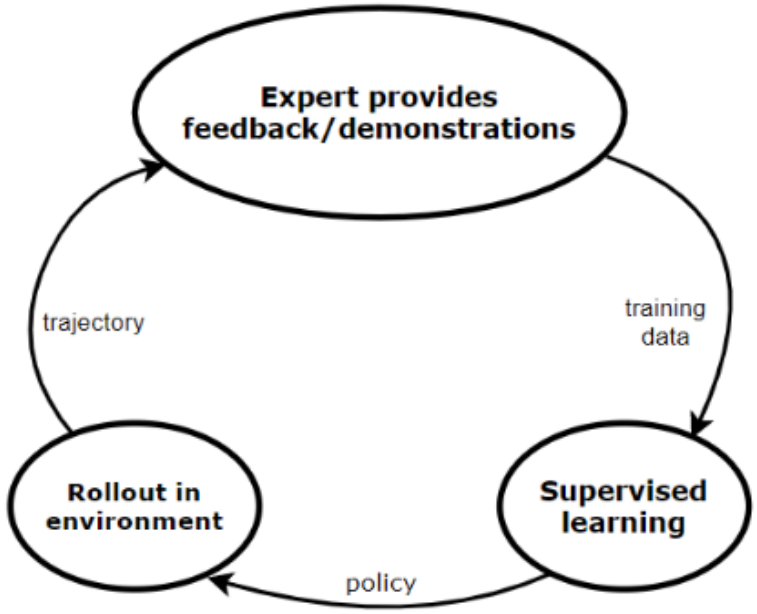
Distribution depends on rollout.
$P(s|\theta)$ = state distribution of π_θ



L'erreur croît quadratiquement avec le nombre d'étapes T des trajectoires ! [RB10]

Interactive Policy Learning

Pour répondre à ce problème, Interactive Policy Learning [AD18] :



- Objectif : Corriger la dérive et/ou Combler les zones atteintes par la politique pour lesquelles on n'a pas de données
- Requiert d'avoir accès à un démonstrateur expert

* Interactive Policy Learning : aggrégation de politiques

Forward Training Algorithm [RB10]

Politique (non stationnaire) pour T steps, initialisée selon l'expert

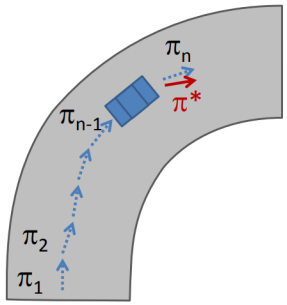
Etats s_i samplés selon la distribution non stationnaire courrante

Initialize $\pi_1^0, \ldots, \pi_T^0$ to query and execute π^*.
for $i = 1$ **to** T **do**
  Sample T-step trajectories by following π^{i-1}.
  Get dataset $\mathcal{D} = \{(s_i, \pi^*(s_i))\}$ of states, actions taken by expert at step i.
  Train classifier $\pi_i^i = \text{argmin}_{\pi \in \Pi} \mathbb{E}_{s \sim \mathcal{D}}(e_\pi(s))$.
  $\pi_j^i = \pi_j^{i-1}$ for all $j \neq i$
end for
Return $\pi_1^T, \ldots, \pi_T^T$

Etiquetages par l'expert de l'action à prendre dans l'état s_i

Apprentissage supervisé selon erreur 0-1 loss

Définition de la nouvelle politique avec les anciennes politiques pour tous les steps sauf i



- A chaque étape des trajectoires, on demande à l'expert ce qu'il aurait fait
- Etats samplés selon les politiques courantes sur les étapes précédentes
- Pour ne pas déséquilibrer les distributions des états s_i lorsque l'on reconsidère la politique selon réponses de l'expert à l'étape i, on ne change pas les politiques sur les étapes précédentes

👍 Tient compte des distributions d'états de la politique de l'agent

👍 Erreurs ne s'accroissent que linéairement avec le temps T

👎 Politiques non-stationnaires pas forcément faciles à modéliser (T réseaux de neurones indépendants ?)

👎 Difficile lorsque T est grand (grand nombre de politiques à apprendre)

* Interactive Policy Learning : SMILE

SMILE [RB10]

Politique initialisée selon l'expert

Initialize $\pi^0 \leftarrow \pi^*$ to query and execute expert.

for $i = 1$ **to** N **do**

Etats s samplés selon la distribution courrante

Etiquetages par l'expert de l'action à prendre dans chaque état s

Execute π^{i-1} to get $\mathcal{D} = \{(s, \pi^*(s))\}$.

Train classifier $\hat{\pi}^{*i} = \text{argmin}_{\pi \in \Pi} \mathbb{E}_{s \sim \mathcal{D}}(e_\pi(s))$.

Apprentissage supervisé selon erreur 0-1 loss

$\pi^i = (1-\alpha)^i \pi^* + \alpha \sum_{j=1}^{i} (1-\alpha)^{j-1} \hat{\pi}^{*j}$.

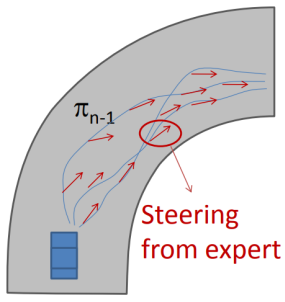
end for

Mixture de politiques

Remove expert queries: $\tilde{\pi}^N = \frac{\pi^N - (1-\alpha)^N \pi^*}{1-(1-\alpha)^N}$

Retrait de l'expert de la mixture

Return $\tilde{\pi}^N$



- Pendant N itérations, on sample à chaque étape selon une mixture de :
  - L'expert π^* (requiert interaction)
  - Une des politiques apprises sur les états atteints aux itérations précédentes
- Plus on avance dans les itérations :
  - Moins on a de chances de questionner l'expert
  - $\Rightarrow$ En fin de processus, on peut retirer l'expert π^* sans impact
  - Moins la nouvelle politique apprise n'a de poids dans la mixture
  - $\Rightarrow$ + de poids sur les politiques pour lesquelles on a appris une correction à l'itération suivante
- 👍 Erreurs en $\mathcal{O}(T \log(T)\epsilon + 1)$, avec ϵ l'erreur de classification moyenne sur π^*
- 👍 Politiques stationnaires plus faciles à implémenter
- 👍 Possibilité de réduire les interventions de l'expert en réutilisant des annotations
- 👎 Beaucoup de stochasticité, apprentissage lent

Interactive Policy Learning : DAgger

DAgger [RGB11]

Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
  Sample T-step trajectories using π_i.
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i
  and actions given by expert.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end for
Return best $\hat{\pi}_i$ on validation.

