An Algorithmic Perspective on Imitation Learning

Takayuki Osa
University of Tokyo
osa@edu.k.u-tokyo.ac.jp

Joni Pajarinen
Technische Universität Darmstadt
pajarinen@ias.tu-darmstadt.de

Gerhard Neumann
University of Lincoln
gneumann@lincoln.ac.uk

J. Andrew Bagnell
Carnegie Mellon University
dbagnell2@andrew.cmu.edu

Pieter Abbeel
University of California, Berkeley
pabbeel@cs.berkeley.edu

Jan Peters
Technische Universität Darmstadt
mail@jan-peters.net

Full text available at: http://dx.doi.org/10.1561/2300000053
Editorial Scope

Topics

Foundations and Trends® in Robotics publishes survey and tutorial articles in the following topics:

- Mathematical modelling
- Kinematics
- Dynamics
- Estimation methods
- Artificial intelligence in robotics
- Software systems and architectures
- Sensors and estimation
- Planning and control
- Human-robot interaction
- Industrial robotics
- Service robotics

Information for Librarians

Foundations and Trends® in Robotics, 2017, Volume 7, 4 issues. ISSN paper version 1935-8253. ISSN online version 1935-8261. Also available as a combined paper and online subscription.
An Algorithmic Perspective on Imitation Learning

Takayuki Osa
University of Tokyo
osa@edu.k.u-tokyo.ac.jp

Joni Pajarinen
Technische Universität Darmstadt
pajarinen@ias.tu-darmstadt.de

Gerhard Neumann
University of Lincoln
gneumann@lincoln.ac.uk

J. Andrew Bagnell
Carnegie Mellon University
dbagnell2@andrew.cmu.edu

Pieter Abbeel
University of California, Berkeley
pabbeel@cs.berkeley.edu

Jan Peters
Technische Universität Darmstadt
mail@jan-peters.net
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3 Model-Free and Model-Based Behavioral Cloning Methods</td>
<td>52</td>
</tr>
<tr>
<td>3.4 Model-Free Behavioral Cloning Methods in Action-State</td>
<td>53</td>
</tr>
<tr>
<td>3.5 Model-Free Behavioral Cloning for Learning Trajectories</td>
<td>64</td>
</tr>
<tr>
<td>3.6 Model-Free Behavioral Cloning for Task-Level Planning</td>
<td>93</td>
</tr>
<tr>
<td>3.7 Model-Based Behavioral Cloning Methods</td>
<td>99</td>
</tr>
<tr>
<td>3.8 Robot Applications with Model-Free BC Methods</td>
<td>107</td>
</tr>
<tr>
<td>3.9 Robot Applications with Model-Based BC Methods</td>
<td>111</td>
</tr>
<tr>
<td>4 Inverse Reinforcement Learning</td>
<td>115</td>
</tr>
<tr>
<td>4.1 Problem Statement</td>
<td>116</td>
</tr>
<tr>
<td>4.2 Model-Based and Model-Free IRL Methods</td>
<td>118</td>
</tr>
<tr>
<td>4.3 Design Choices for Inverse Reinforcement Learning Methods</td>
<td>118</td>
</tr>
<tr>
<td>4.4 Model-Based Inverse Reinforcement Learning Methods</td>
<td>120</td>
</tr>
<tr>
<td>4.5 Model-Free Inverse Reinforcement Learning Methods</td>
<td>136</td>
</tr>
<tr>
<td>4.6 Interpretation of IRL with the Maximum Entropy Principle</td>
<td>139</td>
</tr>
<tr>
<td>4.7 Inverse Reinforcement Learning under Partial Observability</td>
<td>141</td>
</tr>
<tr>
<td>4.8 Robot Applications with IRL Methods</td>
<td>145</td>
</tr>
<tr>
<td>5 Challenges in Imitation Learning for Robotics</td>
<td>150</td>
</tr>
<tr>
<td>5.1 Behavioral Cloning vs Inverse Reinforcement Learning</td>
<td>150</td>
</tr>
<tr>
<td>5.2 Open Questions in Imitation Learning</td>
<td>152</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>159</td>
</tr>
<tr>
<td>References</td>
<td>160</td>
</tr>
</tbody>
</table>
Abstract

As robots and other intelligent agents move from simple environments and problems to more complex, unstructured settings, manually programming their behavior has become increasingly challenging and expensive. Often, it is easier for a teacher to demonstrate a desired behavior rather than attempt to manually engineer it. This process of learning from demonstrations, and the study of algorithms to do so, is called imitation learning. This work provides an introduction to imitation learning. It covers the underlying assumptions, approaches, and how they relate; the rich set of algorithms developed to tackle the problem; and advice on effective tools and implementation.

