WDS

DataScience@Web

THIRD-PERSON IMITATION LEARNING

Ren Yanjie

2021.08.05

2021 WDS summer seminar

Content

- Part 1： Introduction
- Part 2： GAN and GAIL
- Part 3： Third-Person Imitation Learning
- Part 4： Conclusion

Part 1: Introduction

- 1.1 Imitation Learning Concepts
- 1.2 Imitation Learning Methods

1.1 Imation Learning Concepts

- Motivation:
  - No Reward Function Available
- Setting:
  - Input: Expert Demonstration $\{\hat{\tau}_1, \hat{\tau}_2, \cdots, \hat{\tau}_K\}$ $\quad \hat{\tau}_i =< s_1^i, a_1^i, s_2^i, a_2^i, \ldots, s_{n_n i+1}^i >$
  - State Function $p(\cdot|s_t, a_t)$ is available
- Goal
  - To Get A Policy Function π(a|s)
- Main Methods:
  - Behaviour Cloning (BC)
  - Inverse Reinforcement Learning (IRL)
  - **Generative Adversarial Imitation Learning (GAIL)**

1.2 Imation Learning Methods

- Behaviour cloning
  - $\hat{\tau} = \{(s_1 \hat{a}_1 s_2, \hat{a}_2 \ldots\}$ ⇒ $\{(s_1, \hat{a}_1),\ (s_2, \hat{a}_2),\ \cdots\}$ (input label)
  - Pros: instuitive, efficient, *off-line*
  - Cons: limit of train data, error-accumulate(step-data), never better than expert
  - Used to initialize model
- Inverse Reinforcement Learning
  - Train The Reward Function ⇒ Reinforcenment Learning
  - Pros: solved error-accumulate problem
  - Cons: **time-consuming**, liner reward function restriction

Part 2: GAN and GAIL

- 2.1 GAN
- 2.2 GAIL
- 2.3 Pros And Cons

2.1 GAN

Image Generation

Database

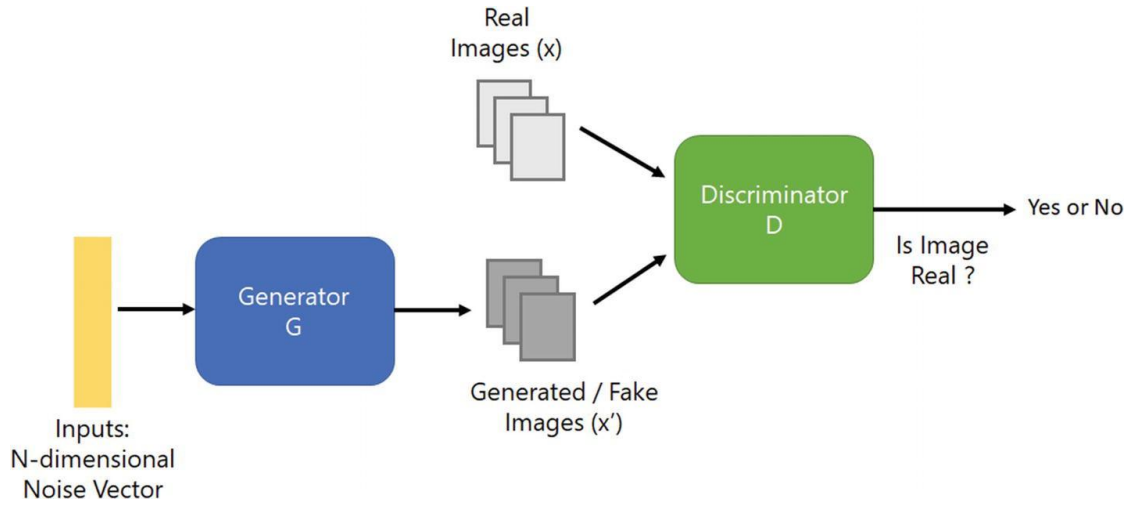
Discriminator

$$\begin{bmatrix} 0.3 \\ -0.1 \\ \vdots \\ -0.7 \end{bmatrix} \begin{bmatrix} 0.1 \\ -0.1 \\ \vdots \\ 0.7 \end{bmatrix} \begin{bmatrix} -0.3 \\ 0.1 \\ \vdots \\ 0.9 \end{bmatrix}$$

In a specific range

NN Generator

2.1 GAN



- Basic Concepts:
  - Generator G(s,θ)
    - s: input vector in which e ~N(-1, 1) or N(0,1)
    - θ: neural network argument
    - $G(s_0, \theta)$: depends on the task
  - Discriminator D(x,φ)
    - x: the output of generator or the sample data
    - φ: the neural network argument
    - D(x,φ): a scalar which denotes the probability of sample data