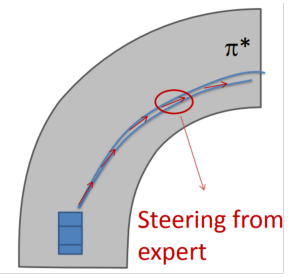
Sample de trajectoires selon Mixture entre expert et politique courante

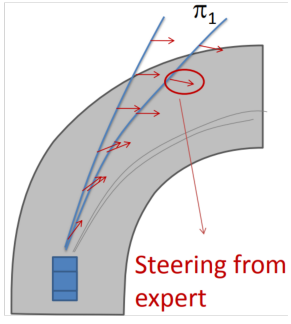
Etiquetages par l'expert de l'action à prendre dans chaque état s

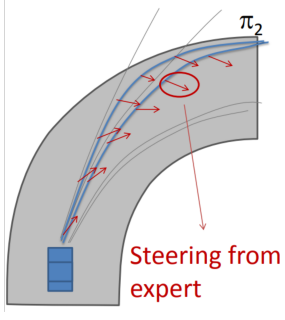
Ajout à l'ensemble d'apprentissage

Apprentissage supervisé selon erreur 0-1 loss

Selection de la meilleure politique







- Aggrégation de données, plutôt qu'aggrégation de politiques
- Pendant N itérations, on sample k trajectoires selon à chaque étape une mixture de :
  - L'expert π^* (requiert interaction)
  - La politique apprise sur les couples (etat-action) collectés aux itérations précédentes
- Avec $\beta_i \leq (1 - \alpha)^{i-1}$, plus on avance dans les itérations :
  - Moins on a de chances de sampler selon expert
  - Plus les données correspondent aux distributions d'états de l'agent

👍 Erreurs en $\mathcal{O}(T(\epsilon_N + \gamma_N)$, avec ϵ_N l'erreur moyenne sur $\mathcal{D}$ et γ_N le regret moyen sur $\mathcal{D}$ selon $\pi_{1:N}$

⇒ Vidéo SMILE/DAgger on Super Tux Kart

Imitation Learning Vs. Reinforcement Learning

Possibilité de coupler avec RL pour aller plus loin ?

Imitation Learning

- Requiert des démonstrations d'entraînement et/ou l'accès à un expert
- Peut également utiliser un modèle du monde (e.g., Guided Policy Search, PLATO)

👍 Plus stable que les méthodes de Renforcement

👎 Politique limitée aux données d'entraînement... mais que l'on peut compléter si on a accès à un expert

👎 Ne sera jamais meilleur que l'expert

Renforcement

- Pas de données d'entraînement requises, exploration

👎 Convergence difficile

👍 Mais pas de limite sur la politique découverte

Hybridation Imitation Learning / Reinforcement Learning

Idée : Utiliser les démonstrations pour initialiser le renforcement

1. collected demonstration data $(\mathbf{s}_i, \mathbf{a}_i)$
2. initialize π_θ as $\max_\theta \sum_i \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i)$
3. run π_θ to collect experience
4. improve π_θ with any RL algorithm

use demonstratin to help exploration

improve better than demonstration

Mais risque d'oublier les données de démonstration au fur et à mesure de l'apprentissage

1. collected demonstration data $(\mathbf{s}_i, \mathbf{a}_i)$
2. initialize π_θ as $\max_\theta \sum_i \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i)$
3. run π_θ to collect experience
4. improve π_θ with any RL algorithm

Collected experience may drift off from demonstration

Bad samples may corrupt the policy π

Hybridation Imitation Learning / Reinforcement Learning

Une autre possibilité est d'utiliser à la fois les trajectoires de demo et celles acquises par simulation

- Off-Policy Q-learning avec Replay Buffer conservant toujours les observations de demonstration
- ou Importance Sampling pour Policy Gradient

$$\nabla_\theta J(\theta) = \sum_{\tau \in \mathcal{D}} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) \left(\prod_{t'=1}^{t} \frac{\pi_\theta(\mathbf{a}_{t'}|\mathbf{s}_{t'})}{q(\mathbf{a}_{t'}|\mathbf{s}_{t'})} \right) \left(\sum_{t'=t}^{T} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \right) \right]$$

from experience and demonstration

avec pour q_{demo} 2 possibilités :

- dépend d'une politique apprise par Behavior Cloning
- utilise une distribution de Dirac : $q_{demo}(\tau) = \frac{1}{N}\delta(\tau \in \mathcal{D})$, avec N la taille de $\mathcal{D}$

$\Rightarrow$ Toutes les techniques pour réduire la variance en Off-Policy peuvent s'appliquer (Weighted IS, Retrace, etc.)

Hybridation Imitation Learning / Reinforcement Learning

Enfin, possibilité de combiner imitation et renforcement dans un objectif commun

- Combinaison linéaire :

$$E_{\pi_\theta}[r(\mathbf{s},\mathbf{a})] + \lambda \sum_{(\mathbf{s},\mathbf{a})\in\mathcal{D}_{\text{demo}}} \log \pi_\theta(\mathbf{a}|\mathbf{s})$$

RL — Imitation learning

Deux possibilités :

- Pour Policy Gradients [Raj+17] :

$$g_{aug} = \sum_{(s,a)\in\rho_\pi} \nabla_\theta \ln \pi_\theta(a|s) A^\pi(s,a) + \sum_{(s,a^*)\in\rho_D} \nabla_\theta \ln \pi_\theta(a^*|s) w(s,a^*)$$

policy gradient — make policy closer to demonstrations

weighting function

$$w(s,a) = \lambda_0 \lambda_1^k \max_{(s',a')\in\rho_\pi} A^\pi(s',a') \quad \forall (s,a) \in \rho_D$$