We intend this paper to serve two audiences. First, we want to familiarize machine learning experts with the challenges of imitation learning, particularly those arising in robotics, and the interesting theoretical and practical distinctions between it and more familiar frameworks like statistical supervised learning theory and reinforcement learning. Second, we want to give roboticists and experts in applied artificial intelligence a broader appreciation for the frameworks and tools available for imitation learning.

We organize our work by dividing imitation learning into directly replicating desired behavior (sometimes called behavioral cloning [Bain and Sammut, 1996]) and learning the hidden objectives of the desired behavior from demonstrations (called inverse optimal control [Kalman, 1964] or inverse reinforcement learning [Russell, 1998]). In addition to method analysis, we discuss the design decisions a practitioner must make when selecting an imitation learning approach. Moreover, application examples—such as robots that play table tennis [Kober and Peters, 2009] and programs that play the game of Go [Silver et al., 2016]—illustrate the properties and motivations behind different forms of imitation learning. We conclude by presenting a set of open questions and point towards possible future research directions.

Programming autonomous behavior in machines and robots traditionally requires a specific set of skills and knowledge. However, human experts know how to demonstrate the desired task even if they do not know how to program the necessary behavior in a machine or robot. The purpose of imitation learning is to efficiently learn a desired behavior by imitating an expert’s behavior. The application of imitation learning is not limited to physical systems. It can be a powerful tool to design autonomous behavior in systems such as web sites, computer games, and mobile applications. Any system that requires autonomous behavior similar to human experts can benefit from imitation learning.

However, imitation learning may be essential for robotics. It is now considered to be a key technology for applications such as manufacturing, elder care, and the service industry. These robots will be expected to work closely with humans in a dramatic shift from prior uses of robots. Powerful robotic manipulators are dangerous and have therefore been used mainly in constrained, predefined industrial applications; employees must undergo special training before working with them. This is changing due to recent advances in robotics from compute to the use of light, compliant, and safe robotic manipulators. They
1.1. **Key successes in Imitation Learning**

are ideal for applications where robots work alongside people, such as collaborating with human operators and reducing the physical workload of care givers. These applications require efficient, intuitive ways to teach robots the motions they need to perform from domain experts who may not possess special skills or knowledge about robotics.

In recent years, imitation learning has been investigated as a way to efficiently and intuitively program autonomous behavior [Schaal, 1999, Argall et al., 2009, Billard et al., 2008, Billard and Grollman, 2013, Bagnell, 2015, Billard et al., 2016]. In imitation learning, a human demonstrates how to perform a task. A robotic system learns a policy to execute the given task by imitating the demonstrated motions. Numerous imitation learning methods have been developed and imitation learning has become a gigantic field of research. As a consequence, capturing the entire field of imitation learning is not a trivial task.

The purpose of this survey is to provide a structural understanding of existing imitation learning methods and its relationship with other fields from supervised learning to control theory. We will describe what has been developed in the field of imitation learning and what should be developed in the future.

1.1 **Key successes in Imitation Learning**

One of the earliest and most well-known imitation learning success stories was the autonomous driving project Autonomous Land Vehicle In a Neural Network (ALVINN) at Carnegie Mellon University [Pomerleau, 1988]. In ALVINN, a neural network learned how to map input images to discrete actions in order to drive a vehicle. ALVINN’s neural network had one hidden layer with five units. Its input layer had 30 by 32 units; its output layer had 30 units. Although the structure of this network was simple compared to modern neural networks with millions of parameters, the system succeeded at driving autonomously across the North American continent.

