Chapter 8
Imitation Learning

Zihan Ding

Abstract To alleviate the low sample efficiency problem in deep reinforcement learning, imitation learning, or called apprenticeship learning, is one of the potential approaches, which leverages the expert demonstrations in sequential decision-making process. In order to provide the readers a comprehensive understanding about how to effectively extract information from the demonstration data, we introduce the most important categories in imitation learning, including behavioral cloning, inverse reinforcement learning, imitation learning from observations, probabilistic methods, and other methods. Imitation learning can either be regarded as an initialization or a guidance for training the agent in the scope of reinforcement learning. Combination of imitation learning and reinforcement learning is a promising direction for efficient learning and faster policy optimization in practice.

Keywords Imitation learning · Apprenticeship learning · Demonstration · Reinforcement learning · Behavioral cloning · Inverse reinforcement learning · Generative adversarial networks · Sample efficiency

8.1 Introduction

As we know, reinforcement learning (RL), especially model-free reinforcement learning, suffers from low sample efficiency as discussed in the chapter of present challenges in reinforcement learning (Chap. 7). Hundreds of thousands of examples are usually needed to solve an uncomplicated task with human-level performance. However, humans can learn to solve the tasks with significantly shorter periods of time and a much smaller number of samples. Apart from improving the efficiency of reinforcement learning algorithms themselves through more elaborate algorithm design with mathematical guarantees, we can actually let the agent leverage

Z. Ding (✉)
Imperial College London, London, UK
e-mail: zhding@mail.ustc.edu.cn

H. Dong et al. (eds.), *Deep Reinforcement Learning*,
https://doi.org/10.1007/978-981-15-4095-0_8

additional information resource, like expert demonstrations. The expert demonstrations contain biased choices for the policy with prior knowledge, which can be distilled and transferred into agent policies in reinforcement learning, through a proper learning process. The task of learning from an expert is called **imitation learning (IL)** (also known as apprenticeship learning). Humans and animals are born to learn from mimicking other individuals of the same kind, which inspires the method of imitation learning for an intelligent agent to learn from demonstrations by others. On the other hand, supervised learning is much more efficient in the aspect of data usage compared with reinforcement learning, due to the benefits of labeled data. Therefore, the method of supervised learning can be incorporated into the agent's learning process to improve its efficiency if demonstrations are provided in a labeled format.

In this chapter, we introduce different approaches for learning a policy with demonstrations. The overview of algorithms and methods in imitation learning categories is shown in Fig. 8.1. We will introduce detailed methods of imitation learning in the following sections, summarized in several main categories including (1) behavioral cloning (BC), (2) inverse reinforcement learning (IRL), (3) imitation learning from observations (IfO, or ILFO in some other literature (Sun et al. 2019)), (4) probabilistic methods, and (5) other approaches. BC is the most simple and straightforward way of using the demonstration data in a supervised learning manner, which is widely applied due to its simplicity and is usually regarded as a cornerstone to build more advanced methods on. IRL is useful in applications where it may be difficult to write down an explicit reward function specifying exactly how different desiderata should be traded off. For example, how much attention should be paid on taking care of different reflectors for automatic driving vehicles based on visual observations is hard to specify through reward engineering. IRL is an approach to recover the unknown reward function from the demonstration data and uses it for further reinforcement learning process. The IfO actually solves the drawback of imitation learning that it usually requires actions as labels for the state inputs, which often happens in human imitation learning process. The methods from a probabilistic inference view include using Gaussian mixture regression or Gaussian process regression to represent the demonstration data and therefore guide the action policy, which is a more efficient alternative for deep neural network methods in some cases. There are also other approaches like directly feeding demonstrations into a replay buffer for off-policy reinforcement learning, etc. After introducing basic categories of different imitation learning methods, we will discuss the relationship of imitation learning and reinforcement learning, like applying imitation learning as an initialization of reinforcement learning in order to improve the learning efficiency of reinforcement learning. Finally, we introduce some other specific methods in imitation learning with reinforcement learning, which are either combinations of previous conceptions or outliers of the summarized categories as Fig. 8.1.

The concept of imitation learning can be defined with the apprenticeship learning formalism (Abbeel and Ng 2004): the learner finds a policy π that performs no worse than expert π_E with respect to an unknown reward function $r(s, a)$. We

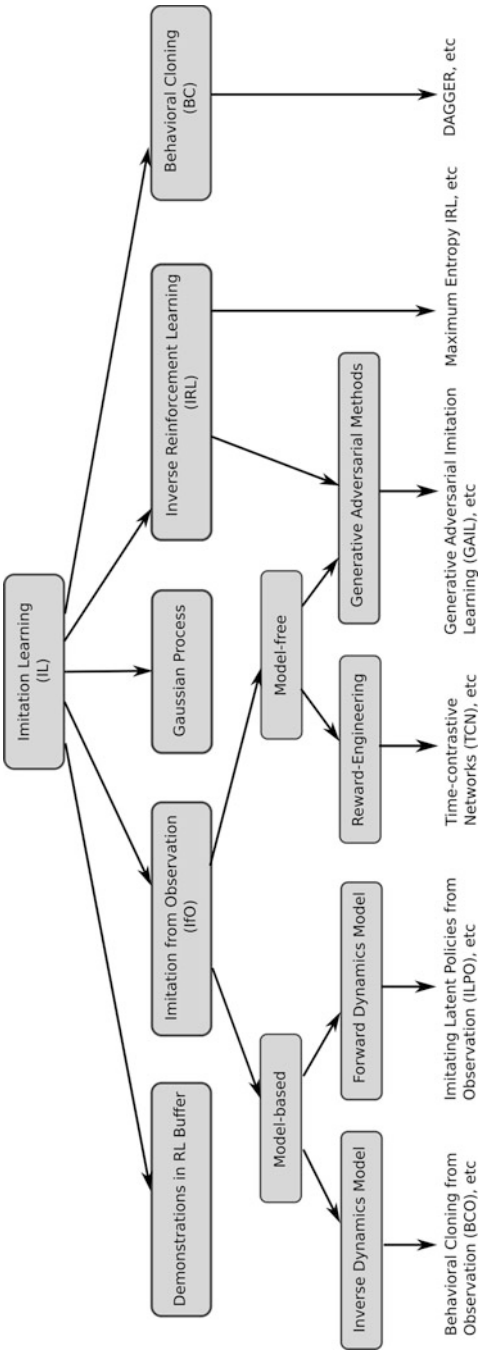

Fig. 8.1 Overview of imitation learning algorithms

define the occupancy measure $\rho_\pi \in \mathcal{D} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ of a policy $\pi \in \Pi$ as: $\rho_\pi(s,a) = \pi(a|s)\sum_{t=0}^{\infty}\gamma^t p(S_t = s|\pi)$ (Puterman 2014), which is a joint distribution of state and action estimated with current policy. Owing to the one-to-one correspondence between Π and $\mathcal{D}$, an imitation learning problem is equivalent to a matching problem between $\rho_\pi(s,a)$ and $\rho_{\pi_E}(s,a)$. A general objective of imitation learning is to learn a policy:

$$\hat{\pi} = \arg\min_{\pi\in\Pi} \psi^*(\rho_\pi - \rho_{\pi_E}) - \lambda H(\pi) \tag{8.1}$$

where ψ^* is a distance measurement between ρ_π and ρ_{π_E}, and $H(\pi)$ is a normalization term with trade-off factor λ. For instance, the normalization term can be defined as the γ-discounted causal entropy of policy π: $H(\pi) \triangleq \mathbb{E}_\pi[-\log\pi(s,a)]$. The overall goal of imitation learning is to increase the similarity of the distribution of $\{(s,a)\}$ samples from current policy and the distribution of those in demonstration dataset, with respect to the some constraints on policy parameters.

8.2 Behavioral Cloning: Supervised Learning Approach

The imitation learning with demonstrations can be naturally regarded as a supervised learning task, if the demonstration data is provided with labels (e.g., a good action for the state can be regarded as a label). In reinforcement learning circumstances, the labeled demonstration data $\mathcal{D}$ usually contains the pairs of state and action as: $\mathcal{D} = \{(s_i, a_i)|i = 1, \ldots, N\}$, where N is the size of the demonstration dataset and index i indicates the s_i and a_i are at the same time step. The state-action pairs can be shuffled for training under the MDP assumption, i.e. the optimal action only depends on the current state. Considering an initial policy π_θ parameterized by θ with input state s and output deterministic action $\pi_\theta(s)$ in reinforcement learning settings, we have demonstrations dataset $\mathcal{D} = \{(s_i, a_i)|i = 1, \ldots, N\}$ generated from experts, which could be used to train the policy, with an objective as follows:

$$\min_\theta \sum_{(s_i,a_i)\sim\mathcal{D}} ||a_i - \pi_\theta(s_i)||_2^2 \tag{8.2}$$

The cases with stochastic policies $\pi_\theta(\tilde{a}|s)$ in some specific formats, e.g. Gaussian policy, etc., can be handled as well using the reparameterization trick:

$$\min_\theta \sum_{\tilde{a}_i\sim\pi(\cdot|s_i),(s_i,a_i)\sim\mathcal{D}} ||a_i - \tilde{a}_i||_2^2 \tag{8.3}$$

This supervised learning approach to directly imitate the expert demonstration is called the **behavioral cloning (BC)** in the literature.

8.2.1 Challenges of BC

- **Covariate Shift**: Although imitation learning can provide a relatively good performance for the cases similar to samples in the demonstration dataset (used for policy training), it could still suffer from bad generalization for samples it never meets during training, as the demonstration dataset only contains finite samples. For example, the new samples during testing can be around another cluster in distribution rather than the same one for training if it is a multimodal distribution, like applying the classifier for cats on distinguishing dogs in practice. As the BC approach boils down the decision-making problem to a supervised learning problem, this well-known problem of covariate shift (Ross and Bagnell 2010) in machine learning can potentially make the learned policy brittle, which is challenging in BC methods. Figure 8.2 further elaborates the covariate shift in BC.
- **Compounding Errors**: BC suffers greatly from the compounding error, a situation where minor errors are compounded over time and finally induce a dramatically different state distribution (Ross et al. 2011). The MDP property of reinforcement learning tasks is the key factor leading to the compounding error, i.e. the amplification effect of consecutive errors. The main reason of the error for each time step can actually be caused by the covariance shift described above, in BC methods. Figure 8.3 shows the compounding errors.