2.1 GAN

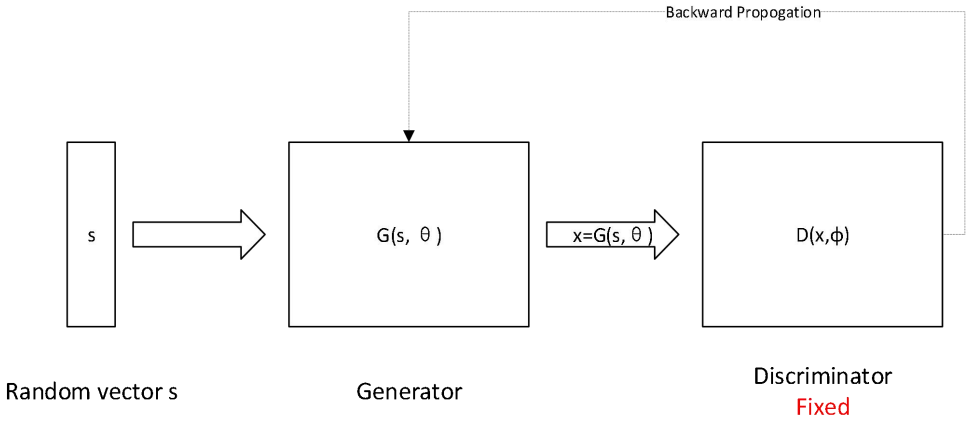
2.1 GAN

- Train the ***generator***
  - Connect the generator G(s,θ) and discriminator D(x,φ)
  - Fix the discriminator parameter φ
  - Define the loss function with cross entropy

$$E(s;\boldsymbol{\theta}) = \ln\left[1 - \underbrace{D(x;\boldsymbol{\phi})}_{\text{越大越好}}\right]; \qquad \text{s.t. } x = G(s;\boldsymbol{\theta}).$$

  - Update the θ with SGD or B-SGD

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \cdot \nabla_{\boldsymbol{\theta}} E(s;\boldsymbol{\theta}).$$



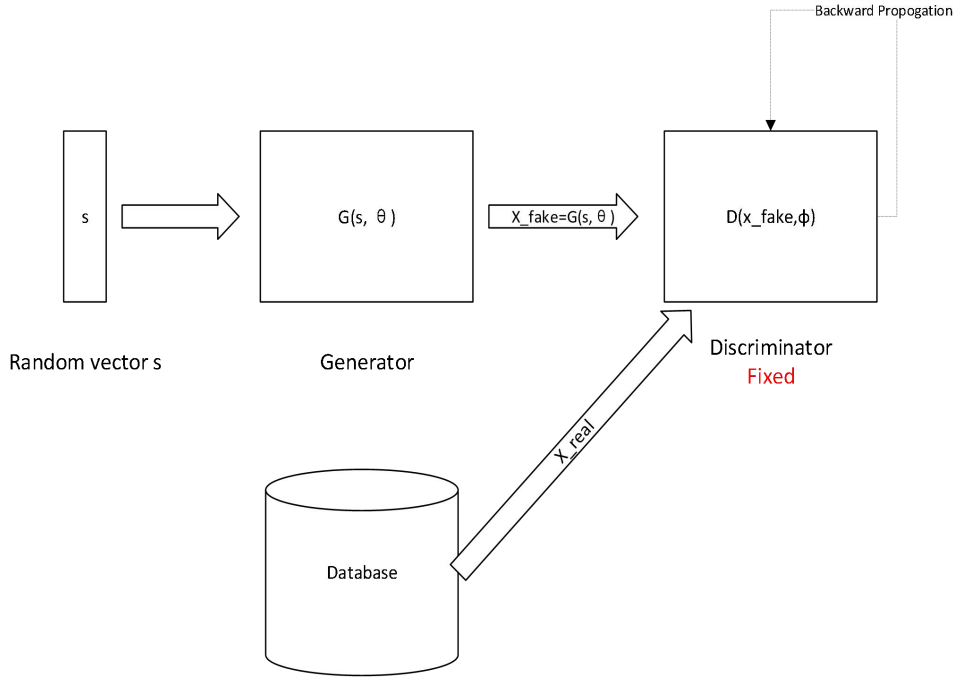
2.1 GAN

- Train the ***discriminator***
  - Take a random sample from the database x^{real}
  - Get the $x^{fake} = G(s;\theta)$
  - Define the loss function with cross entropy

$$F\left(x^{\text{real}}, x^{\text{fake}}; \phi\right) = \ln\left[1 - \underbrace{D\left(x^{\text{real}}; \phi\right)}_{\text{越大越好}}\right] + \ln \underbrace{D\left(x^{\text{fake}}; \phi\right)}_{\text{越小越好}}.$$

