Apprenticeship Learning via Inverse Reinforcement Learning

Rodrigo Kalil
EMAp
FGV

Vinícius Antunes
EMAp
FGV
vinicius.sousa@fgv.edu.br

Abstract

Neste trabalho, replicamos o paper intitulado "Apprenticeship Learning via Inverse Reinforcement Learning"Abbeel and Ng [2004], que aborda o problema de aprendizado em um processo de decisão de Markov onde a função de recompensa não é explicitamente fornecida. Em vez disso, um *expert* é observado demonstrando a tarefa que desejamos aprender a realizar. A função de recompensa que o *expert* tenta maximizar uma função de recompensa expressável como uma combinação linear de *features* conhecidas e é desenvolvido um algoritmo para aprender a tarefa demonstrada pelo *expert*.

1 Introdução

Existem diversos algoritmos para encontrar uma política tima em problemas de decisão sequenciais, que são formalmente definidos como processos de decisão de Markov(MDP). Algoritmos assim geralmente pressupõem uma função de recompensa como dada. A politica ótima e a função valor são determinadas com exatidão se a função de recompensa e as probabilidades de transição de estados do MDP são conhecidas. Representar o problema por MDP é útil porque especificar a função de recompensa é mais fácil do que a política ótima ou a função valor diretamente. Contudo, em alguns casos mesmo a função de recompensa não é facilmente definida manualmente.

No paper, é citada como exemplo a tarefa de aprender a dirigir um carro. Um motorista leva em consideração diversos fatores ao dirigir: obedecer às placas, observar o movimento de outros veículos, manter uma velocidade razoável. Para especificar a função de recompensa para essa tarefa, seria necessário considerar pesos que representam a importância e interação entre todos esses fatores, e mesmo sabendo dirigir, não é simples definir uma função recompensa razoável para "dirigir bem". Então um cientista geralmente tentaria ajustar manualmente essa função, algo trabalhoso e possivelmente inaplicável em áreas de pesquisa.

Tendo isso em vista, mesmo no processo humano de aprendizado as pessoas não enumeram em detalhes os diversos fatores que influenciam na decisão. Há um processo de demonstração da atividade por quem já sabe dirigir. Aprender de um especialista é uma tarefa conhecida por *aprendizado tutorado*. Diversas abordagens de aprendizado tutorado buscam imitar o comportamento do especialista, mapeando seus estados para ações. Como o campo de estudos de aprendizado por reforço considera a função recompensa, e não a valor ou a política ótima, como a definição mais sucinta e robusta da tarefa, seria natural tentar uma abordagem pela qual a função de recompensa é aprendida.

Derivar a função de recompensa com base em observações do comportamento de um especialista é conhecido como *aprendizado por reforço inverso*. No paper, assume-se que o especialista tenta otimizar uma função de recompensa que não conhecemos, mas que pode ser expressa pela combinação linear de alguns fatores conhecidos. Não é garantido no paper que o algoritmo vai recuperar perfeitamente a função de recompensa dol.== especialista, mas sim que ele estima uma função que permite desempenhar na tarefa de forma similar a ele.

2 Preliminares

Daremos nessa sessão um *overview* das preliminares do paper.

Um processo de decisão de Markov (MDP) é uma tuple (S, A, T, γ, D, R), onde S é conjunto finito de estados; A é um conjunto de ações; $T = \{P_{sa}\}$ é um conjunto de distribuições de probabilidade de transição de estados (P_{sa} é a distribuição de transição dada a ação a no estado s); $\gamma \in [0, 1)$ é o fator de desconto; D é a distribuição de estado inicial; e $R : S \to A$ é função de recompensa, assumida como tendo valo absoluto menor que 1. MDP\R denota uma MDP sem função de recompensa.

Para encontrar a solução do problema de aprendizado por reforço inverso, Abbeel and Ng [2004] assumem que existe algum vetor de *features* $\phi : S \to [0, 1]^k$ sobre os estados, e que existe alguma função de recompensa verdadeira $R^*(s) = w^* \cdot \phi(s)$, onde $w^* \in \mathbb{R}^k$. Para garantir que as recompensas são menores que 1, é assumido também que $||w||_1 \leq 1$.