8.2.2 Dataset Aggregation

Dataset Aggregation (DAgger) (Ross et al. 2011) is a more advanced no-regret iterative algorithm for imitation learning from demonstrations following the approach of BC. It proactively chooses demonstration samples that the policy has larger chances to meet afterwards, according to the previous training iterations,

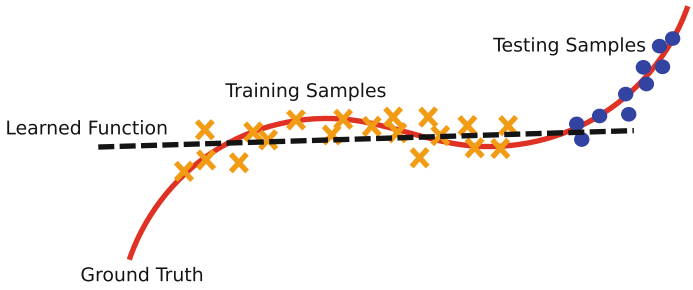

Fig. 8.2 The covariate shift: the learned function (black dashed line) fits well on the training samples (orange "cross"), but has large prediction bias on the testing samples (blue dot). The red line is the ground truth

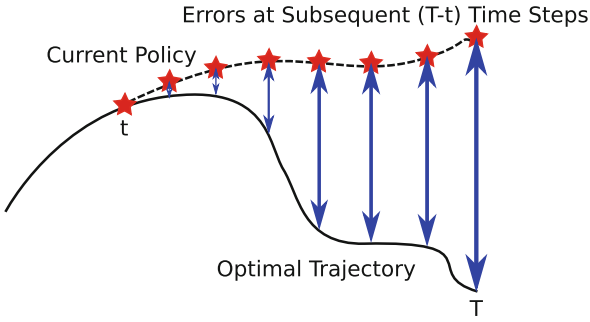

Fig. 8.3 The compounding errors increase along the trajectory chosen by current policy in a task with sequential decisions

which makes DAgger more effective and efficient for online imitation learning in sequential prediction problem like reinforcement learning. The demonstration dataset $\mathcal{D}$ is continuously aggregated with new dataset $\mathcal{D}_i$ for time step i containing expert actions and corresponding states visited by current policy during the whole imitation learning process. So, DAgger also has a drawback that it needs to iteratively interact with the expert, which is usually demanding in real-world applications. The algorithm of DAgger is shown in Algorithm 1, where π^* is the expert policy and β_i is the parameter for soft-updating the policy at iteration i.

Algorithm 1 DAgger

1: Initialize $\mathcal{D} \leftarrow \emptyset$.
2: Initialize the policy $\hat{\pi}_1$ to any policy in policy set Π.
3: **for** i $= 1, 2, \ldots, N$ **do**
4: $\quad \pi_i \leftarrow \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
5: $\quad$ Sample several T-step trajectories using π_i.
6: $\quad$ Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by the expert.
7: $\quad$ Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$. Train current policy $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
8: **end for**
9: Return policy $\hat{\pi}_{N+1}$.

8.2.3 Variational Dropout

A method for alleviating the generalization problem in imitation learning is pre-training with **variational dropout** (Blau et al. 2018), instead of fully cloning the behavior of expert demonstrations in BC methods. The weights pre-trained with the demonstration dataset are parameterized as Gaussian distributions with Gaussian dropout of a certain variance threshold value for initializing reinforcement learning policies. Variational dropout approach for imitation learning (Molchanov et al. 2017) can be taken as a more advanced method for generalization than noise injection in the weights of pre-trained neural networks, it reduces the sensitivity

of choosing the magnitude of noise, which is a useful technique when applying imitation learning for initializing reinforcement learning.

8.2.4 Other Methods in BC

Some other concepts are involved in behavioral cloning as well. For example, some methods provide ways to generalize demonstrations to more general scenarios in a task using framework like **dynamic movement primitives (DMPs)** (Pastor et al. 2009), which apply a set of differential equations to represent any recorded movement. The differential equations in DMP usually contain adjustable weights, as well as non-linear functions to allow the generation of arbitrarily complex movements. Therefore DMP is more of an analytical-form solution compared with the "black-box" deep learning methods in behavioral cloning. Moreover, there exists a method in one-shot imitation learning (Duan et al. 2017) using **soft attention** on demonstrations to generalize model to unseen scenarios in training data. It is a meta-learning scheme to map one demonstration of one task to an effective policy for a variety of tasks. There are some other methods, which will not be discussed here.

8.3 Inverse Reinforcement Learning Approach

Another major category of imitation learning approaches is composed of techniques based on **inverse reinforcement learning (IRL)** (Ng et al. 2000; Russell 1998). The IRL problem is defined to be the problem of extracting a reward function given observed, optimal behavior, represented as expert policy π_E. Instead of directly learning a mapping from states to actions using the demonstration data, IRL-based methods iteratively alternate between using the demonstration to infer a hidden reward/cost function and using reinforcement learning with the inferred reward function to learn an imitating policy. IRL chooses the reward function R to make the policy optimal and moreover to favor solutions that make any single-step deviation from π_E as costly as possible. For all reward functions R satisfying $|R(s)| \leq R_{\max}, \forall s$, IRL method chooses the R^* following:

$$R^* = \arg\max_R \sum_{s \in \mathcal{S}} \left(Q^\pi(s, a_E) - \max_{a \in A \setminus a_E} Q^\pi(s, a) \right) \tag{8.4}$$

where $a_E = \pi_E(s)$ or $a_E \sim \pi(\cdot|s)$ is the expert (optimal) action. IRL-based techniques have been used for a variety of tasks such as maneuvering a helicopter (Abbeel and Ng 2004) and object manipulation (Finn et al. 2016b). IRL (Ng et al. 2000; Russell 1998) tries to extract a reward function from observed optimal behavior, like the expert demonstrations, but the reward function may not be unique (discussed later). A typical method in IRL is to use maximum causal entropy

regularization, which is maximum entropy (MaxEnt) IRL (Ziebart et al. 2010). The MaxEnt IRL can be represented as the following two stages:

$$\mathrm{IRL}(\pi_E) = \arg\max_R \mathbb{E}_{\pi_E}[R(s,a)] - \mathrm{RL}(R) \tag{8.5}$$

$$\mathrm{RL}(R) = \max_\pi H(\pi(\cdot|s)) + \mathbb{E}_\pi[R(s,a)] \tag{8.6}$$

which forms the RL $\circ$ IRL(π_E) policy learning framework. The first formula IRL(π_E) learns a reward function to maximize the difference of reward values between the expert policy and the reinforcement learning policy, and it can be replaced by Eq. (8.4) as the Q value is an estimation of rewards. The second formula RL(R) is the entropy-regularized (forward) reinforcement learning with the learned reward function R from the first formula. The entropy $H(\pi(\cdot|s))$ here is the entropy of the policy distribution given a specific state.

Shannon's information entropy of distribution P over random variable X measures the uncertainty of that probability distribution.

Definition 8.1 The **entropy** of a discrete random variable, X, distributed according to p is

$$H_p(X) = \mathbb{E}_{p(X)}[-\log p(X)] = -\sum_{X\in\mathcal{X}} p(X)\log p(X) \tag{8.7}$$

For the case of stochastic policies in reinforcement learning, the random variables are usually aligned in a vector with the same dimension as the action space. The commonly used distributions are diagonal Gaussian distributions and categorical distributions, the derivation of their entropy is trivial (referred to the chapters for algorithms implementation).

It is also common to see the cost function $c(s,a) = -R(s,a)$, which is minimized in the reinforcement learning process as follows:

$$\mathrm{RL}(c) = \arg\min_\pi -H(\pi) + \mathbb{E}_\pi[c(s,a)] \tag{8.8}$$

where $H(\pi) = \mathbb{E}_\pi[-\log\pi(a|s)]$ is called entropy of policy π. The cost function $c(s,a)$ is usually a measurement of the similarity between distributions from current policy π and the demonstrations dataset. The entropy term $H(\pi)$ can be regarded as a normalization term for the uniqueness of optimality.

By substituting the above formula into the IRL formula Eq. (8.5), we can represent the objective of IRL in a max–min form, which tries to learn a cost function $c(s,a)$ of state s and action a with the objective of maximizing the entropy-regularized reward, as well as learning the policy π.

$$\max_c \left(\min_\pi -\mathbb{E}_\pi[-\log\pi(a|s)] + \mathbb{E}_\pi[c(s,a)]\right) - \mathbb{E}_{\pi_E}[c(s,a)] \tag{8.9}$$

where the π_E denotes the expert policy for generating expert demonstrations and π is the policy trained in reinforcement learning process. The learned cost function will assign high entropy to expert policy and low entropy to other policies.

8.3.1 Challenges of IRL

- **Non-uniqueness of Reward** (or **Reward Ambiguity**): The function search in IRL is ill-posed as the demonstrated behavior could be induced by multiple reward/cost functions. It originates from the concept of reward shaping (Ng et al. 1999), which describes a class of reward transformations that preserve the optimal policy. The main result is that under the following reward transformation:

$$\hat{r}(s, a, s') = r(s, a, s') + \gamma\phi(s') - \phi(s) \tag{8.10}$$

  the optimal policy remains unchanged for any function $\phi : \mathcal{S} \to \mathbb{R}$. The reward function learned with IRL methods from demonstration data only cannot disambiguate between reward functions within the class of above transformations.

  Constraints are thereby imposed on the rewards or the policy to ensure the optimality uniqueness of the demonstrated behavior. For example, the reward function is usually defined to be a linear (Ng et al. 2000; Abbeel and Ng 2004) or convex (Syed et al. 2008) combination of the state features. The learned policy is also assumed to have the maximum entropy (Ziebart et al. 2008) or the maximum causal entropy (Ziebart et al. 2010). These explicit constraints potentially limit the generability of the proposed methods (Ho and Ermon 2016).
- **Intensive Computational Cost**: The IRL could learn a better policy from demonstrations and interactions in the general reinforcement learning process. However, using reinforcement learning to optimize the policy given the inferred reward function requires the agent to interact with its environment, which can be costly from the perspectives of time and safety. Moreover, the IRL step typically requires the agent to solve an MDP in the inner loop of iterative reward optimization (Abbeel and Ng 2004; Ziebart et al. 2008), which can be extremely costly from a computational perspective. However, recently, a number of methods have been developed, which relieve this requirement (Finn et al. 2016b; Ho and Ermon 2016). One of these approaches is called generative adversarial imitation from observation (GAIL) (Ho and Ermon 2016), which will be described in Sect. 8.3.2.