- Pour Q-Learning [Hes+18] :

$$J(Q) = J_{DQ}(Q) + \lambda_1 J_n(Q) + \lambda_2 J_E(Q) + \lambda_3 J_{L2}(Q)$$

regularization loss

Q-learning loss

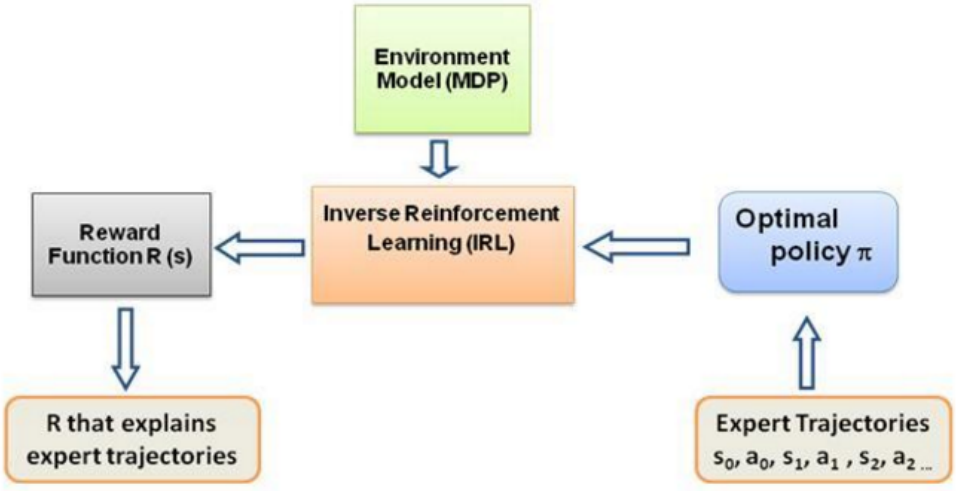
n-step Q-learning loss

$$J_E(Q) = \max_{a\in A}[Q(s,a) + l(a_E,a)] - Q(s,a_E)$$

(marginal loss on example action)

Inverse Reinforcement Learning

Autre possibilité, Inverse Reinforcement Learning :



⇒ Étant donné un ensemble de trajectoires d'expert, l'objectif de l'Inverse Reinforcement Learning (IRL) est de retrouver la **fonction de récompense** associée

- Challenges :
  - La politique optimale est souvent inconnue
  - Le modèle de l'environnement est souvent inconnu
  - Trajectoires de l'expert parfois bruitées

⇒ Apprenticeship Learning : recherche de politique π qui est aussi performante que la politique de l'expert π_E au sens d'une fonction de coût cachée $c_{true} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ [AN04]

Inverse Reinforcement Learning

Une des formulations IRL les plus répandues est MaxEntIRL [Zie+08]

- On suppose que les trajectoires de l'expert sont échantillonnées selon : $p(\tau) = \frac{1}{Z} exp(-c_\theta(\tau))$, avec $c_\theta(\tau) = \sum_t c_\theta(x_t, u_t)$

$\Rightarrow$ L'expert génère des trajectoires selon une probabilité qui décroît exponentiellement avec le coût de la trajectoire

- La fonction de partition Z correspond à la somme des probabilités sur toutes les trajectoires du monde selon la fonction c_θ
- Cette distribution d'Inverse Optimal Control (IOC) suit le principe de maximum d'entropie (maxEnt) : les trajectoires sont uniquement discriminées en fonction de leur espérance de coût

$\Rightarrow$ Si on est capable de calculer Z : on peut maximiser la probabilité des trajectoires de l'expert par maximum de vraisemblance $p(\{\tau\})$...

- ... Mais c'est très difficile. Dans la version de [Zie+08], la fonction de partition était calculée par programmation dynamique
  - Mais seulement possible pour de tous petits problèmes jouets, avec dynamiques simples
  - Impraticable si le modèle de l'environnement est inconnu

Inverse Reinforcement Learning

Guided Cost Learning [FLA16] :

- Estimation de la distribution par des trajectoires samplées de l'environnement $\mathcal{D}_{samp}$ selon $q(\tau_j)$. La negative log-likelihood de l'ensemble de demonstration $\mathcal{D}_{demo}$ selon la distribution IOC est donnée par :

$$\begin{aligned}\mathcal{L}_{\text{cost}}(\theta) &= \frac{1}{N}\sum_{\tau_i \in \mathcal{D}_{demo}} c_\theta(\tau_i) + \log Z \\ &\approx \frac{1}{N}\sum_{\tau_i \in \mathcal{D}_{demo}} c_\theta(\tau_i) + \log \frac{1}{M}\sum_{\tau_j \in \mathcal{D}_{samp}} \frac{\exp(-c_\theta(\tau_j))}{q(\tau_j)}\end{aligned}$$

où on estime par IS la partition selon $\mathcal{D}_{samp}$. Avec $w_j = \frac{\exp(-c_\theta(\tau_j))}{q(\tau_j)}$ et $Z = \sum_{\tau_j \in \mathcal{D}_{samp}} w_j$, ce coût s'optimise par le gradient :

$$\frac{d\mathcal{L}_{\text{cost}}}{d\theta} = \frac{1}{N}\sum_{\tau_i \in \mathcal{D}_{demo}} \frac{dc_\theta}{d\theta}(\tau_i) - \frac{1}{Z}\sum_{\tau_j \in \mathcal{D}_{samp}} w_j \frac{dc_\theta}{d\theta}(\tau_j)$$

- Pour réduire la variance, on peut tirer la moitié du temps une trajectoire de l'expert et considérer, avec le Z précédent, $\widetilde{p}(\tau) = \frac{1}{Z}\exp(-c_\theta(\tau))$:

$$\mathcal{L}_{\text{cost}}(\theta) = \mathbb{E}_{\tau \sim p}[c_\theta(\tau)] + \log\left(\mathbb{E}_{\tau \sim \frac{1}{2}p + \frac{1}{2}q}\left[\frac{\exp(-c_\theta(\tau))}{\frac{1}{2}\widetilde{p}(\tau) + \frac{1}{2}q(\tau)}\right]\right)$$

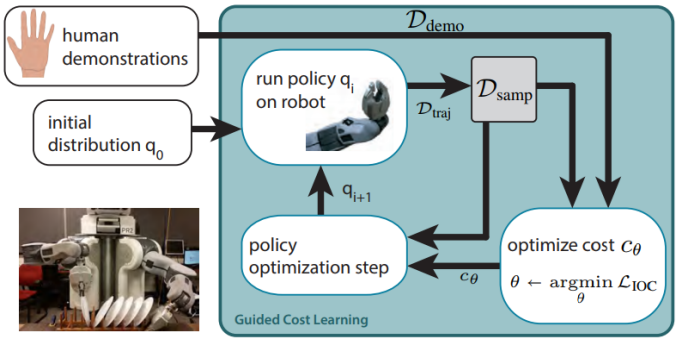
Inverse Reinforcement Learning

Guided Cost Learning [FLA16] :

- Idéalement on doit prendre :

$q(\tau) \triangleq q_\phi(\tau) \propto exp(-c_\theta(\tau))$, car dans ce cas :

$$\frac{1}{M} \sum_{\tau_j \in \mathcal{D}_{samp}} \frac{\exp\left(-c_\theta(\tau_j)\right)}{q(\tau_j)} = \frac{1}{M} \sum_{\tau_j \in \mathcal{D}_{samp}} Z = Z$$