The Kendama robot developed by Miyamoto et al. [1996] is another successful application of imitation learning. In the early days of imitation learning, roboticists were mainly interested in teaching
Introduction

higher-level tasks from human demonstrations, such as “pick,” “move,”
and “place” Kang and Ikeuchi [1993], Kuniyoshi et al. [1994]. In those
settings, lower-level tasks were often considered to be simple, point-to-
point motions. In the late 1990s, this focus shifted from task-level plan-
ing to trajectory-level planning. The term “learning from demonstra-
tion” has become very popular since its use by S. Schaal and G. Atke-
son [Schaal, 1997, Atkeson and Schaal, 1997]. Since then, learning robot
motions has been a key domain of imitation learning.

Recently, learning from human demonstrations has benefited from
developments in deep neural networks. Recurrent neural networks such
as long short-term memory (LSTM) networks [Hochreiter and Schmid-
huber, 1997] have played a significant role in demonstrating how
to succeed in many previously difficult sequential tasks by learning
from demonstrated data. This includes tasks for generating handwriting
[Chung et al., 2015], natural language [Wen et al., 2015], or image
captions [Karpathy and Fei-Fei, 2015]. Furthermore, AlphaGo, the al-
gorithm which was able to beat a human Go master and which we
discuss in more detail in §3.4.2 initializes a deep neural network pol-
cy from human demonstrations [Silver et al., 2016]. Often these recent
approaches require a large amount of data. In §3 we will discuss how
to learn from data to reproduce observed behavior in specific problem
settings.

1.2 Imitation Learning from the Point of View of Robotics

Imitation learning is a class of methods that reproduces desired be-
behavior based on expert demonstrations. In many cases, the experts are
human operators and the learners are robotic systems. Thus, imitation
learning is a technique that enables skills to be transferred from hu-
mans to robotic systems. To perform imitation learning, we need to
develop a system that records demonstrations by experts and learns a
policy to reproduce the demonstrated behavior from the recorded data.
For this purpose, we need to answer the following questions.
1.2. *Imitation Learning from the Point of View of Robotics*

**General Aspects:**

1. **Why and when should imitation learning be used?** This question clarifies the motivation for using imitation learning and what we should do with it.

2. **Who should demonstrate?** In many cases, the experts are human operators. Many imitation learning methods implicitly assume that demonstrations are provided by a single expert. When multiple experts are available, we need to decide which one should be imitated or how we can incorporate demonstrations from multiple experts.

3. **How should we record data of the expert demonstrations?** There are multiple ways of recording the behavior of experts. For example, motion capture systems and teleoperated robotic systems record data from expert behavior. This choice is closely related to the embodiment problem between experts and learners, which will be discussed in §3.7.1.

4. **What should we imitate?** The recorded data often includes redundant information about expert behavior. In such cases, features appropriate for the desired behavior should be selected. Meanwhile, the recorded data also includes unnecessary motions, which should not be imitated. The data must be segmented to extract the motions to be imitated.

**Algorithmic Aspects:**

5. **How should we represent the policy?** Expert behavior can be represented using methods such as symbolic representation, trajectory-based representation, and state-action space representation. The choice depends largely on the design of the entire system.

6. **How should we learn the policy?** Many algorithms for learning the policy have been developed over the past several decades. The choice of the algorithm for learning the policy is closely related to the choice of policy representation.
With regard to the first four questions, several survey papers on imitation learning [Argall et al., 2009, Billard et al., 2008, Billard and Grollman, 2013, Billard et al., 2016], provide a taxonomy of imitation learning from the perspective of robotics. [Argall et al., 2009] indicate that it is essential to design an imitation learning system by considering the correspondence between the expert and the learner, data acquisition methods, and limitations of the demonstration dataset. [Billard et al., 2008, 2016] provide an overview of imitation learning methods and highlight techniques for trajectory learning. However, none of the previous review articles focused on the design decisions needed to develop new imitation learning algorithms to enable answering questions five and six related to the algorithmic aspects discussed above. In addition, these articles did not discuss the algorithmic details of existing methods because the enormous amount of prior work on imitation learning makes it challenging to cover the entire range of previous studies.

In this survey, we provide an overview of existing methods from the algorithmic point of view, which will be useful for both readers beginning the practice of imitation learning and readers who want to achieve a deeper understanding of the theoretical aspects of imitation learning. We discuss the design choices which one should consider in order to develop novel imitation learning algorithms. Although our survey cannot be exhaustive, we discuss the algorithmic details of existing algorithms as much as possible, which will be useful to readers who want to implement imitation learning techniques. Additionally, we develop an information theoretic understanding of existing methods, which will help readers to understand how existing methods relate to each other and figure out how they could be extended.