8.3.2 Generative Adversarial Approach

The generative adversarial imitation learning (GAIL) (Ho and Ermon 2016) borrows the idea of generative adversarial training in generative adversarial networks

(GANs) (Goodfellow et al. 2014). The associated algorithm can be thought of as trying to induce an imitator state-action occupancy measure that is similar to that of the demonstrator. It applies a discriminator in GAN for providing the estimation of an action-value function based on demonstrations. In a general process of action-value based reinforcement learning with algorithms like TRPO, PPO, etc., the action value is provided from the demonstrations with a generative approach as:

$$Q(s,a) = \mathbb{E}_{\mathcal{T}_i}[\log(D_{\omega_{i+1}}(s,a))] \tag{8.11}$$

where $\mathcal{T}_i$ are samples from explorations for iteration i and $D_{\omega_{i+1}}(s,a)$ is the output value from the discriminator with parameters ω_{i+1}. The ω_{i+1} indicates the Q-value is estimated with one-step updated discriminator weights, therefore with iteration $i+1$. The loss function of the discriminator is defined in a general way:

$$\text{Loss} = \mathbb{E}_{\mathcal{T}_i}[\nabla_\omega \log(D_\omega(s,a))] + \mathbb{E}_{\mathcal{T}_E}[\nabla_\omega \log(1 - D_\omega(s,a))] \tag{8.12}$$

where the $\mathcal{T}_i, \mathcal{T}_E$ are samples from explorations and expert demonstrations, respectively. ω are parameters of the discriminator. Figure 8.4 shows the architecture of GAIL.

With the method of GAIL, the policy can be learned with samples generalized from demonstrations with lower computational cost, compared with methods via IRL. It does not need to interact with the expert during training, which happens in method like DAgger and is sometimes not accessible in practice.

This approach can be further generated to multimodal policy for learning across tasks. Multimodal imitation learning with GAN (Hausman et al. 2017) applies a more advanced objective function (additional latent indices for different tasks)

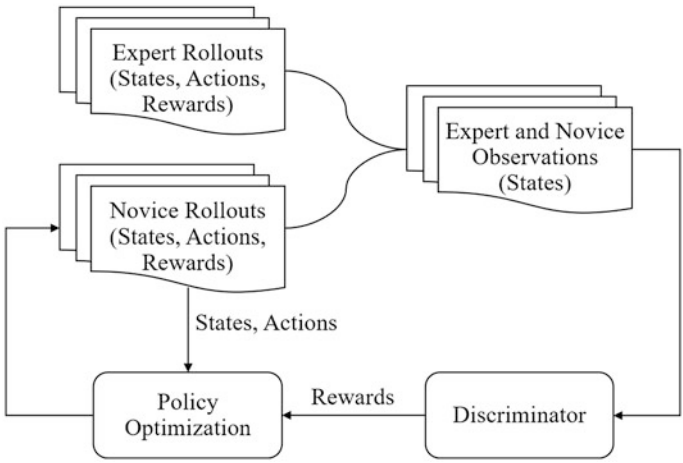

Fig. 8.4 The architecture of GAIL, adapted from Ho and Ermon (2016)

in a generative adversarial form, to automatically segment the demonstrations for different tasks and learn a multimodal policy in an imitation manner.

According to Goodfellow et al. (2014) (details of GANs are introduced in Chap. 1), with infinite data and infinite computation, at optimality, the distribution of generated state-action pairs should exactly match the distribution of demonstrated state-action pairs under the GAIL objective. The downside to this approach, however, is that we bypass the intermediate step of recovering rewards. Specifically, note that we cannot extract reward functions from the discriminator, as $D_\omega(s, a)$ will converge to 0.5 for all (s, a) pairs.

8.3.3 Generative Adversarial Network Guided Cost Learning (GAN-GCL)

As mentioned above, the GAIL method cannot recover the reward function from the demonstration data. A similar work named generative adversarial network guided cost learning (GAN-GCL) optimizes the guided cost learning (GCL) method based on GAN's structure, to extract an optimal reward function from the optimal discriminator trained with the demonstration data. We will describe this method in details.

The GAN-GCL method (specifically the GCL) is based on the maximum causal entropy IRL described above, which considers an entropy-regularized Markov decision process (MDP). The goal in entropy-regularized MDP for reinforcement learning is to maximize the expected entropy-regularized discounted reward:

$$\pi^* = \arg\max_\pi \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T}\gamma^t(r(S_t, A_t) + H(\pi(\cdot|S_t)))\right] \tag{8.13}$$

which is a more specific term used for learning the policy in practice originated from Eq. (8.5). It can be shown that the optimal policy $\pi^*(a|s)$ gives trajectory distribution satisfying $\pi^*(a|s) \propto \exp(Q^*_{soft}(s, a))$ (Ziebart et al. 2010), where $Q^*_{soft}(S_t, A_t) = r(S_t, A_t) + \mathbb{E}_{\tau\sim\pi}[\sum_{t'=t}^{T}\gamma^{t'-t}(r(s_{t'}, a_{t'}) + H(\pi(\cdot|s_{t'})))]$ denotes the soft Q-function (also used in soft actor-critic algorithm).

The IRL problem can be interpreted as solving the maximum likelihood problem:

$$\max_\theta \mathbb{E}_{\tau\sim\pi_E}[\log p_\theta(\tau)] \tag{8.14}$$

where π_E is the expert policy for providing demonstrations, and $p_\theta(\tau) \propto p(S_0)\prod_{t=0}^{T} p(S_{t+1}|S_t, A_t)e^{\gamma^t r_\theta(S_t, A_t)}$ parameterizes the reward function $r_\theta(s, a)$ but with the dynamics (or transition) and initial state distribution of the MDP. $p_\theta(\tau)$ is the trajectory-centric distribution of the demonstration data derived from state-centric π_E, $p_\theta(\tau) \sim \pi_E$. With deterministic transition $p(S_{t+1}|S_t, A_t) = 1$,

this simplifies to an energy-based model $p_\theta(\tau) \propto e^{\sum_{t=0}^{T} \gamma^t r_\theta(S_t, A_t)}$ (Ziebart et al. 2008). The parameterized reward function can be learned through optimizing parameters θ w.r.t the above objective. Similar to processes before, we can introduce the cost function here as the negative discounted cumulative rewards $c_\theta = -\sum_{t=0}^{T} \gamma^t r_\theta(S_t, A_t)$, parameterized by θ. Then the MaxEnt IRL can be viewed as modeling the demonstrations using a Boltzmann distribution in a trajectory-centric formulation, where the energy is given by the cost function c_θ:

$$p_\theta(\tau) = \frac{1}{Z} \exp(-c_\theta(\tau)) \tag{8.15}$$

where τ is the state-action trajectory and $c_\theta(\tau) = \sum_t c_\theta(S_t, A_t)$, and the partition function Z is the integral of $\exp(-c_\theta(\tau))$ over all trajectories consistent with the environment dynamics, for normalizing the probability. Estimating the partition function Z is difficult for large-scale or continuous domains, as precise estimation with dynamic programming for Z can only work in small and discrete domains. Otherwise we need to use approximated estimation methods, like the sampling-based GCL method.

GCL uses importance sampling for estimating Z with a new distribution $q(\tau)$ (the original demonstration distribution is $p(\tau)$) in MaxEnt IRL formulation:

$$\theta^* = \arg\min_\theta \mathbb{E}_{\tau \sim p}[-\log p_\theta(\tau)] \tag{8.16}$$

$$= \arg\min_\theta \mathbb{E}_{\tau \sim p}[c_\theta(\tau)] + \log Z \tag{8.17}$$

$$= \arg\min_\theta \mathbb{E}_{\tau \sim p}[c_\theta(\tau)] + \log \left(\mathbb{E}_{\tau' \sim q} \left[\frac{\exp(-c_\theta(\tau'))}{q(\tau')} \right] \right) \tag{8.18}$$

where the τ' is sampled from distribution q and $q(\tau')$ gives its probability. Therefore q can be optimized through minimizing the KL-divergence between $q(\tau')$ and $\frac{1}{Z} \exp(-c_\theta(\tau'))$ for updating the $q(\tau')$ during learning θ or equivalently as following:

$$q^* = \min \mathbb{E}_{\tau \sim q}[c_\theta(\tau)] + \mathbb{E}_{\tau \sim q}[\log q(\tau)] \tag{8.19}$$

Finn et al. (2016a) proposed to use GAN's manner for the above optimization problem, which optimizes the GCL with GAN structure and is similar to the GAIL method but with different specific formulations.

Note that in GAN the discriminator also tries to approximate one distribution with the other as:

$$D^*(\tau) = \frac{p(\tau)}{p(\tau) + q(\tau)} \tag{8.20}$$

We can apply it here in the GCL of MaxEnt IRL formulation,

$$D_\theta(\tau) = \frac{\frac{1}{Z}\exp(-c_\theta(\tau))}{\frac{1}{Z}\exp(-c_\theta(\tau)) + q(\tau)} \tag{8.21}$$

which leads to the method GAN-GCL. The policy π is trained to maximize $R_\theta(\tau) = \log(1 - D_\theta(\tau)) - \log D_\theta(\tau)$, and the reward function is therefore learned through optimizing the discriminator. The policy is learned through updating the sampling distribution $q(\tau)$ used to estimate the partition function. If the optimality is reached, the optimal cost function $c^*_\theta = -R^*_\theta(\tau) = -\sum_{t=0}^{T} \gamma^t r^*_\theta(S_t, A_t)$ can be learned for evaluating the optimal reward function, and the optimal policy can be derived with $\pi^* = q^*$. GAN-GCL provides an alternative approach for optimizing the MaxEnt IRL problem instead of directly maximizing the likelihood.

8.3.4 Adversarial Inverse Reinforcement Learning (AIRL)

As the above GAN-GCL is trajectory-centric, which means the full trajectories are estimated, it has high variance in estimation compared with estimating the single state-action pair. The adversarial inverse reinforcement learning (AIRL) (Fu et al. 2017) proposes to directly estimate the single state and action:

$$D_\theta(s, a) = \frac{\exp(f_\theta(s, a))}{\exp(f_\theta(s, a)) + \pi(a|s)} \tag{8.22}$$

where the $\pi(a|s)$ is the sampling distribution to be updated and the $f_\theta(s, a)$ is the learned function. The partition function is ignored in the above formula and the normalization of probability values can be guaranteed with *SoftMax* operator or sigmoid output activation in practice. It is proven that at optimality, $f^*(s, a) = \log \pi^*(a|s) = A^*(s, a)$, which gives the advantage function of optimal policy. However, the advantage function is a heavily entangled reward function with a baseline value subtracted. Fu et al. (2017) argue that the reward function cannot be robustly recovered for the changes in environment dynamics. Therefore, they also propose to learn the disentangled reward with AIRL through decoupling the reward function from the advantage function:

$$D_{\theta,\phi}(s, a, s') = \frac{\exp(f_{\theta,\phi}(s, a, s'))}{\exp(f_{\theta,\phi}(s, a, s')) + \pi(a|s)} \tag{8.23}$$

where $f_{\theta,\phi}$ is restricted to a reward approximator g_θ and a shaping term h_ϕ as:

$$f_{\theta,\phi}(s, a, s') = g_\theta(s, a) + \gamma h_\phi(s') - h_\phi(s) \tag{8.24}$$

where the extra approximation of h_ϕ is required.