Uma *policy* π é um mapeamento de estados para distribuições de probabilidade sobre as ações. O valor de uma *policy* é:

$$\begin{aligned} E_{s_0 \sim D}[V^\pi(s_0)] &= E[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi] \\ &= E[\sum_{t=0}^{\infty} \gamma^t w \cdot \phi(s_t)|\pi] \\ &= w \cdot E[\sum_{t=0}^{\infty} \gamma^t \cdot \phi(s_t)|\pi] \end{aligned}$$

s_0 é o estado inicial, e $s_1, ...$ são estados encontrados ao seguir a *policy* π. É definido a esperança das "features" como $\mu(\pi) = E[\sum_{t=0}^{\infty} \gamma^t \cdot \phi(s_t)|\pi] \in \mathbb{R}^k$. usando essa notação tem-se $E_{s_0 \sim D}[V^\pi(s_0)] = w \cdot \mu(\pi)$.

Para estimar a função de recompensa do *expert* π_E, Abbeel and Ng [2004] assumem que se tem a habilidade de observar sequências de estados (trajetórias) gerado pelo *expert* começando de $s_0 \sim D$ e tomando ações segundo π_E. Não é necessário que π_E seja a *policy* ótima sobre $R^* = w^* \cdot \phi$.

Para o algoritmo proposto por Abbeel and Ng [2004], requere-se uma estimativa de $\mu(\pi_E)$. Especificamente, dado um conjunto de m trajetórias $\{s_0^{(i)}, s_1^{(i)}, ...\}_{1=1}^m$ geradas pelo *expert*, denota-se a estimativa empírica para μ_E por

$$\hat{\mu}_E = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{\infty} \gamma^t \phi(s_t^{(i)}).$$

3 Algoritmo

O problema é definido como o seguinte: Dada uma MDP R, um mapeamento de features ϕ e a esperança das features do especialista μ_E, encontre a política cuja performance é próxima da do especialista sob a função de recompensa desconhecida $R = w^{*T}\phi$. Para isso, achamos a política $\widetilde{\pi}$ tal que $||\mu(\widetilde{\pi}) - \mu_E||_2 \leq \epsilon$. Com tal política, para qualquer $w \in R^k$, sendo $||w||_1 \leq 1$, temos que:

$$\begin{aligned} &|E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi_E\right] - E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\widetilde{\pi}\right] && (1) \\ &= |w^T \mu(\widetilde{\pi} - w^T \pi_E| && (2) \\ &\leq ||w||_2 \cdot ||\widetilde{\pi} - \pi_E||_2 && (3) \\ &\leq 1 \cdot \epsilon = \epsilon && (4) \end{aligned}$$

Assim o problema se reduz a encontrar a política que induz esperanças para as features próximas às do especialista. O algoritmo para aprendizado tutorado proposto para tal tarefa é:

- Pegue aleatoriamente uma política $\pi^{(0)}$, e compute ou aproxime por Monte Carlo; defina $i = 1$.
- Compute $t^{(i)} = max_{w:||w||_2 \leq 1} min_{j \in 0..(i-1)} w^T(\mu_E - \mu^{(j)})$; e faça de $w^{(i)}$ o que alcança esse máximo.
- Se $t^{(i)} \leq \epsilon$, termine.
- Usando o algoritmo RL, compute a política ótima $\pi^{(i)}$ para o MDP usando recompensa $R = (w^{(i)})^T$.
- Compute ou estime $\mu^{(i)} = \mu(\pi^{(i)})$.
- Incremente i; e volte ao passo 2.

Ao terminar, o algoritmo retorna o conjunto de i políticas obtidas.