Let us illustrate how different design choices of imitation learning algorithms can be made in different applications. Figure 1.1 shows three applications of imitation learning: 1) an RC helicopter, 2) robotic surgery, and 3) quadruped robot locomotion. In these applications, design of the policies for motion planning and control vary. [Abbeel et al., 2010] demonstrates acrobatic RC helicopter flight by learning from trajectories demonstrated by a human expert. In this system, the desired
1.2. Imitation Learning from the Point of View of Robotics

(a) Learning of acrobatic RC helicopter maneuvers \cite{Abbeel2010}. The trajectories for acrobatic flights are learned from a human expert’s demonstrations. To control the system with highly nonlinear dynamics, iterative learning control was used.

(b) Learning with a teleoperated system \cite{Osa2014} where a position/velocity controller is available. To generalize the trajectory to different situations, a mapping from task situations to trajectories is learned from demonstrations under various situations.

(c) Learning quadruped robot locomotion \cite{Zucker2011}. The footstep planning was addressed as an optimization of the reward/cost function, which was recovered from the expert demonstrations. Learning the reward/cost function allows the footstep planning strategy to be generalized to different terrains.

**Figure 1.1:** Observations $y$ and control inputs $u$ for imitation learning in (a) helicopter flight, (b) surgery, and (c) locomotion. Motion planning is formulated in different ways in these examples.
trajectories of acrobatic flights were learned from demonstrations with a supervised learning method. Osa et al. [2017b] also learned trajectories for autonomous knot tying from demonstrations by a human expert. To generalize a trajectory, Osa et al. [2017b] learned a direct mapping from task situations (contexts) to trajectories using demonstrations recorded under various situations. Contrary to Abbeel et al. [2010], Osa et al. [2017b], Zucker et al. [2011] formulated footstep planning for quadruped robot locomotion as an optimization of the reward/cost function. The reward/cost function was recovered from demonstrations. In Zucker et al. [2011], learning the reward/cost function as a function of terrain features enables the footstep planning strategy to be generalized to different terrains. Learning such reward/cost functions for manipulation tasks like as knot-tying [Osa et al., 2017b] is not trivial, since complex manipulation tasks often require nonlinear reward/cost functions.

Methods for learning policies also differ between applications. The observation and control inputs of the RC helicopter system are much noisier than those of the other two systems, and its dynamics are highly nonlinear [Abbeel et al., 2010]. Therefore, it is essential to estimate the true state using various sensory information and learn an adaptive controller through iterations of trials to achieve acrobatic RC helicopter flight. On the other hand, we can assume that the system state is precisely known and a position/velocity controller is available in the case of the tele-operation system in [Osa et al., 2014], which simplifies imitation learning significantly. In [Osa et al., 2014], the conditional trajectory distribution given a context can be learned with a simple regression method, and the planned trajectory can be executed by a standard velocity controller. In locomotion planning for a quadruped robot in Zucker et al. [2011], estimating the reward/cost function requires an iterative learning process with virtual simulation of the learned policy. As one can see from these examples, learning methods can be very different between applications.

To apply imitation learning, it is essential to identify the structure of the system, formulate a given problem, and design an algorithm to solve the problem efficiently. In this survey, we focus on the algorithmic aspects of imitation and discuss necessary design choices, exploring
1.3 Differences between Imitation Learning and Supervised Learning

various solutions proposed by previous studies.

In the rest of this chapter, we introduce several concepts in machine learning that are essential to understand imitation learning algorithms. We discuss the design choices of imitation learning algorithms in Chapter 2. We describe the details of behavioral cloning methods and inverse reinforcement learning methods in Chapters 3 and 4 respectively. To conclude, we list open questions of imitation learning in Chapter 5.

1.3 Key Differences between Imitation Learning and Supervised Learning

The imitation learning problem has special properties that distinguish it from the better known supervised learning setting [Shalev-Shwartz and Ben-David, 2014]: 1) the solution may have important structural properties including constraints (for example, robot joint limits), dynamic smoothness and stability, or leading to a coherent, multi-step plan [Bagnell, 2015]; 2) the interaction between the learner’s decisions and its own input distribution (an on-policy versus off-policy distinction), and 3) the increased necessity of minimizing the typically high cost of gathering examples.