- Problème de RL selon θ courant :
  $\min_q E_q[c_\theta(\tau)] - \mathcal{H}(\tau)$
  - $\Rightarrow$ Peut être optimisé par GuidedPolicySearch [LA14] (iLQG avec coût de KL et décisions gaussiennes)
  - $\Rightarrow$ Converge vers $q(\tau) \propto exp(-c_\theta(\tau))$ (maximum d'entropie) [Zie10]

Algorithm 1 Guided cost learning

1: Initialize $q_k(\tau)$ as either a random initial controller or from demonstrations
2: **for** iteration $i = 1$ to I **do**
3: Generate samples $\mathcal{D}_{\text{traj}}$ from $q_k(\tau)$
4: Append samples: $\mathcal{D}_{\text{samp}} \leftarrow \mathcal{D}_{\text{samp}} \cup \mathcal{D}_{\text{traj}}$
5: Use $\mathcal{D}_{\text{samp}}$ to update cost c_θ using Algorithm 2
6: Update $q_k(\tau)$ using $\mathcal{D}_{\text{traj}}$ and the method from [LA14] to obtain $q_{k+1}(\tau)$
7: **end for**
8: **return** optimized cost parameters θ and trajectory distribution $q(\tau)$

Algorithm 2 Nonlinear IOC with stochastic gradients

1: **for** iteration $k = 1$ to K **do**
2: Sample demonstration batch $\hat{\mathcal{D}}_{\text{demo}} \subset \mathcal{D}_{\text{demo}}$
3: Sample background batch $\hat{\mathcal{D}}_{\text{samp}} \subset \mathcal{D}_{\text{samp}}$
4: Append demonstration batch to background batch: $\hat{\mathcal{D}}_{\text{samp}} \leftarrow \hat{\mathcal{D}}_{\text{demo}} \cup \hat{\mathcal{D}}_{\text{samp}}$
5: Estimate $\frac{d\mathcal{L}_{\text{IOC}}}{d\theta}(\theta)$ using $\hat{\mathcal{D}}_{\text{demo}}$ and $\hat{\mathcal{D}}_{\text{samp}}$
6: Update parameters θ using gradient $\frac{d\mathcal{L}_{\text{IOC}}}{d\theta}(\theta)$
7: **end for**
8: **return** optimized cost parameters θ

Ce qui revient à optimiser :

$$\max_{c \in \mathcal{C}} \min_{\pi \in \Pi} \left(-H(\pi) + \mathbb{E}_\pi[c(s,a)]\right) - \mathbb{E}_{\pi_E}[c(s,a)], \text{ avec } c(s,a) = \sum \gamma^t c(s_t, a_t)$$

*Equivalence entre GAN et IRL

Soit une archiecture GAN, dont le coût (classique) du discriminateur est :

$$\mathcal{L}_{\text{discriminator}}(D) = \mathbb{E}_{\mathbf{x}\sim p}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim G}[-\log(1 - D(\mathbf{x}))]$$

Le coût du générateur peut prendre plusieurs formes :

- Il est classiquement définit comme l'opposé du discriminateur mais celà mène souvent à des gradients nuls lorsque la discrimination est trop simple
- Une autre version commune est de considérer la confusion du discriminateur sur les samples générés :

$$\mathcal{L}_{\text{generator}}(G) = \mathbb{E}_{\mathbf{x}\sim G}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim G}[\log(1 - D(\mathbf{x}))]$$

Le discriminateur peut prendre également plusieurs formes :

- Il est classiquement définit sous la forme d'une sigmoïde : $D(\tau) = \frac{exp(f_\theta(\tau))}{exp(f_\theta(\tau))+1}$
- En notant que le discriminateur optimal est donné par $D^*(\tau) = \frac{p(\tau)}{p(\tau)+q(\tau)}$, avec $p(\tau)$ la distribution (inconnue) des trajectoires réelles et $q(\tau)$ la distribution du générateur [Goo+14], on peut considérer $D(\tau) = \frac{\frac{1}{Z}exp(-c_\theta(\tau))}{\frac{1}{Z}exp(-c_\theta(\tau))+q(\tau)}$, avec Z un paramètre à estimer [Fin+16]

  $\Rightarrow$ Discriminateur optimal lorsque $\frac{1}{Z}exp(-c_\theta(\tau)) = p(\tau)$
  $\Rightarrow$ GAN potentiellement plus stable car $p(\tau)$ et $q(\tau)$ modélisés explicitement et indépendamment dans le discriminateur.

*Equivalence entre GAN et IRL

Soit $\tilde{p}(\tau) = \frac{1}{Z} exp(-c_\theta(\tau))$, $\mu(\tau) = \frac{1}{2}p(\tau) + \frac{1}{2}q(\tau)$ et $\tilde{\mu}(\tau) = \frac{1}{2}\tilde{p}(\tau) + \frac{1}{2}q(\tau)$.

Le coût du discriminateur peut alors s'écrire :

$$\begin{aligned}
\mathcal{L}_{\text{discriminator}}\ (D_\theta) &= \mathbb{E}_{\tau\sim p}\left[-\log \frac{\frac{1}{Z}\exp\left(-c_\theta(\tau)\right)}{\tilde{\mu}(\tau)}\right] + \mathbb{E}_{\tau\sim q}\left[-\log \frac{q(\tau)}{\tilde{\mu}(\tau)}\right] \\
&= \log Z + \mathbb{E}_{\tau\sim p}\left[c_\theta(\tau)\right] + \mathbb{E}_{\tau\sim p}[\log \tilde{\mu}(\tau)] \\
&- \mathbb{E}_{\tau\sim q}[\log q(\tau)] + \mathbb{E}_{\tau\sim q}[\log \tilde{\mu}(\tau)] \\
&= \log Z + \mathbb{E}_{\tau\sim p}\left[c_\theta(\tau)\right] - \mathbb{E}_{\tau\sim q}[\log q(\tau)] + 2\mathbb{E}_{\tau\sim \mu}[\log \tilde{\mu}(\tau)]
\end{aligned}$$

Pour θ fixé, le minimum de ce coût est atteint lorsque :

$$\partial_Z \mathcal{L}_{\text{discriminiator}}\ (D_\theta) = 0$$

$$\frac{1}{Z} = \mathbb{E}_{\tau\sim\mu}\left[\frac{\frac{1}{Z^2}\exp\left(-c_\theta(\tau)\right)}{\tilde{\mu}(\tau)}\right]$$

$$Z = \mathbb{E}_{\tau\sim\mu}\left[\frac{\exp\left(-c_\theta(\tau)\right)}{\tilde{\mu}(\tau)}\right]$$