Na iteração i, i-1 polítias já foram achadas. A otimização no passo 2 pode ser encarada como um passo de aprendixado por reforço inverso, no qual tenta-se adivinhar a recompensa otimizada pelo especialista. A maximização nesse passo é equivalente a:

$$max_{t,w} t \tag{5}$$

$$s.t. w^T \mu_E \geq w^T \mu^{(j)} + t, j = 0, ..., i-1 \tag{6}$$

$$||w||_2 \leq 1 \tag{7}$$

Então, se o algoritmo termina com um t menor que ϵ, temos que: $\forall w$ com $||w||_2 \leq 1 \exists i$ tal que $w^T \mu^{(i)} \geq w^T \mu_E - \epsilon$

O algoritmo tenta, portanto, encontrar uma recompensa tal que $E_{s0\ D}[V^{\pi_E}(s_0)] \geq E_{s0\ D}[V^{\pi^{(i)}}(s_0)] + t$; uma recompensa na qual o especialista desempenha melhor por uma "margem" de t que qualquer política encontrada anteriormente. Por conta da restrição de norma 2, o algoritmo não pode ser colocado como um programa linear, mas somente como um programa quadrático.

Essa otimização é equivalente à usada em *support vector machines*, para achar o hiperplano marginal máximo separando dois conjuntos de pontos. A equivalência ocorre se associamos a classe 1 à esperança das features do especialista e -1 para as esperanças obtidas nas iterações anteriores. Assim, um SVM pode também ser usado para encontrar $w^{(i)}$.

Assim, há pelo menos uma política no conjunto retornado pelo algoritmo cuja performance sob R^* é pelo menos tão boa quanto a performance do especialista menos ϵ. Após o uso do algoritmo, o desenhista do agente pode manualmente examinar as políticas encontradas e selecionar à mão a que performa melhor. O paper afirma que uma extensão desse método garante que no máximo k+1 políticas do conjunto precisam ser analisadas.

Uma alternativa à análise humana do conjunto resposta é achar o ponto mais próximo de política do especialista resolvendo: $min||\mu_E - \mu||_2$ tal que $\mu = \sum_i \lambda_i \mu^{(i)}$, $\lambda_i \geq 0$, $\sum_i \lambda_i = 1$.

3.1 Um algoritmo mais simples

Abbeel and Ng [2004] propõem uma mudança no algoritmo, que eles chamam de método de projeção, chamando o algoritmo descrito anteriormente, baseada em Programação Quadrática, de método max margin. Basicamente, o método de projeção substitui o segundo passo do algorítimo com o seguinte:

- Faça $\overline{\mu}^{(i-1)} = \overline{\mu}^{(i-2)} + \frac{(\mu^{(i-1)} - \overline{\mu}^{(i-2)})^T(\mu_E - \overline{\mu}^{(i-2)})}{(\mu^{(i-1)} - \overline{\mu}^{(i-2)})^T(\mu^{(i-1} - \overline{\mu}^{(i-2)})}(\mu^{(i-1} - \overline{\mu}^{(i-2)})$
- Faça $w^{(i)} = \mu_E - \overline{\mu}^{(i-1)}$
- Faça $t^{(i)} = ||\mu_E - \overline{\mu}^{(i-1)}||_2$

Na primeira iteração, $w^{(1)} = \mu_E = \mu^{(0)}$ e $\overline{\mu}^{(0)} = \mu^{(0)}$.

Abbeel and Ng [2004] dão a justificativa esse método no paper completo, que eles citam no paper que temos acesso, mas não conseguimos achar esse paper completo, visto que o site deles está fora do ar. Mesmo não tendo a justificativa, ainda temos resultado de convergência desse algoritmo e testamos ele nos nossos experimentos.