As we learn a policy \( \pi \) from a dataset \( \mathcal{D} \), imitation learning is closely related to supervised learning, and is particularly related to the field of structured prediction [Daumé III et al., 2009, Ratliff et al., 2006a, Taskar, 2005], where the task is to learn a mapping from inputs \( \mathbf{x} \) to a complex, structured output \( \mathbf{y} \) (plans, parse trees, complex motions). Reductions of structured prediction to sequential decision [Daumé III et al., 2009], and reductions of imitation learning to structured prediction [Ratliff et al., 2006b] show the close connection, and cross-fertilization between these research areas has been important for both. In practice, distinctions arise because of the structural properties of policies we attempt to imitate, and the difficulty of "resetting" state and restarting predictions is too costly or even infeasible in most imitation learning settings because a physical system is often involved.

In addition, it is often the case that the embodiments of the expert and the learner are different. For example, when transferring human skills to a humanoid robot, the motion captured from a human expert
Introduction

may be infeasible for the humanoid. In such a case, the demonstrated motion needs to be adapted to be feasible for the humanoid. This kind of adaptation is less common in the standard supervised learning.

In machine learning, the prediction problem where the source domain distribution and the target domain distribution are different is often referred to as “covariate shift” or “domain adaptation” [Sugiyama, 2015]. In imitation learning, the source domain corresponds to expert demonstrations and the target domain to learner reproductions. In imitation learning, the demonstration dataset does not cover all possible situations since collecting expert demonstrations to cover all situations is usually too expensive and time-consuming. As a result, the learner often encounters states which were not encountered by the expert during demonstrations, which means that the target domain distribution is different from the source distribution. Therefore, covariate shift or domain adaptation is closely related to imitation learning [Bagnell, 2015].

Imitation learning is also closely related to reinforcement learning (RL), which tries to obtain a policy that maximizes an expected reward [Sutton and Barto, 1998] signal. In RL, we employ a reward function that encourages a desired behavior. However, in imitation learning we often assume optimal (or at least “good”) expert demonstrations which are not available in basic reinforcement learning, and which provide prior knowledge that allows for dramatically more efficient methods. Recent work by Sun et al. [2017] demonstrates a potentially exponential decrease in sample complexity in learning a task by imitation rather than by trial-and-error reinforcement learning, and empirical results have long shown such benefits Silver et al. [2016] Kober and Peters [2009] Abbeel et al. [2010]. Moreover, in the imitation learning setting, as we detail below, we may or may not have access to a true reward function.

1.4 Insights for Machine Learning and Robotics Research

As imitation learning offers intuitive ways to program robotic motions by demonstrating the desired motion, imitation learning attracted interests from robotic researchers. The robotics community has devel-
1.5. Statistical Machine Learning Background

oped many imitation learning methods for motion planning and robot control. When planning a trajectory for a robotic system, it is often necessary to make sure that a planned trajectory satisfies some constraints such as smooth convergence to a new goal state. For this reason, robotics researchers have developed “custom” trajectory representations that explicitly satisfy constraints necessary for robotic applications. Machine learning techniques are often used as a part of such frameworks. However, robotics researchers need to be aware that rich set of algorithms have been developed by the machine learning community and some of new algorithms might eliminate the need for customizing policy or trajectory representation.

For machine learning researchers, imitation learning offers interesting practical and theoretical problems, which differ from standard supervised and reinforcement learning settings. Although imitation learning is closely related to structured prediction, it is often challenging to apply existing machine learning methods to imitation learning, especially robotic applications. In imitation learning, collecting demonstrations and performing rollouts are often expensive and time-consuming. Therefore, it is necessary to consider how to minimize these costs and perform learning efficiently. In addition, embodiments and observability of the learner and the expert are different in many applications. In such cases, the demonstrated motion needs to be adapted based on the learner’s embodiment and observability. These difficulties in imitation learning present new challenges to machine learning researchers.