8.4 Imitation Learning from Observation (IfO)

First, imitation learning from observation (IfO) is imitation learning without fully observable actions. One typical example of IfO is learning from the videos, in which the ground-truth actions of objects are not available from the frames only, but humans can still learn from videos like mimicking the actions. Therefore the examples of learning from videos are common to see in the literature of IfO. IfO regards imitation learning from a different perspective, compared with other methods introduced above. Therefore, there are inevitable overlappings of some specific methods introduced in this section with some methods introduced above, but in the IfO category. When you read this section, you should keep in mind that the IfO methods are in most cases orthogonal to other categories of methods as it looks at the imitation learning in a different perspective and focuses on the problem of unobservable actions.

The aforementioned algorithms, however, can hardly handle the demonstrations with partial or unobservable actions. One idea to learning from these demonstrations is to first recover actions from states and then adopt standard imitation learning algorithms to learn a policy from the recovered state-action pairs. For example, Torabi et al. recover actions from states by learning a dynamic model of state transitions, and then use a BC algorithm to find the optimal policy (Torabi et al. 2018a). However, the performance of this method is highly dependent on the learned dynamic model and may fail when the states transit with noise. Instead, Merel et al. proposed to learn from only state (or state feature) trajectories. They extended the GAIL framework to learn a control policy from only states of motion capture demonstrations (Merel et al. 2017) and showed that partial state features without demonstrator actions suffice for adversarial imitation. Similarly, Eysenbach et al. pointed out that the policy should control which states the agent visits, and thus use the states to train a policy by maximizing mutual information between the policy and the state trajectories (Eysenbach et al. 2018). Other studies have also tried to learn from raw observations instead of states. For instance, Stadie et al. extracted features from observations by the domain adaptation method to ensure that experts and novices are in the same feature space (Stadie et al. 2017). However, only using demonstrated states or state features may require a huge number of environmental interactions during the training since any possible information from actions is ignored.

In order to provide more clear structure about advanced methods in IfO, we organize the methods of IfO in the literature into two general groups: (1) model-based algorithms, and (2) model-free algorithms, which also follow one of the main taxonomies in reinforcement learning (as shown in Chap. 3). Next, we discuss the features of each group and present relevant algorithms from the literature as examples.

8.4.1 Model-Based

Similar to model-based reinforcement learning (as in Chap. 9), if the model of the environment can be learned precisely with low consumption, it can benefit the learning process through efficient planning. As imitation learning is about mimicking a sequential of actions instead of a single one in the interactive process with the environment, it inevitably involves the dynamics of the environment, which can be learned with model-based approaches. According to different types of dynamics models, the model-based IfO methods can be categorized as: (1) inverse dynamics models or (2) forward dynamics models.

Inverse Dynamics Models An inverse dynamics model is a mapping from state transitions $\{(S_t, S_{t+1})\}$ to actions $\{A_t\}$ (Hanna and Stone 2017). One work by Nair et al. (2017) in this category learns to predict a sequence of actions for rope manipulation with the sequences of images of a human manipulating a rope from initial condition to a goal condition, which requires to learn a pixel-level inverse dynamics model as follows:

$$A_t = M_\theta(I_t, I_{t+1}) \tag{8.25}$$

where the A_t is the predicted action by the inverse dynamics model M with the input pair of images I_t, I_{t+1}, and the model is parameterized by θ. A convolutional neural network is used for learning the inverse dynamics model. The robot collects rope manipulation samples with an exploration policy automatically. The collected samples are used for learning the inverse dynamics model, after which the robot conducts planning with the learned model and desired states from the human demonstration. The learned inverse dynamics model M_θ^* can actually serve as the policy for choosing actions similar to the demonstration with respect to the desired frame I^e:

$$A_t = M_\theta^*\left(I_t, I_{t+1}^e\right) \tag{8.26}$$

Another work called reinforced inverse dynamics modeling (RIDM) (Pavse et al. 2019) applies a reinforced post-training for fine-tuning the learned inverse dynamics model after training on samples collected with a pre-defined exploration policy. The pre-trained inverse dynamics model, as said above, is regarded as the policy for the agent in a reinforcement learning setting and a sparse reward function R can be applied for reinforcement learning fine-tuning process:

$$\theta^* = \arg\max_\theta \sum_t R\left(S_t, M_\theta^{pre}\left(S_t, S_{t+1}^e\right)\right) \tag{8.27}$$

where M_θ^{pre} is the pre-trained model and fine-tuned here in a reinforcement learning manner.

The covariance matrix adaptation evolution strategy (CMA-ES) or Bayesian optimization (BO) methods can be applied for optimizing the model for a low-dimensional cases. However, the author assumes that each observation transition is reachable through the application of a single action. Targeting at removing this unnecessary assumption, Pathak et al. (2018) allow the agent to execute multiple actions until it gets close enough to the next demonstrated frame.

The algorithms introduced above try to recover the policy with the inverse dynamics model for each single demonstration state. The behavioral cloning from observation (BCO) algorithm proposed by Torabi et al. (2018a), on the other hand, tries to recover the demonstration dataset with full observation-action pairs using the learned inverse dynamics model, and then learn the policy with the augmented demonstration dataset in a regular imitation learning manner, as shown in Fig. 8.5.

Guo et al. (2019) propose to apply a tensor-based model to infer the unobserved actions of the expert state sequences (the IfO problem), which is shown in Fig. 8.6. The policy of the agent is then optimized via a hybrid objective combining reinforcement learning and imitation learning as:

$$\theta^* = \arg\min_{\theta} \mathrm{L}_{RL}(\pi(a|s;\theta)) - \mathbb{E}_{\left(S_t^e, S_{t+1}^e\right)\sim\mathcal{D}}\left[\log \pi_\theta\left(M\left(S_t^e, S_{t+1}^e\right)|S_t^e\right)\right] \tag{8.28}$$

where the L_{RL} is a regular reinforcement learning loss term with policy π parameterized by θ. $\mathcal{D}$ is the demonstration dataset, and the second term is the behavioral cloning loss for maximizing the likelihood of predicting "expert" actions given the expert states s^e and inverse dynamics model M. Guo's method is, in a way, a combination of RIDM and BCO methods. Instead of using a parameterized inverse dynamics model like in other methods above, the inverse dynamics model M here is a low-rank tensor model with advantages over deep neural networks. The reward signals are required for providing the reinforcement learning loss, which is similar to RIDM.

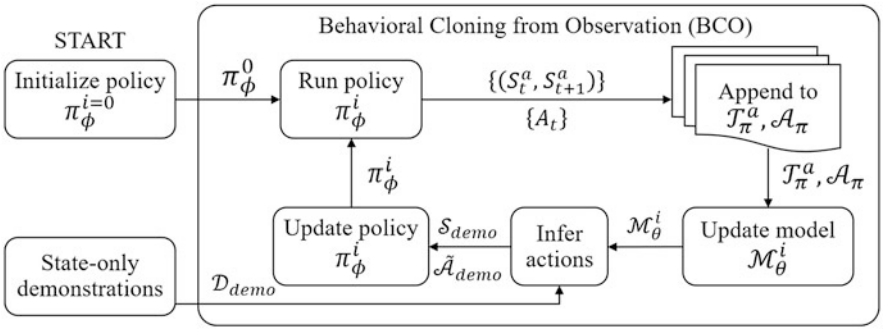

Fig. 8.5 The learning framework of Behavioral Cloning from Observation (BCO), adapted from Torabi et al. (2018a)

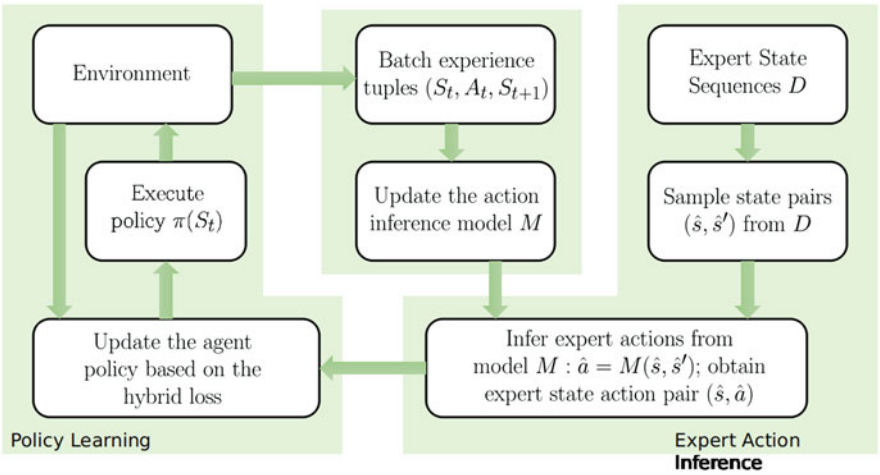

Fig. 8.6 The learning framework of hybrid reinforcement learning with expert state sequences framework, adapted from Guo et al. (2019)

Forward Dynamics Models A forward dynamics model is a mapping from state-action pairs, $\{(S_t, A_t)\}$, to the next states, $\{S_{t+1}\}$. One typical method leveraging the forward dynamics model in IfO is called imitating latent policies from observation (ILPO) (Edwards et al. 2018). ILPO applies two networks in its learning process: the latent policy network and the action remapping network. The latent policy network includes an action inference module which maps the state S_t to a latent action z, and a forward dynamics module which predicts the next state S_{t+1} given current state S_t and the latent action z. The update rules of these two modules are as follows:

$$\omega^* = \arg\min \mathbb{E}_{\left(S_t^e, S_{t+1}^e\right)\sim\mathcal{D}}\left[||G_\omega(S_t^e, z) - S_{t+1}^e||_2^2\right] \tag{8.29}$$

for the latent dynamics model G_ω and

$$\theta^* = \arg\max \mathbb{E}_{\left(S_t^e, S_{t+1}^e\right)\sim\mathcal{D}}\left[|| \sum_z \pi_\theta\left(z|S_t^e\right) G_\omega\left(S_t^e, z\right) - S_{t+1}^e||_2^2\right] \tag{8.30}$$

for the latent policy $\pi_\theta(\cdot|z)$, where $\mathcal{D}$ is the expert demonstration dataset.

However, since the latent action produced in the latent policy network may not necessarily be the true action in real dynamics of the environment, the action remapping network is applied for associating the latent actions to the true actions. The usage of latent actions requires no interactions with the environment during learning the latent model and latent policy, while the remapping action network only needs limited number of interactions with the environment, which makes the algorithm efficient in the learning process.

8.4.2 Model-Free

Apart from model-based IfO methods with the learned dynamics models, there are also model-free IfO methods, which is another main category for learning without the models. The models can be hard to learn well for highly complicated dynamics, as in regular reinforcement learning settings. There are two main approaches for model-free IfO: (1) generative adversarial methods and (2) reward-engineering methods. The generative approach is similar as in regular IL, but with the states as demonstrations only.