Ce qui correspond exactement au terme Z considéré par Guided Cost Learning (GCL) :

$$\mathcal{L}_{\text{cost}}\ (\theta) = \mathbb{E}_{\tau\sim p}\left[c_\theta(\tau)\right] + \log\left(\mathbb{E}_{\tau\sim\mu}\left[\frac{\exp\left(-c_\theta(\tau)\right)}{2\tilde{\mu}(\tau)}\right]\right)$$

*Equivalence entre GAN et IRL

D'autre part, on a :

$$\begin{aligned}
\partial_\theta \mathcal{L}_{\text{discriminator}}\ (D_\theta) &= \mathbb{E}_{\tau\sim p}\left[\partial_\theta c_\theta(\tau)\right] + \mathbb{E}_{\tau\sim\mu}\left[\partial_\theta log\widetilde{\mu}(\tau)\right] \\
&= \mathbb{E}_{\tau\sim p}\left[\partial_\theta c_\theta(\tau)\right] - \mathbb{E}_{\tau\sim\mu}\left[\frac{\frac{1}{Z}\exp\left(-c_\theta(\tau)\right)\partial_\theta c_\theta(\tau)}{\widetilde{\mu}(\tau)}\right] \\
&= \mathbb{E}_{\tau\sim p}\left[\partial_\theta c_\theta(\tau)\right] \\
&+ \left(\mathbb{E}_{\tau\sim\mu}\left[\frac{-\exp\left(-c_\theta(\tau)\right)\partial_\theta c_\theta(\tau)}{\widetilde{\mu}(\tau)}\right] / \mathbb{E}_{\tau\sim\mu}\left[\frac{\exp\left(-c_\theta(\tau)\right)}{\widetilde{\mu}(\tau)}\right]\right) \\
&= \mathbb{E}_{\tau\sim p}\left[\partial_\theta c_\theta(\tau)\right] + \partial_\theta \log\left(\mathbb{E}_{\tau\sim\mu}\left[\frac{\exp\left(-c_\theta(\tau)\right)}{\widetilde{\mu}(\tau)}\right]\right) = \partial_\theta \mathcal{L}_{\text{cost}}\ (\theta)
\end{aligned}$$

$\Rightarrow$ Optimiser $\mathcal{L}_{\text{discriminator}}\ (D_\theta)$ du GAN revient au même que minimiser $\mathcal{L}_{\text{cost}}\ (\theta)$ du modèle d'IRL GCL [Fin+16]

Enfin :

$$\begin{aligned}
\mathcal{L}_{\text{generator}}\ (q) &= \mathbb{E}_{\tau\sim q}[\log(1 - D(\tau)) - \log(D(\tau))] \\
&= \mathbb{E}_{\tau\sim q}\left[\log\frac{q(\tau)}{\widetilde{\mu}(\tau)} - \log\frac{\frac{1}{Z}\exp\left(-c_\theta(\tau)\right)}{\widetilde{\mu}(\tau)}\right] \\
&= \mathbb{E}_{\tau\sim q}\left[\log q(\tau) + \log Z + c_\theta(\tau)\right] \\
&= \log Z + \mathbb{E}_{\tau\sim q}\left[c_\theta(\tau)\right] + \mathbb{E}_{\tau\sim q}[\log q(\tau)] = \log Z + \mathcal{L}_{\text{sampler}}\ (q)
\end{aligned}$$

avec $\mathcal{L}_{\text{sampler}}\ (q) = \mathbb{E}_{\tau\sim q}\left[c_\theta(\tau)\right] - \mathcal{H}(\tau)$ le problème d'optimisation interne de Guided Cost Learning.

$\Rightarrow$ Puisque Z est fixe lors de l'optimisation du générateur, optimiser le générateur est donc également équivalent à l'optimisation d'échantillonnage de GCL !

Inverse Reinforcement Learning

$\Rightarrow$ Connexion entre GANs et IRL

- Modification mineure de GAN permet de se ramener à un modèle classique d'IRL [Fin+16]
- Différences cependant :
  - Objectif principal de l'IRL : retrouver la fonction de coût qui drive l'expert (génération = problème de RL interne)
  - Objectif principal des GANs : obtenir un générateur qui approche la distribution des trajectoires observées (discrimination = problème de classification interne)
- GANs a priori plus rapides que IRL pour des tâches d'Imitation Learning

Inverse Reinforcement Learning

Generative Adversarial Imitation Learning (GAIL) [HE16]

- Recherche de générateur inspiré de [HGE16] qui propose un modèle d'IRL basé sur TRPO en alternant les deux étapes :
  1. Echantillonnage de la politique courante pour optimisation de la fonction de coût
  2. Optimisation de la politique selon un pas de TRPO en considérant la fonction de coût courante
- Problème si le vrai coût n'appartient pas à la classe $\mathcal{C}$ définie (linéaire ou convexe dans [HGE16] pour résolution analytique)

$\Rightarrow$ GAIL simplifie le problème en remplaçant la fonction de coût par un discriminateur GAN

Algorithm 1 Generative adversarial imitation learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
2: **for** $i = 0, 1, 2, \ldots$ **do**
3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
4: Update the discriminator parameters from w_i to w_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \tag{17}$$