1.5 Statistical Machine Learning Background

To understand imitation learning algorithms, familiarity with several concepts in statistical machine learning is essential. In this section, we briefly introduce the notation we use and these concepts.

1.5.1 Notation and Mathematical Formalization

Before introducing important concepts in machine learning, we introduce the notation in this article. Table 1.1 summarizes our notation. Throughout this survey, we use the bold style for vector values, and the
non-bold style for scalar values. Demonstrations by an expert are often given as a set of trajectories. In this case, the dataset of demonstrations is given by $D = \{\tau^0, \ldots, \tau^m\}$. We use the lower script to denote the time index; $x_t$ represents the state of the system at time step $t$. We review many methods that manipulate probability distributions in various ways. To make equations concise, the probability distribution induced by the experts’ policy is denoted by $q$, and the distribution induced by the learner’s policy is denoted by $p$. For example, $p(\tau)$ represents the probability distribution over trajectories induced by the learner’s policy. The term “action” is mainly used in machine learning community, and “control input” is mainly used in robotic community and control theory community. Since imitation learning methods have been developed in all of these communities, we use the word “action”.

**Table 1.1:** Table of Notation. We use a notation common in the control literature for states and controls.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>system state</td>
</tr>
<tr>
<td>$s$</td>
<td>context</td>
</tr>
<tr>
<td>$\phi$</td>
<td>feature vector</td>
</tr>
<tr>
<td>$u$</td>
<td>control input/action</td>
</tr>
<tr>
<td>$\tau$</td>
<td>trajectory</td>
</tr>
<tr>
<td>$\pi$</td>
<td>policy</td>
</tr>
<tr>
<td>$D$</td>
<td>dataset of demonstrations</td>
</tr>
<tr>
<td>$q$</td>
<td>probability distribution induced by an expert’s policy</td>
</tr>
<tr>
<td>$p$</td>
<td>probability distribution induced by a learner’s policy</td>
</tr>
<tr>
<td>$t$</td>
<td>time</td>
</tr>
<tr>
<td>$T$</td>
<td>finite horizon</td>
</tr>
<tr>
<td>$N$</td>
<td>number of demonstrations</td>
</tr>
<tr>
<td>$E$</td>
<td>superscript representing an expert e.g. $\pi^E$ denotes an expert’s policy</td>
</tr>
<tr>
<td>$L$</td>
<td>superscript representing a learner e.g. $\pi^L$ denotes a learner’s policy</td>
</tr>
<tr>
<td>demo</td>
<td>superscript representing a demonstration by an expert e.g. $\tau^{\text{demo}}$ denotes a trajectory demonstrated by an expert</td>
</tr>
</tbody>
</table>
1.5. Statistical Machine Learning Background

and “control input” interchangeably. We use the term “context” to refer to the condition relevant to the task. The context \( s \) can be the initial state of the system \( x_0 \) or the state of relevant objects. For instance, the position of the ball can be part of the context in a hitting-a-ball task. We use \( T \) to denote the finite horizon of the trajectory. Therefore, the total number of the time steps of a single trajectory is \( T + 1 \) in our notation.

1.5.2 Markov Property

A sequence of states \( x_0, \ldots, x_t \) is a Markov chain if at any time \( t \), the future states \( x_{t+1}, x_{t+2}, \ldots \) depend on the history \( x_0, \ldots, x_t \) only through the present state \( x_t \) [Serfozo 2009]. In other words, the next state \( x_{t+1} \) only depends on the current state \( x_t \) in a Markov chain. This property is called the Markov property.

1.5.3 Markov Decision Process

A Markov decision process (MDP) is a process that satisfies the Markov property. If the state and action spaces are finite, then it is called a finite Markov decision process (finite MDP) [Sutton and Barto 1998]. An MDP is defined as a tuple \( (X, U, P, \gamma, D, R) \). \( X \) is a finite set of states; \( U \) is a set of control inputs; \( P \) is a set of state transitions probabilities; \( \gamma \in [1, 0) \) is a discount factor; \( D \) is the initial-state distribution from which the initial state \( x_0 \) is drawn; and \( R : X \to \mathbb{R} \) is the reward function.