Generative Adversarial Methods A general framework in the generative adversarial IfO is modified from previously introduced GAIL method in IRL for regular IL. Instead of feeding the state-action pairs into the discriminator, only the states are compared with the discriminator from either explored samples of current policy or the expert demonstration, which gives the loss:

$$\text{Loss} = \mathbb{E}_{\mathcal{D}}[\nabla_\omega \log(D_\omega(s))] + \mathbb{E}_{\mathcal{D}^e}[\nabla_\omega \log(1 - D_\omega(s))] \tag{8.31}$$

where $\mathcal{D}$ is the explored sample set with current policy and $\mathcal{D}^e$ is the demonstration dataset. Different specific algorithms will have different specific forms and modifications based on that.

For example, Merel et al. (2017) developed a variant of GAIL with only partially observed state features and without access to actions to provide human-like motions for humanoids via the GAN's structure. Similar as RIDM method and hybrid reinforcement learning method in model-based IfO, it also applies the reinforcement learning module together with an imitation learning module, but with a hierarchical structure. The reinforcement learning module is a high-level controller built on the low-level controller with the BC method for capturing the motion features of humanoid. Trajectories of states and actions are collected during the interaction process of a stochastic policy π and the environment, which corresponds to the generator in GAN framework. The state-action pairs are transformed into features, z, in which the actions may be excluded. The demonstration data are assumed to be in the same feature space according to the original paper. Either demonstration data or generated data are evaluated by the discriminator to yield a probability of the data being demonstration data. The output value of the discriminator is then used as a reward to update the imitation policy using reinforcement learning, similar to Eq. (8.12) in GAIL. An additional context variable is also applied for learning multi-behavior policies. The loss of the discriminator can be written as:

$$\text{Loss} = \mathbb{E}_{z\sim s, s\sim\mathcal{D}}[\nabla_\omega \log(D_\omega(z, c))] + \mathbb{E}_{z^e\sim s^e, s^e\sim\mathcal{D}^e}\left[\nabla_\omega \log(1 - D_\omega(z^e, c^e))\right] \tag{8.32}$$

where z, z^e are encoded features of s, s^e sampled from reinforcement learning explorations $\mathcal{D}$ and expert demonstrations $\mathcal{D}^e$, respectively, and c, c^e are the context variables indicating different behaviors.

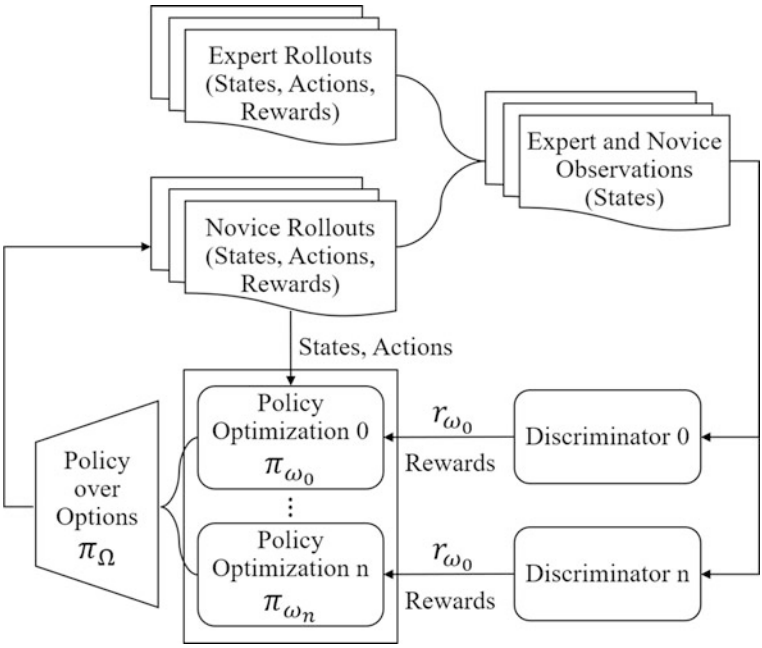

Fig. 8.7 The architecture of OptionGAN, adapted from Henderson et al. (2018)

The OptionGAN (Henderson et al. 2018) proposed by Henderson et. al. applies the option framework in hierarchical reinforcement learning (details in Chap. 10) to recover the joint reward-policy options with generative adversarial architecture using only observed states, as shown in Fig. 8.7. With the decomposition of policies, it is able to not only learn well on simple tasks, but also learn a general policy over options for complicated continuous control tasks.

One potential problem of IfO with above methods is that, even if the learned optimal policy is able to generate a state distribution that is very similar to the expert policy, it still does not mean that the actions are exactly the same for both imitation policy and the expert policy for all states. A simple example by Torabi et al. (2019d) would be, in a ring-like environment, two agents that move with the same speed but different directions (i.e., one clockwise and another one counter-clockwise) would result in each exhibiting the same state distribution even though their behaviors are opposite to one another (i.e., different action distributions given the states).

One way of solving above mismatch problem in action distributions is to feed a sequence of states instead of a single one to the discriminator, like proposed by Torabi et al. (2019b) and Torabi et al. (2018b), a similar algorithm but only with the difference that the discriminator considers state transitions, $\{(S_t, S_{t+1})\}$, as the input instead of single states. This changes the loss function of the discriminator to be

$$\mathbb{E}_{\mathcal{D}}[\nabla_\omega \log(D_\omega(S_t, S_{t+1}))] + \mathbb{E}_{\mathcal{D}^e}[\nabla_\omega \log(1 - D_\omega(S_t, S_{t+1}))] \tag{8.33}$$

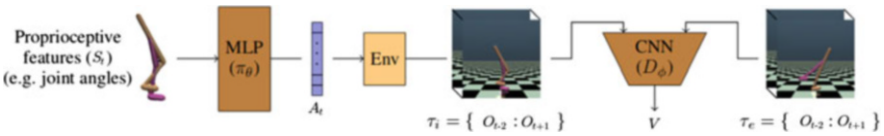

Fig. 8.8 Imitation learning from observations only, using the proprioceptive state. Figure is adapted from Torabi et al. (2019c)

where the state sequence can also be chosen to be longer than two in practice.

Another work by Torabi et al. (2019c) leverages the proprioceptive features as the state input for the policy instead of the observed images, to model the humans or animals proprioception-based control in reinforcement learning agents. Because of the low dimensions of the proprioceptive features, the policy can be represented by a simple multi-layer perceptron (MLP) instead of a convolutional neural network (CNN), while the discriminator still takes a sequence of observed images as inputs from both the explored samples and the expert demonstration, as shown in Fig. 8.8. The low-dimensional proprioceptive features make the learning process more efficient as well.

As mentioned in Chap. 7, the low sample efficiency is one of the key problems in present reinforcement learning algorithms, which also holds for the imitation learning and IfO areas. As the generative adversarial approaches are within the IRL domain, those methods introduced above can suffer from intensive computational cost as mentioned in Sect. 8.3. These adversarial imitation algorithms often require large numbers of demonstration examples and learning iterations to learn a policy imitating a demonstrator's behavior successfully. To further improve the sample efficiency of above methods, Torabi et al. (2019a) proposed to apply the linear quadratic regulators (LQR) (Tassa et al. 2012) as a trajectory-centric reinforcement learning method during the policy learning process, which has potential to make it possible to apply the algorithm for real-robot imitation learning.

The above works are mostly based on the basic assumptions that the demonstration data space and the imitators' learning space are consistent. However, when there is a mismatch between the two spaces, for example, the changes of viewpoint by placing the camera in different positions in the three-dimensional space for providing observation, the general imitation learning methods will have a degradation in performance. The difference of the spaces of demonstration and the imitator can be either in action space or the state space. For the difference in action spaces, Żołna et al. (2018) proposed to use pairs of states with random time gaps instead of consecutive states as the input of the discriminator, which can be regarded as dataset augmentation with noise for more robust and general performances. In their own experiments, it is indeed shown to improve the performances of imitator's policy with different action spaces from the demonstration. While for the difference of the state space, like the viewpoint changing mentioned above, Stadie et al. (2017) have proposed to apply a classifier to distinguish among samples of different viewpoints, with the output values of some initial layers in the discriminator as

inputs. The proposed method leverages the idea of domain confusion to learn the domain agnostic features, where the domain indicates different viewpoints in this case. The confusion is maximized in the first layers of the discriminator (as a feature extractor) but minimized for the classifier, which also leverages the adversarial training framework. After training, the learned features of the extractor (first several layers of the discriminator) are invariant to the viewpoint.

There are also some other works in this field. Sun et al. (2019) proposed the first provably efficient algorithm in IfO, called Forward Adversarial Imitation Learning (FAIL), which can learn a near-optimal policy with the number of samples in a polynomial relationship with all relevant parameters but independent of the number of unique observations. The minimax game in FAIL learns a policy that matches the state distribution of the next state given the policies of the previous time steps. Recently, a method called Action-Guided Adversarial Imitation Learning (AGAIL) proposed by Sun and Ma (2019) tries to leverage the states and incomplete actions in demonstrations, which is a combination of IfO and traditional IL. The discriminator is used for discriminating single states, similar to the approach of Merel et al. (2017) described before. Additionally, a guided Q-network is employed to learn the true posterior of $p(a^e|a \sim \pi(s^e))$ in a supervised learning manner, where (s^e, a^e) denote samples of expert demonstration.

Reward-Engineering Methods The generative adversarial approach naturally provides the reward signals, from which the imitation policy can learn in a reinforcement learning manner. Apart from the generative adversarial approach, there are also methods with reward engineering for solving model-free IfO. Actually, the method RIDM in model-based IfO is a method with reward engineering mentioned in the previous section. The reward engineering indicates the need of manually designed reward functions for learning an imitation policy in a reinforcement learning manner with expert demonstrations. Reward engineering transfers the supervised learning approach of imitation learning into a reinforcement learning problem through formalizing the reward function for the reinforcement learning agent. What needs to be noticed is that the manually designed reward function does not have to be the true reward function leading to the expert policy, but more of an estimation from the demonstration dataset or prior knowledge about the tasks. For example, Kimura et al. (2018) proposed to use the Euclidean distance of the predicted next state by the predictor and the true next state by the demonstrator as the reward function. Then the reward function is used for learning an imitation policy in general reinforcement learning settings.