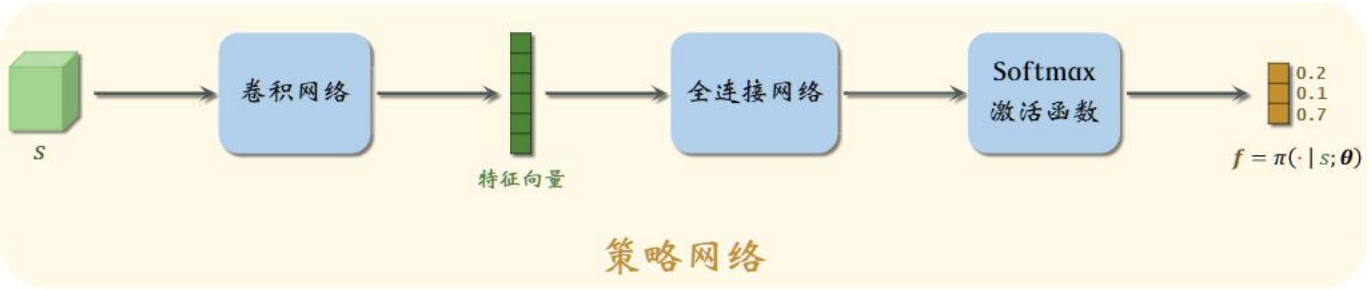
  - Update the φ with SGD or Mini-Batch SGD

$$\phi \leftarrow \phi - \eta \cdot \nabla_{\phi} F\left(x^{\text{real}}, x^{\text{fake}}; \phi\right).$$



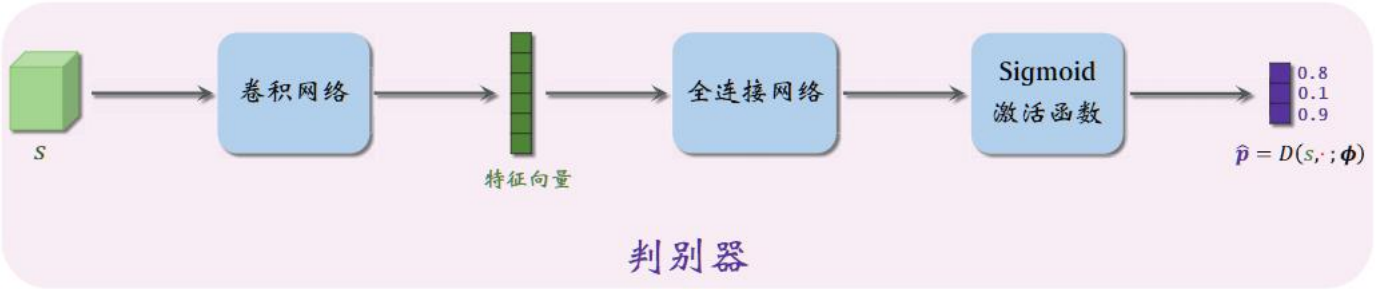
2.2 GAIL

- Expert Demonstration (Database)
  - $\tau = [s_1, a_1, s_2, a_2, \cdots, s_m, a_m].$
  - $\mathcal{X} = \{\tau^{(1)}, \tau^{(2)}, \cdots, \tau^{(k)}\}.$
- Generator
  - $\pi(a|s;\theta) \quad \cdots\cdots\cdots\cdots \quad G(s;\theta)$
  - output $f = \pi(\cdot \mid s;\theta).$
  - given initial state s_1 , generator will get a trajectory $\tau = [s_1, a_1, s_2, a_2, \cdots, s_n, a_n].$



2.2 GAIL

- Discriminator
  - $\widehat{\boldsymbol{p}} = D(s, \cdot \,|\, \boldsymbol{\phi}).$
  - $\widehat{p}_a = D\big(s, a; \boldsymbol{\phi}\big) \in (0,1), \qquad \forall\, a \in \mathcal{A}.$