1.5.4 Entropy

Given the random variable \( x \) and its probability distribution \( p(x) \), the entropy

\[
H(p) = - \int p(x) \ln p(x) dx
\]

is defined as the amount of information conveyed by transmitting \( x \) [Bishop 2006]. Note that the entropy \( H(x) \) is a convex function.
### 1.5.5 Kullback-Leibler (KL) Divergence

In the field of information geometry, the KL divergence is used to quantify a difference between two probability distributions [Kullback and Leibler, 1951], i.e.,

\[
D_{\text{KL}}(p(x)||q(x)) = \int p(x) \ln \frac{p(x)}{q(x)} \, dx.
\]  

Since the KL divergence identifies a difference between two probability distributions, it is useful for cases in which stochastic policies are going to be learned, or stochastic trajectories result from a deterministic policy. Please note that the KL divergence is not symmetric, therefore \(D_{\text{KL}}(p||q) \neq D_{\text{KL}}(q||p)\). The KL divergence can be obtained as a Bregman divergence derived from the negative entropy [Amari, 2016] and is widely used as a measure in multiple imitation learning approaches.

### 1.5.6 Information and Moment Projections

One common approach to learning a policy from a dataset is to consider “projecting” that dataset onto the space of the policy model. Information theory emphasizes two kinds of projections: the Information(I)-projection and the Moment(M)-projection [Bishop, 2006]. Using the Kullback-Leibler (KL) divergence [Kullback and Leibler, 1951], the I-projection is

\[
p^* = \arg \min_p D_{\text{KL}}(p \parallel q),
\]  

and, the M-projection

\[
p^* = \arg \min_p D_{\text{KL}}(q \parallel p).
\]

As the KL divergence is not symmetric, these two projections result in different solutions when a given distribution is multi-modal as shown in Figure 1.2. While the M-projection averages over the several modes, the I-projection concentrates on a single mode. Performing the I-projection is often not straight-forward, although the M-projection can often be performed relatively easily by maximizing the likelihood with respect to a given training dataset [Bishop, 2006].

Full text available at: http://dx.doi.org/10.1561/2300000053
1.5. Statistical Machine Learning Background

![Image](attachment:image.png)

**Figure 1.2:** Illustration of I- and M- projections. Given a distribution with two modes as shown in black, M-projection will give a solution that averages over two modes as shown in red. On the contrary, I-projection will give a solution that concentrates on one of the modes.

1.5.7 The Maximum Entropy Principle

Let us consider a probability distribution $p(x)$ that matches the features of an unknown distribution $q$, i.e. it satisfies

$$E_p[\phi(x)] = E_q[\phi(x)],$$

where $q(x)$ is an unknown probability distribution and $E_q[\phi(x)]$, which is the expectation of a feature function $\phi(x)$, is available. As there are typically an infinite amount of such distributions, we need an additional constraint to obtain a unique solution [Amari, 2016].

The maximum entropy principle [Jaynes, 1957] suggests to choose a distribution that maximizes the entropy

$$H(p) = -\int p(x) \ln p(x) dx$$

among the distributions that satisfy $E_p[\phi(x)] = E_q[\phi(x)]$. From this constrained optimization program, the maximum entropy distribution can be computed as

$$p(x) \propto \exp (w^T \phi(x)), \quad (1.5)$$

where $w$ is a vector-valued Lagrangian multiplier for the feature matching constraint. While the maximum entropy principle does not directly translate into a practical algorithm, it uncovers an interesting observation. Every distribution that is in a log-linear representation given by Equation [1.5] is the maximum entropy distribution that can match specific feature expectations given by the feature vector $\phi(x)$. This is
true for typical distributions from the exponential family such as the Gaussian distribution, which is the maximum entropy distribution that matches first and second order moments. The notion of Maximum Entropy generalizes to Maximum Causal Entropy, which turns out to be a natural notion of uncertainty for dynamical systems [Ziebart et al., 2013].