5: Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

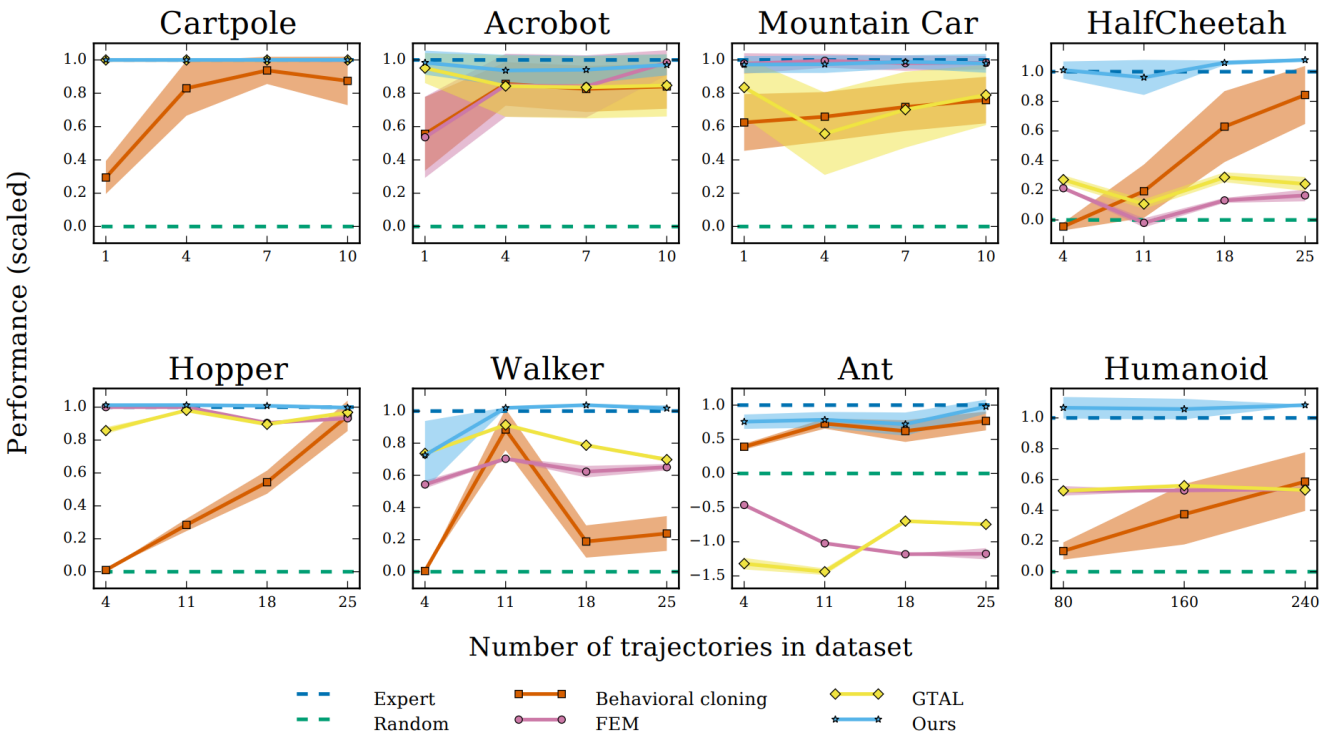
$$\begin{aligned} &\hat{\mathbb{E}}_{\tau_i}\left[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)\right] - \lambda \nabla_\theta H(\pi_\theta), \\ &\text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \,|\, s_0 = \bar{s}, a_0 = \bar{a}] \end{aligned} \tag{18}$$

6: **end for**

Inverse Reinforcement Learning

Performances de GAIL comparé à :

- Behavioral cloning
- FEM : modèle de [HGE16] avec coût linéaire
  $c \in \mathcal{C}_{\text{linear}} = \left\{ \sum_i w_i f_i : \|w\|_2 \leq 1 \right\}$)
- GTAL : modèle de [HGE16] avec coût convexe
  $c \in \mathcal{C}_{\text{convex}} = \left\{ \sum_i w_i f_i : \sum_i w_i = 1, w_i \geq 0 \forall i \right\}$



Imitation Learning : MALIGAN [Che+17]

Objectif : contrer le problème du reward mouvant $D_\phi(\tau)$ de GAIL

- Alors que dans les cas du GAN classique, on travaille avec des sorties continues ce qui permet de retropropager à travers le discriminateur
- Dans le cadre de GAIL : problème de RL avec récompense $D_\phi(\tau)$ mouvante : moins stable

Constat : à l'optimum on a $D_\phi(\tau) = \frac{p_{data}(\tau)}{p_{data}(\tau)+p_\theta(\tau)}$ pour toute trajectoire τ

$\Rightarrow$ Donc : $p_{data}(\tau) = \frac{D_\phi(\tau)}{1-D_\phi(\tau)} p_\theta(\tau)$, $p_\theta(\tau)$ la distribution du générateur

Soit : $q(\tau) = \frac{r_D(\tau)}{Z(\theta)} p_\theta(\tau)$, avec $r_D(\tau) = \frac{D_\phi(\tau)}{1-D_\phi(\tau)}$ et $Z(\theta)$ une fonction de partition permettant de s'assurer que la distribution intègre bien à 1 (pas forcément le cas sans, du fait des erreurs d'estimation de $D(\tau)$)

Idée : En supposant $D(\tau)$ suffisamment bien optimisé à chaque itération, $q(\tau) \approx p_{data}(\tau)$ relativement constante tout au long de l'apprentissage.

Proposition : Minimiser la divergence $L_G(\theta) = \mathrm{KL}\left(q(\mathbf{x}) \| p_\theta(\mathbf{x})\right)$

$$\nabla L_G = \mathbb{E}_q \left[\nabla_\theta \log p_\theta(\tau)\right] = \mathbb{E}_{p_\theta}\left[\frac{q(\tau)}{p_\theta(\tau)} \nabla_\theta \log p_\theta(\tau)\right]$$

$$= \frac{1}{Z(\theta)} \mathbb{E}_{p_\theta}\left[r_D(\tau) \nabla_\theta \log p_\theta(\tau)\right] \approx \sum_{i=1}^{m} \left(\frac{r_D(\mathbf{x}_i)}{\sum_i r_D(\tau_i)} - b\right) \nabla \log p_\theta(\tau_i)$$

* Imitation Learning : MALIGAN [Che+17]

Algorithm 1 MaliGAN

Require: A generator p with parameters θ.
A discriminator $D(x)$ with parameters θ_d.
A baseline b.

1: **for** number of training iterations **do**
2: **for** k steps **do**
3: Sample a minibatch of samples $\{\mathbf{x}_i\}_{i=1}^m$ from p_θ.
4: Sample a minibatch of samples $\{\mathbf{y}_i\}_{i=1}^m$ from p_d.
5: Update the parameter of discriminator by taking gradient ascend of discriminator loss

$$\sum_i [\nabla_{\theta_d} \log D(\mathbf{y}_i)] + \sum_i [\nabla_{\theta_d} \log(1 - D(\mathbf{x_i}))]$$

6: **end for**
7: Sample a minibatch of samples $\{\mathbf{x}_i\}_{i=1}^m$ from p_θ.
8: Update the generator by applying gradient update

$$\sum_{i=1}^m \left(\frac{r_D(\mathbf{x}_i)}{\sum_i r_D(\mathbf{x}_i)} - b\right) \nabla \log p_\theta(\mathbf{x}_i)$$

9: **end for**

* Imitation Learning : ColdGAN [Sci+20]

Application : génération de langue conditionnée selon X

Probabilité d'un texte :

$$p_\theta(Y|X) = \prod_{t=1}^{N} p(y_t|y_1, ..., y_{t-1}, X, \theta)$$

avec $p(y_t = i|y_1, ..., y_{t-1}, X, \theta) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$ où :

- z est le vecteur de logits produits par le réseau de neurones conditionné par $y_1, ..., y_{t-1}$ et X
- T est un paramètre de température.