Another reward-engineering approach is called time-contrastive networks (TCNs) proposed by Sermanet et al. (2018) (Fig. 8.9). To handle the multi-viewpoint problem as mentioned before, which is important for learning from human behaviors, the TCN method learns a viewpoint invariant representation capturing the relationships among objects with the TCN network using several (two in the original paper) synchronous camera views from different perspectives. The adversarial training is therefore applied in the embedded representation space instead of the original state space in other IfO methods. The representation is

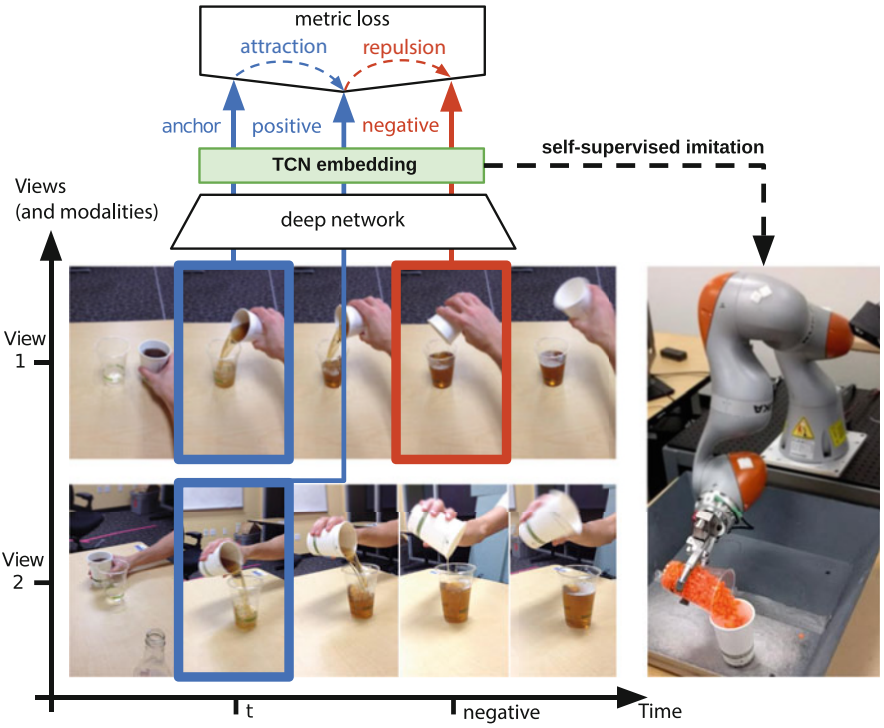

Fig. 8.9 The learning framework of time-contrastive network (TCN) with a triplet loss for observation embedding in self-supervised imitation learning from observations only. Figure is from Sermanet et al. (2018)

trained with a triplet loss with the TCN embedding network. The triplet loss is set to disperse the temporal neighbors of consecutive frames in the video demonstration with similar visual features but different actual dynamic states and also to attract those simultaneous frames from different viewpoints but with the same dynamic state in the embedding space. Therefore, the imitation policy can learn with unlabeled video of human demonstrations in a self-supervised learning manner. Similar as in Kimura's work, the reward function is defined to be the Euclidean distance between the state of demonstration and the state of the agent at each time step, but in the embedding space instead of the state space. The TCN is designed to work for single frame state embedding. Dwibedi et al. (2018) extended the work of TCN to multiple frames embedding for better representing the patterns in trajectory. Aytar et al. (2018) also took a similar approach, learning an embedding function for the YouTube video frames based on the demonstration, to solve the hard-exploration tasks like Montezuma's Revenge and Pitfall mentioned in the exploration challenge of Chap. 7. It can handle the small variance in domain like video artifacts and color changes. The measurement of closeness between the imitator's embedded states and some demonstrator's embedded states is also used as the reward function.

As mentioned in the previous section of the adversarial generative approach, a classifier can be employed to distinguish among observations from different viewpoints. A classifier can also be used to predict the order of frames in the demonstration as proposed by Goo and Niekum (2019), via a shuffle-and-learn (Misra et al. 2016) training manner. A reward function can be defined with respect to the learned classifier for training the imitator policy. Also in previous sections of the adversarial generative approach, the mismatch between the state spaces, like being caused by the viewpoints, can be handled with an invariant feature representation. However, instead of using the output values of the discriminator with demonstration states and the imitator's state as inputs, it can also train an imitation policy with a reward function defined to be the Euclidean distance between the two kinds of states in the representation space, as proposed by Gupta et al. (2017) and Liu et al. (2018).

8.4.3 Challenges of IfO

With the above mentioned methods developed in IfO, the agent can learn the policy from the observed states only, but still suffers from several problems as mentioned in the survey by Torabi et al. (2019d), which are listed as the challenges below:

- **Embodiment Mismatch**: The embodiment mismatch generally describes the differences of appearances (for visual-based control), dynamics, and other features between the imitator's domain and the demonstrator's domain. A typical example would be letting a robotic arm mimic the motion of a human's arm. Due to the significant differences in the controlling dynamics and perspectives of looking at the agents, the imitation learning process can be potentially very hard to conduct. Even determining if the robot is in the same state as the human's arm can be difficult. One way to solve this is to learn the hidden correspondences or latent representations that are invariant for the differences of the two domains, and conducting the imitation learning based on the correspondences or in that learned representation space. One IfO method developed to address this problem learns a correspondence between the embodiments using autoencoders in a supervised fashion (Gupta et al. 2017). The autoencoder is trained in such a way that the encoded representations are invariant with respect to the embodiment features. Another method learns the correspondence in an unsupervised fashion with a small amount of human supervision (Sermanet et al. 2018).
- **Viewpoint Difference**: As mentioned in several methods described above, like the TCN and some methods in model-based IfO, the difference of viewpoints can degrade the performance of the imitation policy significantly, for visual-based control with demonstration data provided by images or videos from a camera. Generally an encoding model for representing the states in a viewpoint invariant space is required as in Sieb et al. (2019), or a classifier for predicting the specific viewpoint for one frame as in Stadie et al. (2017). Another IfO approach that attempts to address this issue learns a context translation model to translate an

observation by predicting it in the target context (Liu et al. 2018). The translation is learned using data that consists of images of the target context and the source context, and the task is to translate the frame from the source context to that of the target. It would require the similar samples of the target context to be collected as in the source context.

8.5 Probabilistic Methods

Apart from the parameterized methods with deep neural networks (DNN), a variety of probabilistic inference methods are applied for imitation learning as well, especially in the robot motion domains, which include Gaussian mixture regression (GMR) (Calinon 2016), dynamical movement primitives (DMPs) (Pastor et al. 2009), probabilistic movement primitives (ProMPs) (Paraschos et al. 2013), kernelized movement primitives (KMP) (Huang et al. 2019), Gaussian process regression (GPR) (Schneider and Ertel 2010), and GMR-based GP process (Jaquier et al. 2019). As this book aims at introducing deep reinforcement learning with parameterization methods using DNN, we will only briefly discuss about probabilistic methods because the combination of probabilistic methods with DRL is non-trivial, unlike other previous approaches introduced in this chapter.

However, even if it is hard to apply the probabilistic methods on DRL tasks like using the supervised imitation learning as an initialization for reinforcement learning introduced in next section, probabilistic methods are still attractive to be investigated for imitation learning with several advantageous properties. Unlike the deterministic prediction results given by DNN, the covariance matrices of the prediction distributions computed by GMR, ProMPs, and KMP encode the variability of the predicted trajectory. This can be useful when applying the learned model on predicting or decision-making tasks where the belief of the prediction is also important, like ensuring the safety during robotic manipulation or vehicle driving cases. Apart from that, probabilistic methods usually have analytic solutions with the support of probabilistic theory, which is different from the "black-box" optimization process of DNN-based methods. This also makes probabilistic methods able to be solved within a short time when the amount of data is small. Probabilistic methods like GMR-based GP process have the quick adaptability for the unseen input datapoints, which will be discussed in the following sections. For probabilistic methods in IL, the dataset is considered to be provided in a labeled format with pairs of input and output, which is usually the set of state-action pairs $\{(s_i, a_i)|i = 0, \ldots, N\}$ for common reinforcement learning cases or time-state pairs $\{(t, S_t)|t = 0, \ldots, N\}$ (Jaquier et al. 2019) for the time-aligned demonstrations.

GMR-based GP regression is a combination of Gaussian mixture regression and Gaussian process regression. GMR exploits the Gaussian conditioning theorem to estimate the distribution of output data given input data. A Gaussian Mixture Model (GMM) is used to fit on the joint distribution of input and output datapoints with an Expectation Maximization (EM) algorithm. The conditional means and covariances

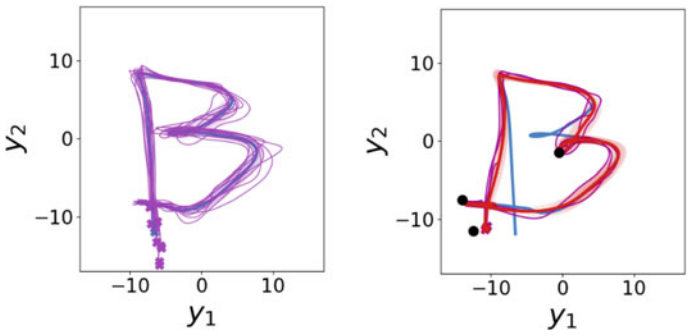

Fig. 8.10 GMR-based GP process for imitation learning. The left image shows the prior mean of the process and the sample trajectories in blue and purples lines, respectively. The right image shows the prior mean (same as the left one), sampled, and predicted trajectories in blue, pink, and red lines, respectively. The three black dots in the right image are observations. Figure is adapted from Jaquier et al. (2019)

given observed input can be solved in closed form, and the output can therefore be predicted via a linear combination of conditional expectations with the test input datapoints. GP aims at learning a deterministic input–output relationship, just like DNN's approach, based on a Gaussian prior over potential objective functions. The GMR-based GP is combined as a GP with its prior mean equal to the conditional mean of the GMR model, and with its kernel in the form of a sum of all separable kernels associated with the components of the corresponding GMM. This combination takes the advantage of the ability of GPs to encode various prior beliefs through the mean and kernel functions and allows the variability information retrieved by GMR to be encapsulated in the uncertainty estimated by the GP. When given new and unseen input observation points, the GMR-based GP method is able to quickly adapt to them and predict reasonable outputs as shown in Fig. 8.10. For a two-dimensional trajectories estimation process, the left image in Fig. 8.10 shows the given samples in purple lines, and the prior mean in blue line. The right image is the GMR-based GP process with 3 new observation points in black, and with the pink lines showing the sampled trajectories and red line as prediction. This method is testified to have a great performance on leveraging demonstrations but quickly adapting to new datapoints, which can be applied on manipulating the robot to avoid obstacles with demonstrations.

8.6 IL as Initialization for RL

The basic setup for applying imitation learning is to learn a policy without any reinforcement signals but only the demonstrations data, which means the learned policy through imitation learning is the final policy from the demonstrations. However, in practice, the policy learned from imitation learning is usually not general enough,

especially for unseen cases. Therefore, we can leverage the imitation learning in the reinforcement learning process, which improves the learning efficiency of reinforcement learning. For example, a pre-trained policy using demonstration data can be used to initialize the policy in reinforcement learning. More about these approaches will be discussed later. Therefore, we do not require the policy from imitation learning to be optimal, but good enough with a relatively simple imitation learning process, like applying a supervised learning approach. So, we only choose some of the simple and straightforward methods described below as an initialization method for subsequent reinforcement learning processes. Those fancier techniques in imitation learning will provide a better initialization policy with no doubts, but may have drawbacks as longer pre-training time and so on.