2.2 GAIL: Algorithm

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**

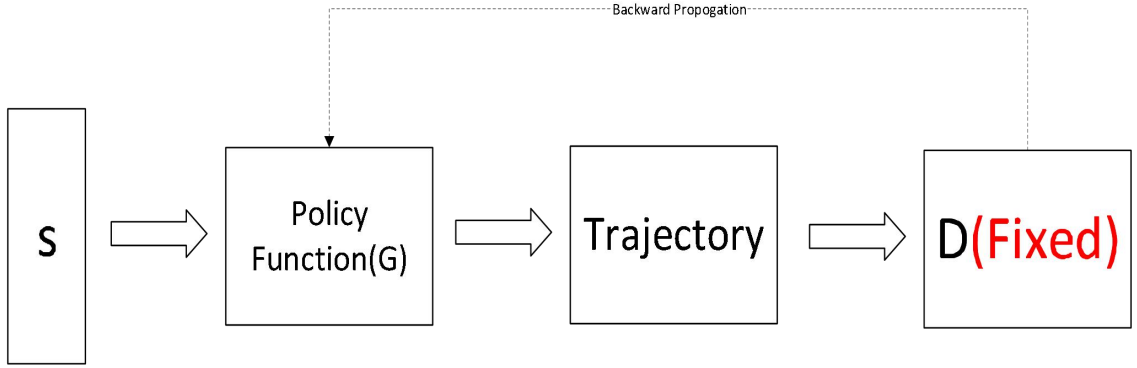
2.2 GAIL: Algorithm

- Train the generator (i.e. policy function $\pi(a|s;\boldsymbol{\theta}_{\text{now}})$)
  - θ_{now}: current parameter of policy function
  - use current policy function to get a trajectory $\tau = [s_1, a_1, s_2, a_2, \cdots, s_n, a_n]$.
  - $D(s_t, a_t; \varphi)$ can output the quality of (s_t, a_t)

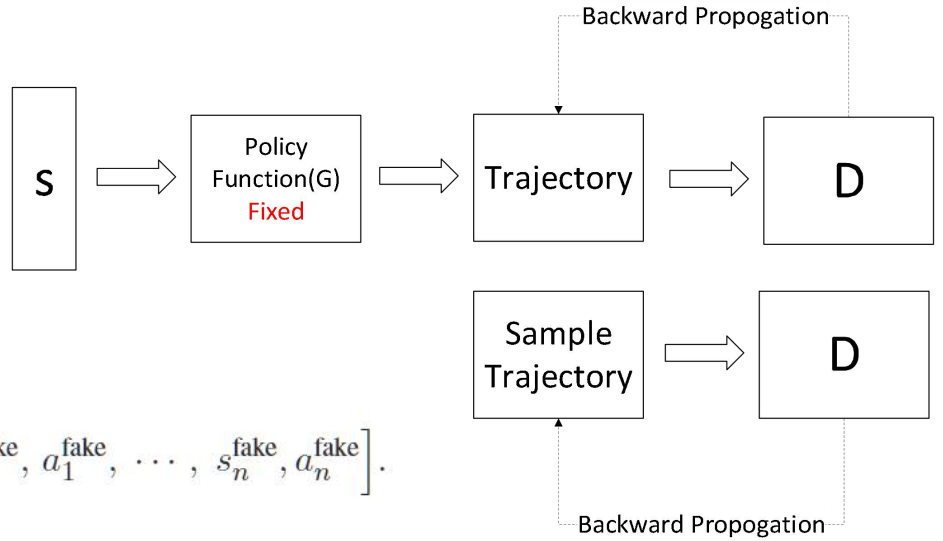
$$u_t = \ln D(s_t, a_t; \phi)$$

$$s_1, a_1, u_1, \quad s_2, a_2, u_2, \quad \cdots, \quad s_n, a_n, u_n.$$

$$\tilde{L}(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_{\text{now}}) \triangleq \frac{1}{n}\sum_{t=1}^{n} \frac{\pi(a_t|s_t;\boldsymbol{\theta})}{\pi(a_t|s_t;\boldsymbol{\theta}_{\text{now}})} \cdot u_t.$$



2.2 GAIL: Algorithm



- Train the discriminator (i.e. policy funct$\pi(a|s;\boldsymbol{\theta}_{\text{now}})$)
  - Random sample $\tau^{\text{real}} = \left[s_1^{\text{real}}, a_1^{\text{real}}, \cdots, s_m^{\text{real}}, a_m^{\text{real}}\right]$.
  - Use current policy function to get a trajectory $\tau^{\text{fake}} = \left[s_1^{\text{fake}}, a_1^{\text{fake}}, \cdots, s_n^{\text{fake}}, a_n^{\text{fake}}\right]$.
  - Define the loss function

$$F\left(\tau^{\text{real}}, \tau^{\text{fake}}; \boldsymbol{\phi}\right) = \underbrace{\frac{1}{m}\sum_{t=1}^{m} \ln\left[1 - D\left(s_t^{\text{real}}, a_t^{\text{real}}; \boldsymbol{\phi}\right)\right]}_{D \text{ 的输出越大，这一项越小}} + \underbrace{\frac{1}{n}\sum_{t=1}^{n} \ln D\left(s_t^{\text{fake}}, a_t^{\text{fake}}; \boldsymbol{\phi}\right)}_{D \text{ 的输出越小，这一项越小}}.$$