4 Resultados teóricos

Na seção anterior, diversas conclusões foram feitas sob a suposição de que o algoritmo termina com $t \leq \epsilon$. No paper é apresentado um teorema que mostra que o algoritmo desenvolvido converge em um número pequeno de iterações. Ele se baseia no seguinte lema:

4.1 Lema

Seja dado um MDP\R, com características $\phi : S \to [0,1]^k$, e um conjunto de políticas Π, $\mu^{(i)} \in M$. Seja $\pi^{(i+1)}$ a política ótima para o MDP\R aumentada com a recompensa $R(s) = (\hat{\mu}_E - \mu^{(i)}) \cdot \phi(s)$, ou seja, $\pi^{(i+1)} = \arg\max_\pi (\hat{\mu}_E - \mu^{(i)}) \cdot \mu(\pi)$. Defina $\tilde{\mu}^{(i+1)} = \frac{(\hat{\mu}_E - \mu^{(i)}) \cdot (\mu^{(i+1)} - \mu^{(i)})}{\|\mu^{(i+1)} - \mu^{(i)}\|_2^2} (\mu^{(i+1)} - \mu^{(i)}) + \mu^{(i)}$, ou seja, a projeção de $\hat{\mu}_E$ na linha através de $\mu^{(i+1)} = \mu(\pi^{(i+1)})$, $\mu^{(i)}$. Então,

$$\frac{\|\hat{\mu}_E - \tilde{\mu}^{(i+1)}\|_2}{\|\hat{\mu}_E - \mu^{(i)}\|_2} \leq \frac{k\sqrt{k^2 + (1-\gamma)^2 \|\hat{\mu}_E - \mu^{(i)}\|_2^2}}{\|\hat{\mu}_E - \mu^{(i)}\|_2}$$

e o ponto $\tilde{\mu}^{(i+1)}$ é uma combinação convexa de $\mu^{(i)}$ e $\mu^{(i+1)}$.

4.2 Teorema 1

Dado um MDP\R, características $\phi : S \to [0,1]^k$, e qualquer $\epsilon > 0$, o algoritmo de aprendizagem por tutoria (tanto versões de margem máxima quanto projeção) termina com $t(i) \leq \epsilon$ após no máximo

$$n = O\left(\frac{k}{(1-\gamma)^2 \epsilon^2} \log \frac{k}{(1-\gamma)\epsilon}\right)$$

iterações.

Dado um ponto $\mu^{(i)}$, o Lema 3 fornece uma maneira de construir um ponto $\tilde{\mu}^{(i+1)} \in M^{(i+1)}$ com uma distância para $\hat{\mu}_E$ que é menor por um fator dado pela equação. Enquanto para o iterado atual $\mu^{(i)}$ tivermos $\|\hat{\mu}_E - \mu^{(i)}\|_2 \geq \epsilon$, temos

$$\frac{\|\hat{\mu}_E - \tilde{\mu}^{(i+1)}\|_2}{\|\hat{\mu}_E - \mu^{(i)}\|_2} \leq \frac{k\sqrt{k^2 + (1-\gamma)^2 \epsilon^2}}{\|\hat{\mu}_E - \mu^{(i)}\|_2}.$$

O algoritmo de margem máxima define $\mu^{(i+1)} = \arg\min_{\mu \in M^{(i+1)}} \|\hat{\mu}_E - \mu\|_2$. Portanto, temos

$$\frac{t(i+1)}{t(i)} \leq \frac{k\sqrt{k^2 + (1-\gamma)^2 \epsilon^2}}{\|\hat{\mu}_E - \mu^{(i)}\|_2}.$$

O algoritmo de projeção define $\mu^{(i+1)} = \tilde{\mu}^{(i+1)}$, mantendo o controle de um único ponto (que é uma combinação convexa de pontos obtidos anteriormente), para o qual a distância $t(\cdot)$ é reduzida em cada iteração pelo mesmo fator:

$$\frac{t(i+1)}{t(i)} \leq \frac{k\sqrt{k^2 + (1-\gamma)^2 \epsilon^2}}{\|\hat{\mu}_E - \mu^{(i)}\|_2}.$$

Como a distância máxima em M é $\sqrt{k}/(1-\gamma)$, temos

$$t(i) \leq \left(\frac{\sqrt{k}\sqrt{(1-\gamma)^2 \epsilon^2 + k}}{(1-\gamma)\epsilon}\right)^i \frac{\sqrt{k}}{1-\gamma}.$$