1.5.8 Background: Reinforcement Learning

Reinforcement learning is a class of methods that autonomously learns policies through iterations of trials and evaluations. The goal of reinforcement learning is to learn a policy $\pi$ that maps the state of the system to the control input so as to maximize the expected reward $J(\pi)$. The reward $r_t$ represents the quality of the given state, action or trajectory at time $t$. For example, $r_t$ could be large when a robot is close to the desired trajectory and small when the robot is far from the trajectory, or, $r_t$ could be large for stable robot grasps and small for unstable ones. With a finite horizon $T$, the expected return is given by the accumulation of the reward at each time step,

$$J(\pi) = \mathbb{E} \left[ \sum_{t=0}^{T} r_t \bigg| \pi \right].$$  \hspace{1cm} (1.6)

Alternatively, the discounted accumulated reward is used for the infinite horizon scenario, i.e.,

$$J(\pi) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r_t \bigg| \pi \right],$$  \hspace{1cm} (1.7)

where the discounted factor $\gamma$ controls the trade-off between shorter term rewards and longer term rewards. The desired policy $\pi^*$ is given by

$$\pi^* = \arg \max_{\pi} J(\pi).$$  \hspace{1cm} (1.8)

The value of a state $x$ under a policy $\pi$ can be computed as the expected reward when starting from $x$ and following $\pi$

$$V^\pi(x) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r_t \bigg| x_0 = x, \pi \right].$$  \hspace{1cm} (1.9)
1.6. Formulation of the Imitation Learning Problem

$V^\pi(x_t)$ is often called the value function [Sutton and Barto, 1998]. Likewise, the value of taking action $u$ in state $x$ under a policy $\pi$ can be computed as the expected reward when starting from the action $u$ in a state $x$ and thereafter following policy $\pi$

$$Q^\pi(x, u) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r_t \bigg| x_0 = x, u_0 = u, \pi \right].$$  \hspace{1cm} (1.10)

$Q^\pi(x_t, u_t)$ is often called the action-value function [Sutton and Barto, 1998].

For an overview of reinforcement learning methods, please refer to [Sutton and Barto, 1998, Szepesvari, 2010, Wiering and van Otterlo, 2012, Sugiyama et al., 2013] and for an overview in reinforcement learning in robotics, please refer to [Kober et al., 2013, Deisenroth et al., 2013b].

1.6 Formulation of the Imitation Learning Problem

The goal of imitation learning is to learn a policy that reproduces the behavior of experts who demonstrate how to perform the desired task. Suppose that the behavior of the expert demonstrator (or the learner itself) can be observed as a trajectory $\tau = [\phi_0, ..., \phi_T]$, which is a sequence of features $\phi$. The features $\phi$, which can be the state of the robotic system or any other measurements, can be chosen according to the given problem. Please note that the features $\phi$ do not have to be manually specified, and $\phi$ could be as general as simply pixels in raw images.

Often, the demonstrations are recorded under different conditions, for example, grasping an object at different locations. We will refer to these task conditions as context vector $s$ of the task which is stored together with the feature trajectories. The context $s$ can contain any information relevant to the task, e.g., the initial state of the robotic system or positions of target objects. Note that, as the context describes the current task, it is typically fixed during task execution and the only dynamic aspects of the problem are the state features $\phi_t$. Optionally, a reward signal $r$ that the expert is trying to optimize is also available in some problem settings [Ross and Bagnell, 2014].
In imitation learning, we collect a dataset of demonstrations $D = \{(\tau_i, s_i, r_i)\}_{i=1}^N$ that consists of pairs of trajectories $\tau$, contexts $s$, and optionally reward signals $r$. The data collection process can be both offline and online. Using the collected dataset $D$, a common optimization-based strategy learns a policy $\pi^*$ that satisfies

$$\pi^* = \arg \min_{\phi} D(q(\phi), p(\phi)), \quad (1.11)$$

where $q(\phi)$ is the distribution of the features induced by the experts’ policy, $p(\phi)$ is the distribution of the features induced by the learner, and $D(q, p)$ is a similarity measure between $q$ and $p$. Both offline and online learning scenarios of this problem have been considered [Ross et al., 2011]. Please note that, when the dataset contains demonstrations of multiple tasks and the contexts include information of each task, this problem can be considered multitask learning as in recent work by [Duan et al., 2017, Finn et al., 2017a, b].

In addition, we often have access to an environment such as a simulator or a physical robotic system where we can perform and evaluate a policy through interaction. This simulator can be used to gather new data and iteratively improve the policy to better match the demonstrations.
References


References


References


References


References


References


http://arxiv.org/abs/1604.06508


References


References


Full text available at: http://dx.doi.org/10.1561/2300000053


References


References