2.3 GAIL: Pros and Cons

- Pros
  - Efficient Compared to IRL
  - Exact more features than IRL
- Cons
  - Modal collapse
    - Improved GAIL based on multiple modal
    - Improved GAIL based on
  - Low utilization of generated samples

Part 3: Third-Person Imitation Learning

- 3.1：Basic Concepts
- 3.2：DANN
- 3.3：TPIL

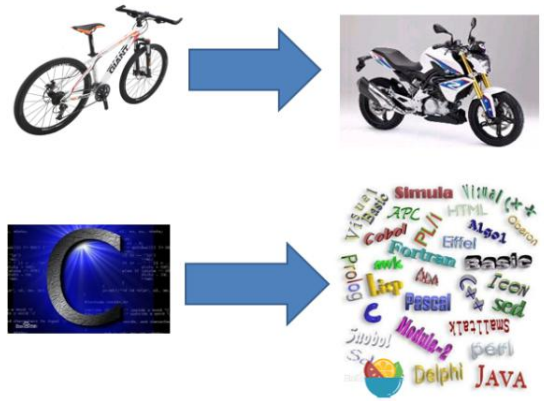
3.1 Third-Person

First Person

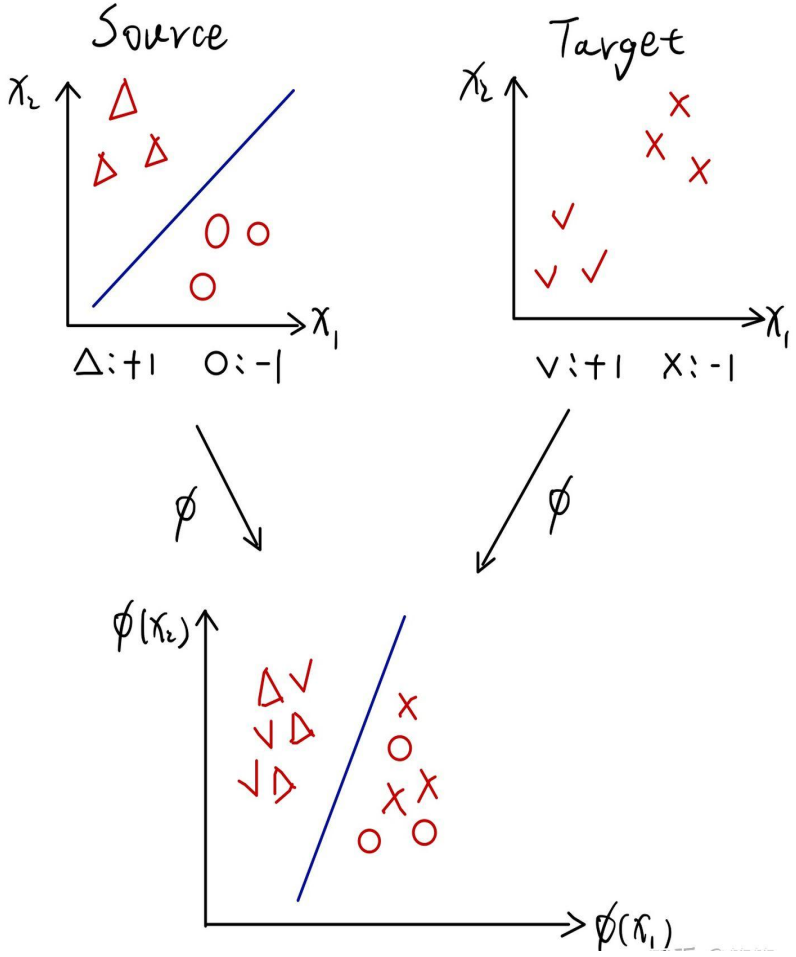
Third Person

3.1 TPIL VS Transfer Learning

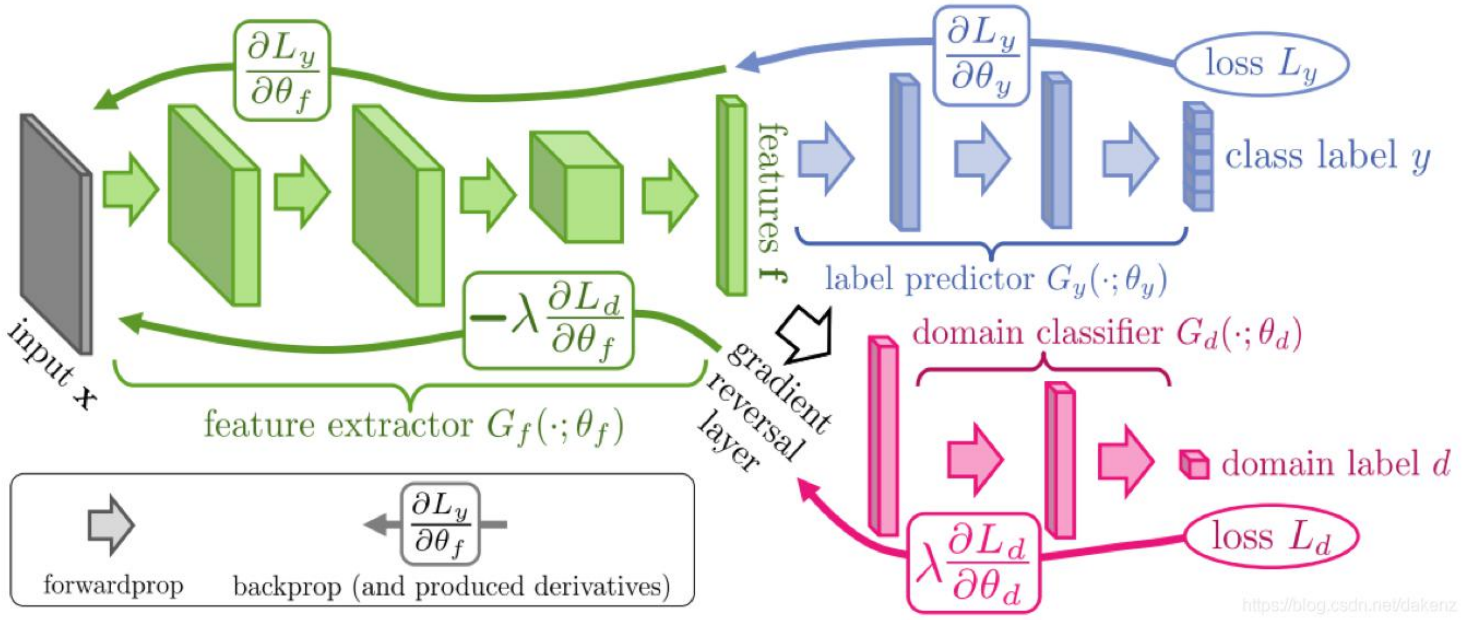
- Transfer Learning: Ability of a system to recognize and apply knowledge and skills learned in previous domains/tasks to novel domains/tasks
- Domain
- TPIL and Tranfer Learning
  - Source Domain: Third person
  - Destion Domin: First Person