Generally, the policy learned from imitating the demonstrations in a supervised manner, including the **BC, DAgger, Variational Dropout, and so on**, can be regarded as a good initialization for reinforcement learning policy, using methods like **policy replacement or residual policy learning** described in the following sections. We will experimentally show the improvement in reinforcement learning with the initialization policy trained using above mentioned supervised learning methods in the following sections.

In addition to the policy replacement approach for initialization of reinforcement learning, residual policy learning (Silver et al. 2018; Johannink et al. 2019) is another approach to realize initialization. It is based on good but imperfect controllers for robot manipulation tasks, and to learn a residual policy on top of that initial controller. For robot manipulation in real world, the initial controller could be a pre-trained policy in simulation; and for robot manipulation in simulation, the initial controller could be from the pre-trained supervised learning with expert trajectories as in Sect. 8.3.2.

The action in residual policy learning follows the combinatorial policy, which is the sum of the initial policy π_{ini} and the residual policy π_{res}:

$$a = \pi_{ini}(s) + \pi_{res}(s) \tag{8.34}$$

In this way, the residual policy learning is able to preserve the initialized policy performance to the best advantage.

Example: DDPG with Residual Policy Learning

We apply the DDPG algorithm for leveraging the demonstrations with residual policy learning. According to the residual policy learning, the actor's policy in DDPG consists of two parts: one is the pre-trained initialization policy, which will be fixed after initialization, and another one is the residual policy to be trained during the learning process. The initialization policy is pre-trained with the demonstration samples generated from inverse kinematics, which is the same as the policy replacement method. The pre-trained initialization policy only works for the

actor part in DDPG. The process of applying residual policy learning in DDPG is as follows:

(1) Initialize all neural networks in DDPG with residual learning, including a general initialization of the critic, target critic, and an initialization with zero-valued final layers for the residual policy and the target residual policy, and an initialization with imitation learning for the policy and the corresponding target, totally six networks. Fix the initialized policy and its target, and start the training process.
(2) Let the agent interact with the environment, and the action value is the sum of the action values from the initialization policy and the residual policy: $a = a_{ini} + a_{res}$; store samples in the form of $(s, a_{res}, s', r, done)$.
(3) Draw samples $(s, a_{res}, s', r, \text{done})$ from the memory buffer, we have

$$Q_{target}(s, a_{res}) = r + \gamma Q^T\left(s, \pi_{res}^T(s)\right) \tag{8.35}$$

where Q^T, π_{res}^T denote the target critic and the target residual policy, respectively. The critic loss is $\text{MSE}(Q_{target}(s, a_{res}), Q(s, a_{res}))$. The objective for the actor is to maximize the action-value function of state s and action a_{res} as follows:

$$\max_{\theta} Q(s, a_{res}) = \max_{\theta} Q(s, \pi_{res}(s|\theta)) \tag{8.36}$$

which can be optimized via deterministic policy gradient.
(4) Repeat above steps (2) and (3) until the policy is converged or near optimal.

Compared with general DDPG algorithm, the difference of applying residual policy learning is just to learn action-value function and the policy with respect to the residual policy actions instead of the overall actions for the agent.

8.7 Other Approaches of Leveraging Demonstrations in RL

8.7.1 Feeding Demonstrations into Replay Buffer

Instead of pre-training a policy to initialize the reinforcement learning policy, deep Q-learning from demonstrations (DQfD) (Hester et al. 2018) leverages demonstrations through directly feeding those expert trajectories into memory buffer of off-policy reinforcement learning. It applies DQN for only discrete action space applications. DQfD uses a replay buffer initialized with all expert demonstrations and then keeps storing new samples in it. It applies the prioritized experience replay to sample the training batch from the replay buffer, and DQfD trains the policy using a combination of a supervised hinge loss for imitating the demonstrations and a general TD loss.

The approach of deep deterministic policy gradient from demonstrations (DDPGfD) (Večerík et al. 2017) is a method similar with the DQfD method as described above, but applies DDPG for continuous action space applications. DDPGfD leverages demonstrations through directly feeding those expert trajectories into memory of off-policy reinforcement learning (e.g., DDPG), to train the policy with both demonstrations and explorations. The prioritized experience replay (Schaul et al. 2015) is used as a natural balance of the two sources of training data. DDPGfD can work on solvable simple tasks for reinforcement learning, while learning from sparse rewards on harder task requires more active exploration during training.

Nair et al. (2018) proposed a method based on DQfD and DDPGfD to have better learning efficiency for hard tasks where further exploration based on demonstrations matters. The policy loss is a combination of policy gradient loss and the behavioral cloning loss, which gives the gradients as follows:

$$\lambda_1 \nabla_\theta J - \lambda_2 \nabla_\theta L_{BC} \tag{8.37}$$

where the J is the general reinforcement learning objective (maximized) and L_{BC} (minimized) is the behavior cloning loss as defined at the beginning of this chapter.

Moreover, the Q-filter technique is applied in this method, which requires the behavioral cloning loss to be only applied to states where the learned critic $Q(s, a)$ determines that the demonstrator action is better than the actor action:

$$L_{BC} = \sum_{i=1}^{N_D} ||\pi(s_i|\theta_\pi) - a_i||^2 \mathbb{1}_{Q(s_i,a_i)>Q(s_i,\pi(s_i))} \tag{8.38}$$

where the N_D is number of samples in demonstration dataset and (s_i, a_i) are sampled from the demonstration dataset. This ensures the policy to explore better actions other than being restricted by the demonstration data.

Using the same approach, QT-Opt (Kalashnikov et al. 2018) and Quantile QT-Opt (Bodnar et al. 2019) algorithms also apply a combination of on-policy buffer and an off-policy demonstration buffer to conduct off-line learning with actor-free CE method with DQN, which achieves the state-of-the-art performances in real-world robot learning tasks based on images.

8.7.2 Normalized Actor-Critic

Normalized actor-critic (NAC) (Gao et al. 2018) is another method for efficient reinforcement learning with demonstrations, and it pretrains a policy as initialization for a refinement reinforcement learning process. The key difference of NAC from other methods is that it uses exactly the same objective for the processes of pre-training an initialization policy with demonstrations and the refinement

reinforcement learning process (not like a combination of supervised loss and reinforcement learning loss in DQfD, or two separate training processes with different loss in policy replacement and behavioral cloning methods), which makes NAC robust to suboptimal demonstrations data.

The NAC method is similar to methods of DDPGfD or DQfD, but trains the policy sequentially from demonstrations and samples from interactions instead of using samples from both sources at the same time.

8.7.3 Reward Shaping with Demonstrations

Reward shaping with demonstrations (Brys et al. 2015) is a method focusing on the initialization of value function instead of the action policy for reinforcement learning. It provides the agent an intermediate reward for enriching the sparse reward signals:

$$R_F(s, a, s') = R(s, a, s') + F^D(s, a, s') \tag{8.39}$$

where the shaping reward F^D from demonstrations D is defined with potential function ϕ in the following form to guarantee the convergence:

$$F^D(s, a, s', a') = \gamma \phi^D(s', a') - \phi^D(s, a) \tag{8.40}$$

and ϕ^D is defined as:

$$\phi^D(s, a) = \max_{(s^d, a)} e^{-\frac{1}{2}\left(s - s^d\right)^T \Sigma^{-1}\left(s - s^d\right)} \tag{8.41}$$

which is to maximize the value of the potential for the state s that is most similar as the demonstration state s^d. The optimized potential function is used to initialize the action-value function Q in reinforcement learning:

$$Q_0(s, a) = \phi^D(s, a) \tag{8.42}$$

The intuition of the reward shaping method is to bias the exploration in favor of those state-action pairs in demonstrations or close to those in demonstrations for accelerating the training process of reinforcement learning. Reward shaping provides a good approach of initialization for the value-evaluation function in reinforcement learning process.

Other methods like unsupervised perceptual rewards (Sermanet et al. 2016) also learn a dense and smooth reward functions with the demonstrations, using features in a pre-trained deep model.

8.8 Summary

Due to the low learning efficiency challenge of reinforcement learning as mentioned in Chap. 7, in this chapter, we introduce imitation learning (IL) as one potential solution leveraging the expert demonstration. The overall chapter is summarized into several main categories. The behavior cloning methods introduced in Sect. 8.2 are the most straightforward way of imitation learning in a supervised learning manner, which can be further combined with reinforcement learning like as an initialization introduced in Sect. 8.6. A more advanced way of combining the imitation learning with reinforcement learning is through IRL by recovering a reward function explicitly or implicitly from demonstration, as in Sect. 8.3. Methods like MaxEnt can explicitly learn the reward function but with heavy computation cost. Other methods in the generative adversarial approach like GAIL, GAN-GCL, AIRL learn in a more efficient way. Another problem is if the actions are missing in the demonstration dataset, like learning from the videos only, how to work properly with imitation learning? This falls into the category of IfO as in Sect. 8.4. Since the IfO problem is from another perspective, those methods mentioned before like BC, IRL can also be applied in IfO with proper modifications. The methods in IfO are generally summarized in model-based and model-free categories. The model-based method learns the dynamics model from samples, and it can actually recover the observation-only demonstration dataset with actions through leveraging the action-state relationship in the model, explicitly or implicitly. Then, the regular imitation learning methods can be applied if the actions are recovered explicitly. Methods like RIDM, BCO, ILPO, etc., fall into this model-based IfO category. For the model-free methods in IfO, either the reward engineering or generative adversarial approach can be applied for providing the reward function to enable reinforcement learning. Methods like OptionGAN, FAIL, AGAIL, and so on are in the category of generative adversarial IfO, while TCN and some other methods are in the category of reward engineering IfO. The two categories here in IfO actually also apply for general IL, like GAIL as a generative adversarial method and recently proposed contrastive forward dynamics (CFD) (Jeong et al. 2019) as a reward-engineering method for learning from demonstration with both observations and actions in IL. Then the probabilistic methods including GMR, GPR, and DMR-based GP are introduced as an alternative for general IL, with high-efficiency learning for relatively low-dimensional cases, as in Sect. 8.5. Finally some other approaches like DDPGfD and DQfD for feeding demonstration data into replay buffer in off-policy reinforcement learning and so on are introduced in Sect. 8.7. The research area of imitation learning is still very active as an efficient approach for solving learning problems, with an organic combination with reinforcement learning.