La plupart des approches fonctionnent en maximisant la log-vraisemblance selon des Y cible

- Mais risque de trop se focaliser sur les Y connus alors que de multiples solutions possibles
- Mais Exposure Bias : à tout $t > 1$ états des réseaux possiblement différents en génération pure (passé généré) par rapport à ce sur quoi les réseaux ont été appris (via Teacher forcing où passé connu)

Proposition : Utiliser GAIL pour apprendre à "imiter" les textes (trajectoires) cibles en considérant $\log(D_\phi(X, Y))$ comme reward

$$\frac{1}{|H|} \sum_{(X,Y)\in H} \log(D_\phi(X, Y)) + \frac{1}{|G|} \sum_{(X,Y)\in G} \log(1 - D_\phi(X, Y))$$

avec H un ensemble de conditions d'entrée X associées à des textes humains Y, et G un ensemble de textes X associés aux Y générés.

* Imitation Learning : ColdGAN [Sci+20]

Problème : Discriminateurs pas bons partout en texte (et surtout pas bons proche du mode de la distribution quand T=0)

	Evaluated on			
	human	$T = 0$	$T = 1$	$T = \infty$
$D_{T=0}$	.79	.17	.84	.92
$D_{T=1}$	.79	.76	.23	.09
$D_{T=\infty}$	.92	.92	.91	.08
$D_{perfect}$	1	0	0	0

Proposition :

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \hat{\pi}_\theta} \left[\frac{\pi_\theta(\tau)}{\hat{\pi}_\theta(\tau)} D_\phi(\tau) \sum_{t=1}^{|\tau|-1} \nabla_\theta \log \pi_\theta \left(\tau_t | \tau_{1:t-1} \right) \right]$$

Avec $\hat{\pi}_\theta$ une distribution avec température $T < 1$ pour se concentrer sur une zone resserrée de l'espace.

Astuce d'ACER pour éviter l'explosion des gradients :

$$\mathbb{E}_{\tau \sim \hat{\pi}_\theta} \left[\min(c, w(\tau)) D_\phi(\tau) \nabla \log \pi_\theta(\tau) \right] + \mathbb{E}_{\tau \sim \pi_\theta} \left[\max \left(0, \frac{w(\tau) - c}{w(\tau)} \right) D_\phi(\tau) \nabla \log \pi_\theta(\tau) \right]$$

$\Rightarrow$ Très bons résultats en génération de résumés abstractifs.

Imitation Learning

Pour résumer, différents types d'imitation learning :

Behavioral Cloning

$$\text{argmin}_{\theta} E_{(s,a^*)\sim P^*} L(a^*, \pi_{\theta}(s))$$

Works well when P^* close to P_{θ}

Inverse RL

Learn r such that:

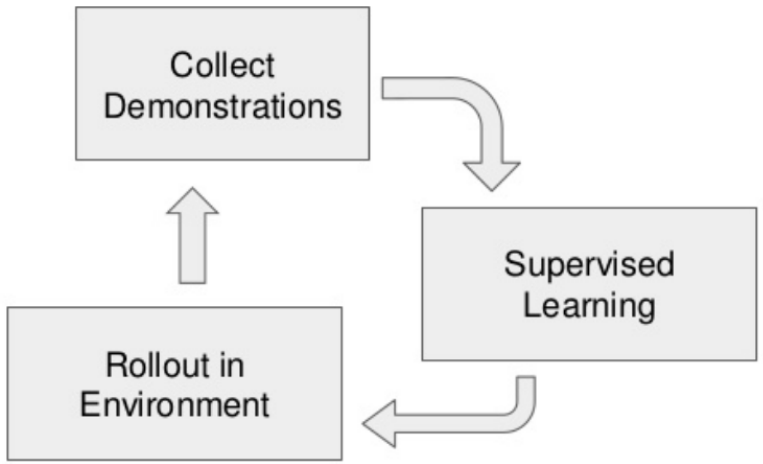
$$\pi^* = \text{argmax}_{\theta} E_{s\sim P(s|\theta)} r(s, \pi_{\theta}(s))$$

RL problem

Assumes learning r is statistically easier than directly learning π^*

Direct Policy Learning

via Interactive Demonstrator



Requires Interactive Demonstrator (BC is 1-step special case)

Imitation Learning

	Direct Policy Learning	Reward Learning	Access to Environment	Interactive Demonstrator	Pre-collected Demonstrations
Behavioral Cloning	Yes	No	No	No	Yes
Direct Policy Learning (Interactive IL)	Yes	No	Yes	Yes	Optional
Inverse Reinforcement Learning	No	Yes	Yes	No	Yes

Sources

- Sergei Levine : http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_9_model_based_rl.pdf
- Michael Zhang : https://michaelrzhang.github.io/gps
- Jonathan Hui : https://medium.com/@jonathan_hui/rl-imitation-learning-ac28116c02fc
- Zoltán Lőrincz : https://medium.com/@SmartLabAI/a-brief-overview-of-imitation-learning-8a8a75c44a9c
- Yisong Yue : https://fr.slideshare.net/yisongyue/imitation-learning-tutorial
- Stephane Ross : https://www.cs.cmu.edu/~sross1/publications/Ross-AIStats11-Slides.pdf
- Emma Brunskill : http://web.stanford.edu/class/cs234/slides/lecture7.pdf
- Rishabh Jangir : https://jangirrishabh.github.io/2016/07/09/virtual-car-IRL/

References I

[AD18] Alexandre Attia et Sharone Dayan. « Global overview of imitation learning ». In : *arXiv preprint arXiv :1801.06503* (2018).

[AN04] Pieter Abbeel et Andrew Y Ng. « Apprenticeship learning via inverse reinforcement learning ». In : *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, p. 1.

[Blu+15] Charles Blundell et al. « Weight uncertainty in neural networks ». In : *arXiv preprint arXiv :1505.05424* (2015).

[Boj+16] Mariusz Bojarski et al. « End to end learning for self-driving cars ». In : *arXiv preprint arXiv :1604.07316* (2016).

[Bot+13] Zdravko I Botev et al. « The cross-entropy method for optimization ». In : *Handbook of statistics*. T. 31. Elsevier, 2013, p. 35-59.