Assim, temos $t(i) \leq \epsilon$ se

$$i \geq \log\left(\frac{\sqrt{k}}{(1-\gamma)\epsilon}\right) / \log\left(\frac{\sqrt{(1-\gamma)^2\epsilon^2+k}}{\sqrt{k}}\right) = O\left(\frac{k}{(1-\gamma)^2\epsilon^2}\log\frac{k}{(1-\gamma)\epsilon}\right).$$

4.3 Discussão

A descrição do teorema 1 no artigo não parece correta. Do ponto em que ele define a razão entre o t de duas iterações subsequentes e quando ele generaliza um **upper-bound** para determinado t, o conteúdo da fração envolvendo k e γ muda, sem que ele dêexplicações.

Não conseguimos concluir se isso se deve à omissão de parte do desenvolvimento, ou se houve erro na escrita.

5 Experimentos

O código e os resultados para os experimentos podem ser encontrados no repositório do Github

Para testar os algoritmos propostos no paper, utilizamos o jogo *Frozen Lake*, onde temos um agente em um tabuleiro 4x4, podendo se mover para cima, para baixo e para os lados uma casa por vez e o objetivo final é chegar no presente no canto inferior direito sem cair em nenhum dos buracos (círculos azuis no mapa). O jogo acaba quando o agente cai em um dos buracos ou quando ele chega no presente.

A função de recompensa "real" R^* consiste no agente ganhar 1 ponto se chegar no presente e perder um ponto se cair em um buraco. $R^* = w^T\phi, \phi \in \{0,1\}^{16}, w \in \{-1,0,1\}^{16}$, onde ϕ representa a posição do agente, sendo 1 na posição do agente e 0 no resto, e w representa a recompensa de cada posição.

Utilizamos o algoritmo de aprendizado de reforço *Q-Learning* para encontrar as *policies*. Para agilizar o treinamento, adicionamos uma heurística para nossos agentes onde eles não podem tentar se mover para fora do tabuleiro, visto que essa ação só leva o agente a ficar parado, fazendo que os jogos acabem levando bem mais tempo para terminar.

Fizemos 2 experimentos, um onde o agente sempre se move na direção pretendida, e outro onde sempre existe 30% de probabilidade do agente se mover aleatoriamente.

Foram utilizados ambos o método de *max margin* e o método de projeção para estimar os pesos w da função de recompensa. O método de max-margin foi implementado usando a biblioteca python **cvxpy**.

Em ambos os experimentos, o *expert* jogou *Frozen lake* 10.000 vezes para aprender a *policy* ótima e a esperança das *features* foi estimada observando o *expert* jogando 10.000 jogos, e a cada iteração do algoritmo de *apprenticeship learning*, os *apprentices* jogaram *Frozen lake* 10.000 para aprender suas *policies* e 1.000 para estimar a esperança de *features*. O fator de desconto γ utilizado para calcular as esperanças de features foi 0,99.

Como podemos ver na Figura 3, no jogo com movimento determinístico, com ambos métodos, foi possível criar agentes que replicassem a política do *expert*. Mesmo assim, as recompensas aprendidas tanto com o método *max-margin* e o método de projeção se diferem bastante das recompensas verdadeiras, mas o *max-margin* conseguiu identificar corretamente onde está o objetivo final (a posição do presente está com recompensa positiva alta), e esse foi um resultado consistente toda vez que executamos esse experimento.

Já com a regra adicional no jogo que o agente tem 30% de probabilidade de se mover aleatoriamente, com nenhum dos dois métodos conseguimos convergir para a *policy* do *expert*. Interessantemente, o método de *max margin* conseguiu novamente identificar o objetivo final, mas esse não foi um resultado consistente nas várias vezes que executamos esse experimento.

O fato de não termos conseguido aproximar a função de recompensa em nenhum dos experimentos, pode ser explicado pelo fato que, mesmo tendo uma etapa de aprendizado por reforço inverso, o algoritmo de *apprentice learning* de Abbeel and Ng [2004] está de fato aproximando as esperança das *features*, não tendo garantias de aproximação da função de recompensa aprendida a função real.