References

Abbeel P, Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the twenty-first international conference on machine learning. ACM, New York, p 1

Aytar Y, Pfaff T, Budden D, Paine T, Wang Z, de Freitas N (2018) Playing hard exploration games by watching YouTube. In: Advances in neural information processing systems, pp 2930–2941

Blau T, Ott L, Ramos F (2018) Improving reinforcement learning pre-training with variational dropout. In: 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, Piscataway, pp 4115–4122

Bodnar C, Li A, Hausman K, Pastor P, Kalakrishnan M (2019) Quantile QT-Opt for risk-aware vision-based robotic grasping. Preprint. arXiv:191002787

Brys T, Harutyunyan A, Suay HB, Chernova S, Taylor ME, Nowé A (2015) Reinforcement learning from demonstration through shaping. In: Twenty-fourth international joint conference on artificial intelligence

Calinon S (2016) A tutorial on task-parameterized movement learning and retrieval. Intel Serv Robot 9(1):1–29

Duan Y, Andrychowicz M, Stadie B, Ho OJ, Schneider J, Sutskever I, Abbeel P, Zaremba W (2017) One-shot imitation learning. In: Advances in neural information processing systems, pp 1087–1098

Dwibedi D, Tompson J, Lynch C, Sermanet P (2018) Learning actionable representations from visual observations. In: 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, Piscataway, pp 1577–1584

Edwards AD, Sahni H, Schroecker Y, Isbell CL (2018) Imitating latent policies from observation. Preprint. arXiv:180507914

Eysenbach B, Gupta A, Ibarz J, Levine S (2018) Diversity is all you need: learning skills without a reward function. Preprint. arXiv:180206070

Finn C, Christiano P, Abbeel P, Levine S (2016a) A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. Preprint. arXiv:161103852

Finn C, Levine S, Abbeel P (2016b) Guided cost learning: deep inverse optimal control via policy optimization. In: International conference on machine learning, pp 49–58

Fu J, Luo K, Levine S (2017) Learning robust rewards with adversarial inverse reinforcement learning. Preprint. arXiv:171011248

Gao Y, Lin J, Yu F, Levine S, Darrell T, et al (2018) Reinforcement learning from imperfect demonstrations. Preprint. arXiv:180205313

Goo W, Niekum S (2019) One-shot learning of multi-step tasks from observation via activity localization in auxiliary video. In: 2019 international conference on robotics and automation (ICRA). IEEE, Piscataway, pp 7755–7761

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Proceedings of the neural information processing systems (Advances in neural information processing systems) conference

Guo X, Chang S, Yu M, Tesauro G, Campbell M (2019) Hybrid reinforcement learning with expert state sequences. Preprint. arXiv:190304110

Gupta A, Devin C, Liu Y, Abbeel P, Levine S (2017) Learning invariant feature spaces to transfer skills with reinforcement learning. Preprint. arXiv:170302949

Hanna JP, Stone P (2017) Grounded action transformation for robot learning in simulation. In: Thirty-first AAAI conference on artificial intelligence

Hausman K, Chebotar Y, Schaal S, Sukhatme G, Lim JJ (2017) Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In: Advances in neural information processing systems, pp 1235–1245

Henderson P, Chang WD, Bacon PL, Meger D, Pineau J, Precup D (2018) OptionGAN: learning joint reward-policy options using generative adversarial inverse reinforcement learning. In: Thirty-second AAAI conference on artificial intelligence

Hester T, Vecerik M, Pietquin O, Lanctot M, Schaul T, Piot B, Horgan D, Quan J, Sendonaris A, Osband I, et al (2018) Deep Q-learning from demonstrations. In: Thirty-second AAAI conference on artificial intelligence

Ho J, Ermon S (2016) Generative adversarial imitation learning. In: Advances in neural information processing systems, pp 4565–4573

Huang Y, Rozo L, Silvério J, Caldwell DG (2019) Kernelized movement primitives. Inter J Robot Res 38(7):833–852

Jaquier N, Ginsbourger D, Calinon S (2019) Learning from demonstration with model-based Gaussian process. Preprint. arXiv:191005005

Jeong R, Aytar Y, Khosid D, Zhou Y, Kay J, Lampe T, Bousmalis K, Nori F (2019) Self-supervised sim-to-real adaptation for visual robotic manipulation. Preprint. arXiv:191009470

Johannink T, Bahl S, Nair A, Luo J, Kumar A, Loskyll M, Ojea JA, Solowjow E, Levine S (2019) Residual reinforcement learning for robot control. In: 2019 international conference on robotics and automation (ICRA). IEEE, Piscataway, pp 6023–6029

Kalashnikov D, Irpan A, Pastor P, Ibarz J, Herzog A, Jang E, Quillen D, Holly E, Kalakrishnan M, Vanhoucke V, et al (2018) QT-Opt: scalable deep reinforcement learning for vision-based robotic manipulation. Preprint. arXiv:180610293

Kimura D, Chaudhury S, Tachibana R, Dasgupta S (2018) Internal model from observations for reward shaping. Preprint. arXiv:180601267

Liu Y, Gupta A, Abbeel P, Levine S (2018) Imitation from observation: learning to imitate behaviors from raw video via context translation. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, Piscataway, pp 1118–1125

Merel J, Tassa Y, Srinivasan S, Lemmon J, Wang Z, Wayne G, Heess N (2017) Learning human behaviors from motion capture by adversarial imitation. Preprint. arXiv:170702201

Misra I, Zitnick CL, Hebert M (2016) Shuffle and learn: unsupervised learning using temporal order verification. In: European conference on computer vision. Springer, Berlin, pp 527–544

Molchanov D, Ashukha A, Vetrov D (2017) Variational dropout sparsifies deep neural networks. In: Proceedings of the 34th international conference on machine learning, vol 70, JMLR.org, pp 2498–2507

Nair A, Chen D, Agrawal P, Isola P, Abbeel P, Malik J, Levine S (2017) Combining self-supervised learning and imitation for vision-based rope manipulation. In: 2017 IEEE international conference on robotics and automation (ICRA). IEEE, Piscataway, pp 2146–2153

Nair A, McGrew B, Andrychowicz M, Zaremba W, Abbeel P (2018) Overcoming exploration in reinforcement learning with demonstrations. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, Piscataway, pp 6292–6299

Ng AY, Harada D, Russell S (1999) Policy invariance under reward transformations: theory and application to reward shaping. In: Proceedings of the international conference on machine learning (ICML), vol 99, pp 278–287

Ng AY, Russell SJ, et al (2000) Algorithms for inverse reinforcement learning. In: Proceedings of the international conference on machine learning (ICML), vol 1, p 2

Paraschos A, Daniel C, Peters JR, Neumann G (2013) Probabilistic movement primitives. In: Advances in neural information processing systems, pp 2616–2624

Pastor P, Hoffmann H, Asfour T, Schaal S (2009) Learning and generalization of motor skills by learning from demonstration. In: 2009 IEEE international conference on robotics and automation. IEEE, Piscataway, pp 763–768

Pathak D, Mahmoudieh P, Luo G, Agrawal P, Chen D, Shentu Y, Shelhamer E, Malik J, Efros AA, Darrell T (2018) Zero-shot visual imitation. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 2050–2053

Pavse BS, Torabi F, Hanna JP, Warnell G, Stone P (2019) RIDM: reinforced inverse dynamics modeling for learning from a single observed demonstration. Preprint. arXiv:190607372

Puterman ML (2014) Markov decision processes: discrete stochastic dynamic programming. Wiley, Hoboken

Ross S, Bagnell D (2010) Efficient reductions for imitation learning. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp 661–668

Ross S, Gordon G, Bagnell D (2011) 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, pp 627–635

Russell SJ (1998) Learning agents for uncertain environments. In: The 11th annual conference on computational learning theory, vol 98, pp 101–103

Schaul T, Quan J, Antonoglou I, Silver D (2015) Prioritized experience replay. In: International conference on learning representations

Schneider M, Ertel W (2010) Robot learning by demonstration with local Gaussian process regression. In: 2010 IEEE/RSJ international conference on intelligent robots and systems. IEEE, Piscataway, pp 255–260

Sermanet P, Xu K, Levine S (2016) Unsupervised perceptual rewards for imitation learning. Preprint. arXiv:161206699

Sermanet P, Lynch C, Chebotar Y, Hsu J, Jang E, Schaal S, Levine S, Brain G (2018) Time-contrastive networks: self-supervised learning from video. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, Piscataway, pp 1134–1141

Sieb M, Xian Z, Huang A, Kroemer O, Fragkiadaki K (2019) Graph-structured visual imitation. Preprint. arXiv:190705518

Silver T, Allen K, Tenenbaum J, Kaelbling L (2018) Residual policy learning. Preprint. arXiv:181206298

Stadie BC, Abbeel P, Sutskever I (2017) Third-person imitation learning. Preprint. arXiv:170301703

Sun M, Ma X (2019) Adversarial imitation learning from incomplete demonstrations. Preprint. arXiv:190512310

Sun W, Vemula A, Boots B, Bagnell JA (2019) Provably efficient imitation learning from observation alone. Preprint. arXiv:190510948

Syed U, Bowling M, Schapire RE (2008) Apprenticeship learning using linear programming. In: Proceedings of the 25th international conference on machine learning. ACM, New York, pp 1032–1039

Tassa Y, Erez T, Todorov E (2012) Synthesis and stabilization of complex behaviors through online trajectory optimization. In: 2012 IEEE/RSJ international conference on intelligent robots and systems. IEEE, Piscataway, pp 4906–4913

Torabi F, Warnell G, Stone P (2018a) Behavioral cloning from observation. Preprint. arXiv:180501954

Torabi F, Warnell G, Stone P (2018b) Generative adversarial imitation from observation. Preprint. arXiv:180706158

Torabi F, Geiger S, Warnell G, Stone P (2019a) Sample-efficient adversarial imitation learning from observation. Preprint. arXiv:190607374

Torabi F, Warnell G, Stone P (2019b) Adversarial imitation learning from state-only demonstrations. In: Proceedings of the 18th international conference on autonomous agents and multiagent systems, international foundation for autonomous agents and multiagent systems, pp 2229–2231

Torabi F, Warnell G, Stone P (2019c) Imitation learning from video by leveraging proprioception. Preprint. arXiv:190509335

Torabi F, Warnell G, Stone P (2019d) Recent advances in imitation learning from observation. Preprint. arXiv:190513566

Večerík M, Hester T, Scholz J, Wang F, Pietquin O, Piot B, Heess N, Rothörl T, Lampe T, Riedmiller M (2017) Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. Preprint. arXiv:170708817

Ziebart BD, Maas AL, Bagnell JA, Dey AK (2008) Maximum entropy inverse reinforcement learning. In: Proceedings of the AAAI conference on artificial intelligence, Chicago, vol 8, pp 1433–1438

Ziebart BD, Bagnell JA, Dey AK (2010) Modeling interaction via the principle of maximum causal entropy. In: Proceedings of the 27th international conference on international conference on machine learning

Żołna K, Rostamzadeh N, Bengio Y, Ahn S, Pinheiro PO (2018) Reinforced imitation learning from observations