[Che+17] Tong Che et al. « Maximum-likelihood augmented discrete generative adversarial networks ». In : *arXiv preprint arXiv :1702.07983* (2017).

[Chu+18] Kurtland Chua et al. « Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models ». In : *CoRR* abs/1805.12114 (2018). arXiv : 1805.12114.

[Cla+18] Ignasi Clavera et al. « Model-Based Reinforcement Learning via Meta-Policy Optimization ». In : *CoRR* abs/1809.05214 (2018). arXiv : 1809.05214.

References II

[DR11] Marc Deisenroth et Carl E Rasmussen. « PILCO : A model-based and data-efficient approach to policy search ». In : *Proceedings of the 28th International Conference on machine learning (ICML-11)*. 2011, p. 465-472.

[FAL17] Chelsea Finn, Pieter Abbeel et Sergey Levine. « Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks ». In : *CoRR* abs/1703.03400 (2017). arXiv : 1703.03400.

[Fin+16] Chelsea Finn et al. « A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models ». In : *arXiv preprint arXiv :1611.03852* (2016).

[FLA16] Chelsea Finn, Sergey Levine et Pieter Abbeel. « Guided Cost Learning : Deep Inverse Optimal Control via Policy Optimization ». In : *CoRR* abs/1603.00448 (2016). arXiv : 1603.00448.

[Goo+14] Ian Goodfellow et al. « Generative adversarial nets ». In : *Advances in neural information processing systems*. 2014, p. 2672-2680.

[HE16] Jonathan Ho et Stefano Ermon. « Generative Adversarial Imitation Learning ». In : *CoRR* abs/1606.03476 (2016). arXiv : 1606.03476.

[Hee+15] Nicolas Heess et al. « Learning Continuous Control Policies by Stochastic Value Gradients ». In : *CoRR* abs/1510.09142 (2015). arXiv : 1510.09142.

References III

[Hes+18] Todd Hester et al. « Deep q-learning from demonstrations ». In : *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[HGE16] Jonathan Ho, Jayesh Gupta et Stefano Ermon. « Model-free imitation learning with policy optimization ». In : *International Conference on Machine Learning*. 2016, p. 2760-2769.

[Jac68] David H Jacobson. « New second-order and first-order algorithms for determining optimal control : A differential dynamic programming approach ». In : *Journal of Optimization Theory and Applications* 2.6 (1968), p. 411-440.

[KR04] Malte Kuss et Carl E Rasmussen. « Gaussian processes in reinforcement learning ». In : *Advances in neural information processing systems*. 2004, p. 751-758.

[Kur+18a] Thanard Kurutach et al. « Model-ensemble trust-region policy optimization ». In : *arXiv preprint arXiv :1802.10592* (2018).

[Kur+18b] Thanard Kurutach et al. « Model-Ensemble Trust-Region Policy Optimization ». In : *CoRR* abs/1802.10592 (2018). arXiv : 1802.10592.

[LA14] Sergey Levine et Pieter Abbeel. « Learning neural network policies with guided policy search under unknown dynamics ». In : *Advances in Neural Information Processing Systems*. 2014, p. 1071-1079.

[LK13] Sergey Levine et Vladlen Koltun. « Guided policy search ». In : *International Conference on Machine Learning*. 2013, p. 1-9.

References IV

[LPB17] Balaji Lakshminarayanan, Alexander Pritzel et Charles Blundell. « Simple and scalable predictive uncertainty estimation using deep ensembles ». In : *Advances in Neural Information Processing Systems*. 2017, p. 6402-6413.

[Luo+18] Yuping Luo et al. « Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees ». In : *arXiv preprint arXiv :1807.03858* (2018).

[ML16] William Montgomery et Sergey Levine. « Guided Policy Search as Approximate Mirror Descent ». In : *CoRR* abs/1607.04614 (2016). arXiv : 1607.04614.

[Nag+17] Anusha Nagabandi et al. « Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning ». In : *CoRR* abs/1708.02596 (2017). arXiv : 1708.02596.

[Oza+15] Naoya Ozaki et al. « Differential dynamic programming approach for robust-optimal low-thrust trajectory design considering uncertainty ». In : *25th International Symposium on Space Flight Dynamics, Munich, Germany*. 2015, p. 19-23.

[Raj+17] Aravind Rajeswaran et al. « Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations ». In : *CoRR* abs/1709.10087 (2017). arXiv : 1709.10087.

References V

[Rao09] Anil V Rao. « A survey of numerical methods for optimal control ». In : *Advances in the Astronautical Sciences* 135.1 (2009), p. 497-528.

[Ras03] Carl Edward Rasmussen. « Gaussian processes in machine learning ». In : *Summer School on Machine Learning*. Springer. 2003, p. 63-71.

[RB10] Stéphane Ross et Drew Bagnell. « Efficient reductions for imitation learning ». In : *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, p. 661-668.

[RGB11] Stéphane Ross, Geoffrey Gordon et Drew Bagnell. « A reduction of imitation learning and structured prediction to no-regret online learning ». In : *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011, p. 627-635.

[RW06] Carl Edward Rasmussen et Christopher KI Williams. « Gaussian Processes for Machine Learning ». In : *Gaussian Processes for Machine Learning, by CE Rasmussen and CKI Williams. ISBN-13 978-0-262-18253-9* (2006).

[Sci+20] Thomas Scialom et al. « ColdGANs : Taming Language GANs with Cautious Sampling Strategies ». In : *Advances in Neural Information Processing Systems 33 : Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Sous la dir. d'Hugo Larochelle et al. 2020.

References VI

[Sut91] Richard S Sutton. « Dyna, an integrated architecture for learning, planning, and reacting ». In : *ACM Sigart Bulletin* 2.4 (1991), p. 160-163.

[TET12] Yuval Tassa, Tom Erez et Emanuel Todorov. « Synthesis and stabilization of complex behaviors through online trajectory optimization ». In : *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, p. 4906-4913.

[TL05] Emanuel Todorov et Weiwei Li. « A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems ». In : *Proceedings of the 2005, American Control Conference, 2005.* IEEE. 2005, p. 300-306.

[Wan+19] Tingwu Wang et al. « Benchmarking Model-Based Reinforcement Learning ». In : *CoRR* abs/1907.02057 (2019). arXiv : 1907.02057.

[Zie+08] Brian D Ziebart et al. « Maximum entropy inverse reinforcement learning ». In : (2008).

[Zie10] Brian D Ziebart. « Modeling purposeful adaptive behavior with the principle of maximum causal entropy ». Thèse de doct. figshare, 2010.