Figure 1: Frozen lake environment.

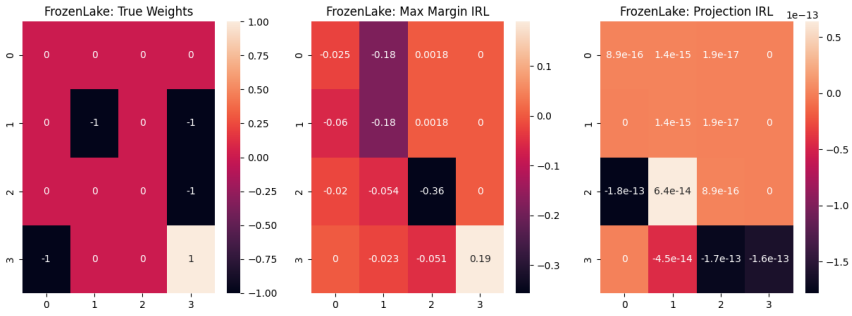

Figure 2: Recompensas estimadas

Não conseguimos replicar os experimentos em Abbeel and Ng [2004] porque os ambientes não estão totalmente especificados no paper. No ambiente *gridworld*, não está especificado qual a condição de parada do jogo nem a distribuição dos estados iniciais. O segundo experimento do paper, uma simulação de direção de carro, também não está suficientemente especificada para a replicação do ambiente.

