Démarche

Mehdi Mounsif

26 février 2018

1 Linear Inverse Reinforcement Learning

Suivant [1] :

- S : states
- A : action
- P_{sa} matrice de transition
- γ facteur discount
- R fonction de récompense

La programmation linéaire permet de découvrir la fonction de récompense par IRL si :

- On dispose de trajectoires d'un expert, dont on a l'assurance qu'il choisit toujours l'action optimale
- On connait la matrice de transition du MDP

Ainsi, avec l'équation de Bellman :

$$V_\pi(s) = R(s) + \sum_{s'} P_{\pi(s)}(s')V_\pi(s')$$

On a :

$$V_\pi = R + \gamma P_{a*} V_\pi(s)$$

D'où :

$$V_\pi = (I - \gamma P_{a*})^{-1} R$$

Puisqu'on sait que l'expert choisit toujours l'action optimale :

$$P_{a*} V_\pi \geq P_a V_\pi \quad \forall a \in A \backslash a*$$

$$P_{a*}(I - \gamma P_{a*})^{-1} R \geq P_a (I - \gamma P_{a*})^{-1} R \quad \forall a \in A \backslash a*$$

$$(P_{a*} - P_a)(I - \gamma P_{a*}) R \geq 0 \quad \forall a \in A \backslash a*$$

L'idée principale est de découvrir une fonction de récompense qui maximise l'écart entre les politiques expert et les autres politiques. Ainsi, pour ce faire, à chaque état on veut maximiser l'écart entre l'expected value de l'action optimale et la plus haute expected value issue des actions suboptimales. Formellement :

$$\text{maximiser} \sum_{i=1}^{N} \min_{a \in A \backslash a*} \{P_{a*}(i) - P_a(i)\}(I - \gamma P_{a*})^{-1} R$$

2 Inverse Reinforcement Learning : Maximum Entropy

L'algorithme précédent, dépendant de la programmation linéaire est limité par ses hypothèses. L'entropie maximum présente des avantages intéressants pour la détermination des solutions optimales. En plus des notations précédentes, on introduit :

- $\zeta = \{(s, a)\}$ une trajectoire
- $f_s \in R^k$ le **feature vector** de l'état s
- $\theta \in R^k$ les paramètres de la fonction de récompense
- $P(\zeta)$ probabilité d'occurence de la trajectoire ζ
- $P(s)$ probabilité de visiter l'état s, nommé **state visitation frequency**
- $P(\zeta) = \prod_{s \in S} P(s)$

On fait les hypothèses suivantes :

- La récompense d'une trajectoire est exprimée linéairement par rapport aux **feature counts**

$$R(\zeta) = \theta^T f_\zeta = \sum_{s \in \zeta} \theta^T f_s$$

- Le principe de l'entropie maximale s'applique : La probabilité des trajectoires telles que celles de l'expert génère des récompenses de grande qualité avec une probabilité exponentiellement plus grande que des récompenses moindres

L'objectif de cette approche est de maximiser la vraisemblance des trajectoires de l'expert.

$$\theta^* = \mathbf{argmax}_\theta L(\theta)$$

$$\theta^* = \mathbf{argmax}_\theta \frac{1}{M} \log P\{\zeta\}|\theta$$

$$\theta^* = \mathbf{argmax}_\theta \frac{1}{M} \log \prod_{s \in S} P(s)$$

$$\theta^* = \mathbf{argmax}_\theta \frac{1}{M} \log \sum_{\zeta} P(\zeta|\theta)$$

$$\theta^* = \mathbf{argmax}_\theta \frac{1}{M} \sum_{\zeta} \log \frac{e^{R(\zeta)}}{Z}$$

$$\theta^* = \mathbf{argmax}_\theta \frac{1}{M} \sum_{\zeta} R(\zeta) - \log(Z)$$

Où M est le nombre de trajectoires disponibles et Z un terme de normalisation :

$$Z = \sum_{\zeta} e^{R(\zeta)}$$

Ainsi, on peut écrire :

$$\theta^* = \mathbf{argmax}_\theta \frac{1}{M} \sum_{\zeta} R(\zeta) - \log \sum_{\zeta} e^{R(\zeta)}$$

$$\theta^* = \mathbf{argmax}_\theta \frac{1}{M} \sum_{\zeta} \theta^T f_\zeta - \log \sum_{\zeta} e^{\theta^T f_\zeta}$$

L'objectif est convexe (?). On dérive pour obtenir les gradients :

$$\nabla_\theta L = \frac{1}{M}\sum_\zeta f_\zeta - \frac{1}{\sum_\zeta e^{R(\zeta)}}\sum_\zeta e^{R(\zeta)}\frac{dR(\zeta)}{d\theta}$$

$$\nabla_\theta L = \frac{1}{M}\sum_\zeta f_\zeta - \sum_\zeta \frac{e^{R(\zeta)}}{\sum_\zeta e^{R(\zeta)}} f_\zeta$$

$$\nabla_\theta L = \frac{1}{M}\sum_\zeta f_\zeta - \sum_\zeta P(\zeta|\theta) f_\zeta$$

Et comme les trajectoires ne sont composés que des états :

$$\nabla_\theta L = \frac{1}{M}\sum_s f_s - \sum_s P(s|\theta) f_s$$

Pour déterminer $P(s|\theta)$, on utilise des approches type Monte-Carlo.

3 Maximum Entropy Deep Inverse Reinforcement Learning

[2]

Une grande partie des travaux précédents repose sur la paramétrisation d'une fonction de récompense basée sur des **features** pré-détérminées. L'utilisation d'approximation de fonctions (ANN) permet de s'abstraire de ce paradigme et de surcroît offre de manière inhérente une meilleure généralisation. Quelques notations de l'article :

- MDP $M = (S, A, \tau, r)$
- $\mathcal{D} = \{\zeta_1, \zeta_2, ..., \zeta_N\}$ les trajectoires d'un expert composée de paires state-action

Dans la notation de MaxEnt, la probabilité de la préférence de l'utilisateur pour une trajectoire est propotionelle à l'exponentielle de la récompense le long du chemin. :

$$P(\zeta|r) = exp\{\sum_{s,a\in\zeta} r_{s,a}\}$$

L'approche MatEnt permet de gérer la sub-optimalité des actions de l'expert. Elle s'apparente à l'inférence Bayesienne, notamment l'estimation du Maximum A Posteriori :

$$\mathcal{L}(\theta) = \log P(D,\theta|r) = \underbrace{\log P(D|r)}_{\mathcal{L}_D} + \underbrace{\log P(\theta)}_{\mathcal{L}_\theta}$$

On a alors :

$$\frac{\delta\mathcal{L}}{d\theta} = \frac{\delta\mathcal{L}_D}{d\theta} + \frac{\delta\mathcal{L}_\theta}{d\theta}$$

On peut écrire :

$$\frac{\delta\mathcal{L}_D}{d\theta} = \frac{\delta\mathcal{L}_D}{dr}\frac{\delta r}{d\theta}$$

$$\frac{\delta\mathcal{L}_D}{d\theta} = (\mu_D - \mathbb{E}[\mu])\frac{\delta}{d\theta}g(f,\theta)$$

Où $r = g(f,\theta)$ et $\mathbb{E}[\mu]) = \sum_\zeta P(\zeta|r)$

Voir algo dans [2]

4 Approches alternatives : Vicarious, HTM, Neurithmic, Capsules

Références

[1] Abbeel, P., and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. 1–.

[2] Wulfmeier, M., Ondruska, P., and Posner, I. Maximum entropy deep inverse reinforcement learning. *CoRR abs/1507.04888* (2015).