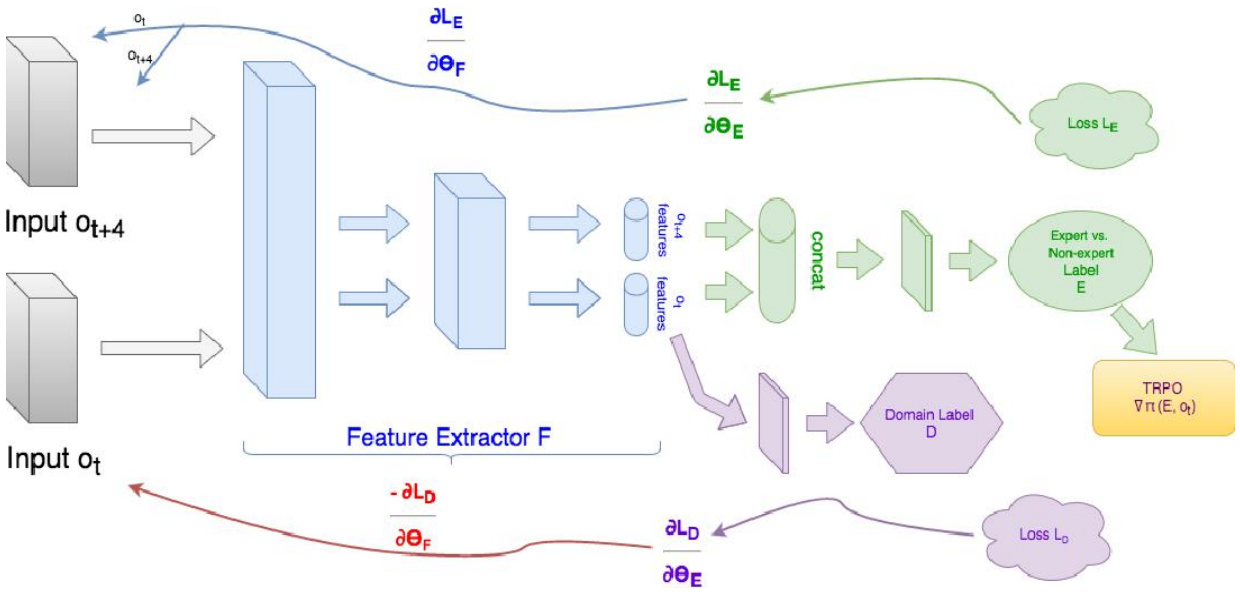
3.2 DANN



3.2 DANN


$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features $\mathbf{f}$
label predictor $G_y(\cdot;\theta_y)$
$-\lambda\frac{\partial L_d}{\partial \theta_f}$
input $\boldsymbol{x}$
gradient reversal layer
feature extractor $G_f(\cdot;\theta_f)$
domain classifier $G_d(\cdot;\theta_d)$
domain label d
loss L_d
$\lambda\frac{\partial L_d}{\partial \theta_d}$
forwardprop
backprop (and produced derivatives)

3.3 TPIL


Input o_{t+4}
Input o_t
Feature Extractor F
o_{t+4} features
o_t features
concat
Expert vs. Non-expert Label E
Domain Label D
TRPO $\nabla\pi(E, o_t)$
Loss L_E
Loss L_D
$\frac{\partial L_E}{\partial \Theta_F}$
$\frac{\partial L_E}{\partial \Theta_E}$
$\frac{-\partial L_D}{\partial \Theta_F}$
$\frac{\partial L_D}{\partial \Theta_E}$

3.3 TPIL

Algorithm 1 A third-person imitation learning algorithm.

Let CE be the standard cross entropy loss.
Let $\mathcal{G}$ be a function that flips the gradient sign during backpropogation and acts as the identity map otherwise.
Initialize two domains, E and N for the expert and novice.
Initialize a memory bank Ω of expert success and of failure in domain E. Each trajectory $\omega \in \Omega$ comprises a rollout of images $o = o_1, \ldots, o_t, \ldots o_n$, a class label c_ℓ, and a domain label d_ℓ.
Initialize $\mathcal{D} = \mathcal{D}_F, \mathcal{D}_R, \mathcal{D}_D$, a domain invariant discriminator.
Initialize a novice policy π_θ.
Initialize numiters, the number of inner policy optimization iterations we wish to run.
for iter in numiters **do**
  Sample a set of successes and failures ω_E from Ω.
  Collect on policy samples ω_N
  Set $\omega = \omega_E \cup \omega_N$.
  Shuffle ω
  for o, c_ℓ, d_ℓ in ω **do**
    for o_t in o **do**
      $\sigma_t = \mathcal{D}_F(o_t)$
      $\sigma_{t+4} = \mathcal{D}_F(o_{t+4})$
      $\mathcal{L}_R = CE(\mathcal{D}_R(\sigma_t, \sigma_{t+4}), c_\ell)$
      $\mathcal{L}_d = CE(\mathcal{D}_D(\mathcal{G}(\sigma_t)), d_\ell)$
      $\mathcal{L} = \lambda \cdot \mathcal{L}_d + \mathcal{L}_R$ [1]
      minimize $\mathcal{L}$ with ADAM.
    end for
  end for
  Collect on policy samples ω_N from π_θ.
  for ω in ω_N **do**
    for ω_t in ω **do**
      $\sigma_t = \mathcal{D}_F(o_t)$
      $\sigma_{t+4} = \mathcal{D}_F(o_{t+4})$
      $\hat{c}_\ell = \mathcal{D}_R(\sigma_t, \sigma_{t+4})$
      $r = \hat{c}_\ell[0]$, the probability that o_t, o_{t+4} were generated via expert rollouts.
      Use r to train π_θ with via policy gradients (TRPO).
    end for
  end for
end for
return optimized policy π_θ

Collect on policy samples ω_N from π_θ.
for ω in ω_N **do**
  for ω_t in ω **do**
    $\sigma_t = \mathcal{D}_F(o_t)$
    $\sigma_{t+4} = \mathcal{D}_F(o_{t+4})$
    $\hat{c}_\ell = \mathcal{D}_R(\sigma_t, \sigma_{t+4})$
    $r = \hat{c}_\ell[0]$, the probability that o_t, o_{t+4} were generated via expert rollouts.
    Use r to train π_θ with via policy gradients (TRPO).
  end for
end for
end for
return optimized policy π_θ

3.3 TPIL

Algorithm 1 A third-person imitation learning algorithm.