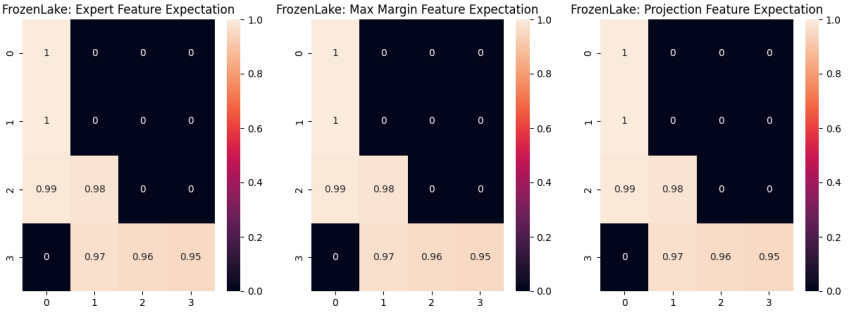

Figure 3: Esperança das *features*

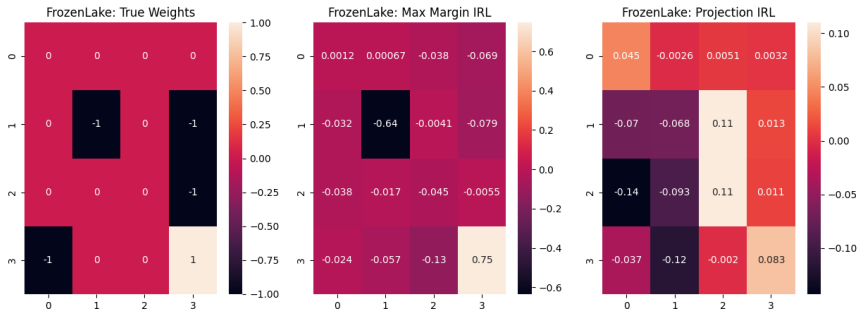

Figure 4: Recompensas estimadas, com movimentos aleatórios

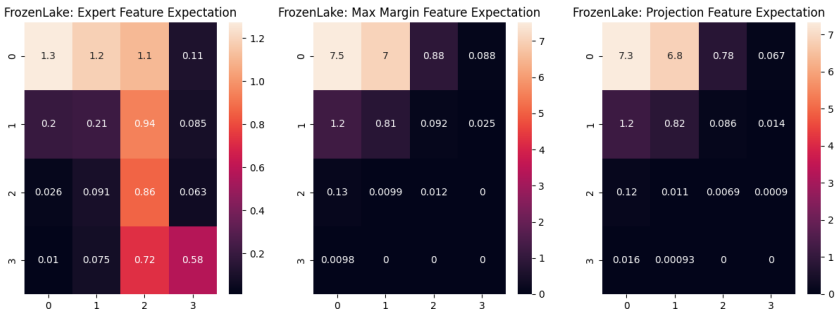

Figure 5: Esperança das *features*, com movimentos aleatórios

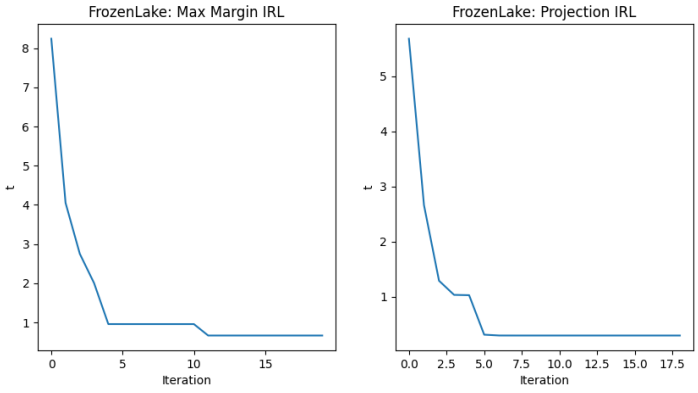

Figure 6: Diferença das esperanças de *features* entre *expert* e *apprentices*, com movimentos aleatórios

6 Discussão

Em *A survey of inverse reinforcement learning*, Adams et al. [2022] apontam algumas limitações do algoritmo de Abbeel and Ng [2004]. Syed and Schapire [2007] levantam que os *apprentices* estão limitados pela qualidade do *expert*. Se a *policy* do *expert* é sub-ótima, a *policy* aprendida também vai ser sub-ótima. Outra limitação do algoritmo de Abbeel and Ng [2004] é a dependência nas demonstrações do *expert*. Valko et al. [2013] estendem a formulação do max margin para um caso semi supervisionado onde as trajetórias providas pelo expert são assumidas como sendo exemplos rotulados e trajetórias providas por não *experts* são assumidas como não sendo rotuladas.

Richad Hu [2021] implementam o algoritmo de Abbeel and Ng [2004] no ambiente do gymnasium cartpole (um poste está amarrado a um carrinho que pode se mover pra esquerda e para direita e o objetivo é manter o poste de pé), usando o método de projeção. Eles mostram que o *apprentices* conseguem replicar o desempenho do *expert*. Esse resultado tem um fator interessante pelo fato que a função de recompensa utilizada pelo *expert* é tempo que o poste fica em pé, enquanto as *features* observadas é a posição do carrinho, velocidade, angulo do poste e velocidade angular do poste. Mesmo não utilizando a *feature* utilizada pela função de recompensa, foi possível aproximar os *apprentices* ao *expert*.

References

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

Stephen Adams, Tyler Cody, and Peter A Beling. A survey of inverse reinforcement learning. *Artificial Intelligence Review*, 55(6):4307–4346, 2022.

Omar Ismail Richad Hu, Daniel Dworakowsky. Apprenticeship learning with inverse reinforcement learning implementation, 2021. URL `https://github.com/rhklite/apprenticeship_inverse_RL?tab=readme-ov-file#readme`.

Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL `https://proceedings.neurips.cc/paper_files/paper/2007/file/ca3ec598002d2e7662e2ef4bdd58278b-Paper.pdf`.

Michal Valko, Mohammad Ghavamzadeh, and Alessandro Lazaric. Semi-supervised apprenticeship learning. In Marc Peter Deisenroth, Csaba Szepesvári, and Jan Peters, editors, *Proceedings of the Tenth European Workshop on Reinforcement Learning*, volume 24 of *Proceedings of Machine Learning Research*, pages 131–142, Edinburgh, Scotland, 30 Jun–01 Jul 2013. PMLR. URL `https://proceedings.mlr.press/v24/valko12a.html`.