Let CE be the standard cross entropy loss.
Let $\mathcal{G}$ be a function that flips the gradient sign during backpropogation and acts as the identity map otherwise.
Initialize two domains, E and N for the expert and novice.
Initialize a memory bank Ω of expert success and of failure in domain E. Each trajectory $\omega \in \Omega$ comprises a rollout of images $o = o_1, \ldots, o_t, \ldots o_n$, a class label c_ℓ, and a domain label d_ℓ.
Initialize $\mathcal{D} = \mathcal{D}_F, \mathcal{D}_R, \mathcal{D}_D$, a domain invariant discriminator.
Initialize a novice policy π_θ.
Initialize numiters, the number of inner policy optimization iterations we wish to run.
for iter in numiters **do**
  Sample a set of successes and failures ω_E from Ω.
  Collect on policy samples ω_N
  Set $\omega = \omega_E \cup \omega_N$.
  Shuffle ω
  for o, c_ℓ, d_ℓ in ω **do**
    for o_t in o **do**
      $\sigma_t = \mathcal{D}_F(o_t)$
      $\sigma_{t+4} = \mathcal{D}_F(o_{t+4})$
      $\mathcal{L}_R = CE(\mathcal{D}_R(\sigma_t, \sigma_{t+4}), c_\ell)$
      $\mathcal{L}_d = CE(\mathcal{D}_D(\mathcal{G}(\sigma_t)), d_\ell)$
      $\mathcal{L} = \lambda \cdot \mathcal{L}_d + \mathcal{L}_R$ [1]
      minimize $\mathcal{L}$ with ADAM.
    end for
  end for
  Collect on policy samples ω_N from π_θ.
  for ω in ω_N **do**
    for ω_t in ω **do**
      $\sigma_t = \mathcal{D}_F(o_t)$
      $\sigma_{t+4} = \mathcal{D}_F(o_{t+4})$
      $\hat{c}_\ell = \mathcal{D}_R(\sigma_t, \sigma_{t+4})$
      $r = \hat{c}_\ell[0]$, the probability that o_t, o_{t+4} were generated via expert rollouts.
      Use r to train π_θ with via policy gradients (TRPO).
    end for
  end for
end for
return optimized policy π_θ

```
for iter in numiters do
  Sample a set of successes and failures ω_E from Ω.
  Collect on policy samples ω_N
  Set ω = ω_E ∪ ω_N.
  Shuffle ω
  for o, c_ℓ, d_ℓ in ω do
    for o_t in o do
      σ_t = D_F(o_t)
      σ_{t+4} = D_F(o_{t+4})
      L_R = CE(D_R(σ_t, σ_{t+4}), c_ℓ)
      L_d = CE(D_D(G(σ_t)), d_ℓ)
      L = λ · L_d + L_R [1]
      minimize L with ADAM.
    end for
  end for
```

Part 4: Conclusion

- 4.1: Imation Learning
  - No Reward Function Available
  - Main methods: Behaviour Cloning/Inverse Reinforcement Learning
- 4.2: Generative Adversial Immitation Learning
- 4.3: Third-person Learning
  - Extend of GAIL
  - DANN && TL && InfoGAN
- 4.4: Some open issues: Manually Setting Award Funtion / Techniques Innovation

Thanks!

Reference

[1]林嘉豪,章宗长,姜冲,郝建业.基于生成对抗网络的模仿学习综述[J].计算机学报,2020,43(02):326-351.

[2] Third-person imitation learning via image difference and variational discriminator bottleneck（student abstract version）. JIANG C,ZHANG Z Z,CHEN Z X,et al. Proceedings of the44th AAAI Conference on Artificial Intelligence . 2020

[3] Third-person imitation learning. Stadie B C,Abbeel P,Sutskever I. Proceedings of the 5th International Conference on Learning Representations （ICLR）. 2017

[4] Deep domain confusion:Maximizing for domain invariance. Tzeng E,Hoffman J,Zhang N,et al. . 2014

[5] Generative adversarial imitation learning. Ho J,Ermon S. Neural Information Processing Systems . 2016

[6] Domain-Adversarial Training of Neural Networks

[7] Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks