ALS Seminar - Imitation Learning

Michael Kutschke

Abstract

Machine Learning is an important tool for developing complex systems. In particular, Reinforcement Learning has been successfull in learning robot behavior. However, it can be difficult to define appropriate reward functions. Especially in robotics, one has to deal with high dimensionality in the state/action space. Imitation of experts is a fundamental part of learning in nature. Imitation, in Machine Learning, should be suited to reduce the state/action space that needs to be explored and thus speed up learning. However, while imitation is a concept that is easy to grasp, the specific implementations differ from case to case as each learning problem comes with a different set of problems. This seminar thesis will discuss some difficulties and solutions to these problems as well as two case studies as an example of what imitation learning can achieve.

1 Motivation

In certain domains, like robotics, there are problems where the solutions are difficult to handcraft. These problems can include complex motoric behavior. The difficulty can both mean the inability to express the solution because doing so by hand would take too much time/effort, as well as the inability because the solution is unknown. In such cases, machine learning techniques are applied. They overcome the need of "hard-coding" complex behaviors, thus potentially allowing for faster learning of more behavior than would be possible if each behavior was hard-coded. One such machine learning technique is reinforcement learning. It is used for sequential decision problems to learn a control policy that assigns every state a corresponding optimal action with regards to rewards associated with the states that are traversed.

1.1 Markov Decision Problem

Assuming the markov property holds that states only depend on the previous state and action taken in that state, a sequential decision problem can be formulated as Markov Decision Problem. We use the definition of [Russell and Norvig, 2004]. A Markov Decision Problem is defined through

- a set S of states
- a set A of actions
- a transition model $T(s, a, s') = p(s'|s, a)$ which is the probability of being in state $s' \in S$ after using action $a \in A$ in state $s \in S$
- a reward function $R : S \to \mathbb{R}$, which assigns a reward to each state (can be zero or even negative)

Given a decay value $\gamma \in (0, 1)$ which is a trade-off between rewards in the near and far future, the expected cumulative reward (or use/value) of a state is given by the Bellman-equation:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s')U(s') \tag{1}$$

which is the reward of the state itself plus the weighted maximal expected value of successor states s', maximized with respect to the possible actions a that the agent can choose in the current state s. This corresponds to maximizing the expected cumulative reward if $\gamma = 1$, otherwise it maximizes the expected cumulative reward weighted by the number of steps necessary to reach the reward (which may be different from maximizing the total cumulative reward!).

The solution of a Markov Decision Problem is an optimal policy $\pi : S \to A$ that, in every state, chooses the action with the highest expected value or use U. That is, such an optimal policy π^* has to fulfill

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s, a, s')U(s') \tag{2}$$

The problem is that some of the functions U in general are unknown, and equation 1 forms a non-linear equality system that cannot be easily solved directly. Therefore learning is employed to approximate U.

1.2 Reinforcement learning

In reinforcement learning, U is approximated iteratively. This iteration step is also known as the Bellman-Update ([Russell and Norvig, 2004]):

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s, a, s')U_i(s') \tag{3}$$

The resulting algorithm is called Value-Iteration. In the limit $i \to \infty$, U converges to it's true value as defined in equation 1 ([Russell and Norvig, 2004]). Other formulations yield more efficient algorithms, but those still use the same idea and share the same properties.

If the state space is large, however, it becomes very costly to approximate $U(s)$ for every s equally well. In most cases, there are only few states that are "interesting" in the sense that the agent will actually visit them in practice. Other states give negative rewards, and it is enough to know that they are "bad", it is not necessary to know "how bad". Therefore, in general one iteration corresponds to one exploratory walk through the state-space updating only states along the path. Obviously, this approximation can lead to suboptimal policies if the agent acts too greedy and does not explore enough of the state space. For example, if the agent never visits states with negative reward, it might not realize that this state in fact leads to regions of higher reward than possible with the current policy.

1.3 A problem with reinforcement learning

In many problems, including robotics, the states, as well as the actions, are described by several different variables, e.g. joint positions and angles. Thus, the state and action spaces can quickly become high-dimensional and thus intractable. Often, the states and actions are not discrete but continuous. Reinforcement learning potentially needs to explore large spaces before any proper policy is learned. Hand-crafted reward functions tend to be sparse, therefore an agent often has to explore significant portions of the space without getting feedback until it finds states granting a reward (positive or negative). Last but not least, as exploration is very costly, one might easily end up with a suboptimal policy because of insufficient exploration.

1.4 Imitation as part of natural learning processes

How does learning work in nature? When humans learn, they often have a teacher that demonstrates the desired behavior. Explanations are helpful for learning, but the learning is essentially faster when there are demonstrations that can be imitated. [Schaal, 1999] defines true imitation as opposed to other phenomena leading to a certain bias in behavior towards earlier demonstrated behavior through three criteria, that must all be met:

1. The imitated behavior is new to the imitator
2. The same task as that of the demonstrator is performed

3. The same task goal is accomplished

These criteria are all met in machine learning problems. It has been shown([Schaal, 1999]) that certain animals are unable to imitate perceived behavior, and that imitation is thus a sign of higher intelligence. Can machine learning be enhanced through imitation in a similar way that the human learning is enhanced through imitation? Also, imitation is a "language" that non-experts can understand and "speak", thus opening up the area of machine/robot learning to non-expert. This would rapidly speed up the development of every-day robots that fold clothes as well as special robots trained for special tasks in, e.g. human rescue.

2 Imitation learning

When talking about imitation, it is often referred to movement imitation. Robot control suffers from dimensionality, as there are often multiple joints and motors involved in a movement or motion. These problems suggest imitation as learning method. However, learning by imitation is not limited to movement imitation.

Imitation learning as machine learning principle is exactly that: a principle, more than a specific algorithm. The algorithms based on imitation learning differ greatly from problem to problem. Some common points are however the belief that there is an optimal behavior that can be learned, as well as an expert who can demonstrate the behavior. These demonstrations do not need to be perfect, as we will see later. Common to all approaches is the reduction of complexity of the learning task - instead of having to explore an intractable state/action space, significant information can be inferred from the demonstrations, and only a significantly smaller part of the state/action-space still needs to be explored.

[Argall et al., 2008] distinguish between imitation learning implementations - among other criteria - according to what is learned(See figure 1):

1. a mapping function $f : Z \to A$ from observed state to action
2. a system dynamics model $T(s'|s, a)$ and possibly a reward function $r(s, a)$
3. a set of pre- and post-conditions $pre(a), post(a)$ that can be used in planning algorithms

Note how this corresponds to the three criteria for imitation in section 1.4. A mapping function directly encodes behavior, while the goal-oriented approach of learning pre- and postconditions of actions, as well as the reward function from the system model approach, emphasize the goal side of imitation. The distinction shows the variety of techniques using imitation learning. It also shows that often, imitation learning is only the starting point for actually learning the desired behavior. Further learning steps derive the final policies from the learned information. Note that deriving a mapping function does not mean the mapping function cannot be further enhanced in subsequent learning steps. In section 4, we discuss an example where an initially learned policy is further enhanced through reinforcement learning. In this survey, we will focus on methods that more or less directly derive a mapping function from the demonstrations.

2.1 Basic concepts of imitation learning

Assuming state transitions are deterministic, imitating a certain movement can be seen as the task of learning a certain trajectory - of positional variables as well as motor control variables. This is a trivial task if the teacher can demonstrate a perfect trajectory that can be fully observed and repeated. This is most often not the case.

In the case described above, and even when state transitions are non-deterministic (meaning there is noise in the actuators, or there are variables that can not directly be influenced by the agent), the policy can be described as a set of differential equations

$$\dot{x} = f(x, y)$$

where x are the control variables (maybe including positional variables) and y are external variables, which may include time, positions of other objects, etc. Expressing this in the terms of state and

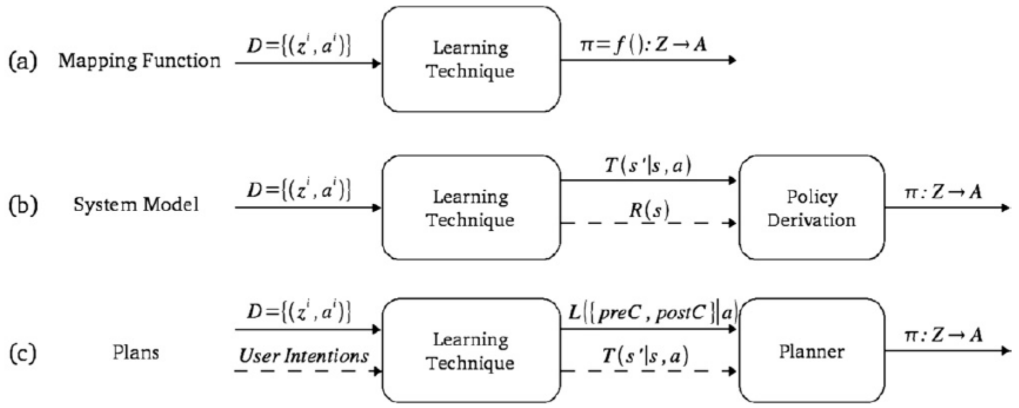

Figure 1: [Argall et al., 2008] Policy derivation using the generalization approach of determining (a) an approximation to the state-action mapping function, (b) a dynamics model of the system and (c) a plan of sequenced actions.

action, (x, y) is the state and $\dot{x}$ is the action performed by the agent. The differential equation is thus just another instantiation of a state-action mapping function.

Now, if the behavior is more complex and consists of several goal-oriented steps - like walking to a certain place, performing situational actions to achieve a certain goal, the problem suddenly becomes a lot more difficult due to the larger state space (which is just one-dimensional and finite if one only wants to learn a (time-dependent) trajectory). Thus, even with imitation learning, the problem of high dimensionality of the state/action-space still persists. A goal-oriented approach is thus still intractable if the movement is not split into (goal-oriented) primitives of higher-level abstraction (like "moving forward", "grasping" etc. or even more complex things)([Schaal, 1999]). Then, the learning task is again reduced to primitives that can either be trajectories or combinations of other primitives. However, there must be a model that tells how the primitives help achieving a goal. Once enough primitives are learned, common planning algorithms can be used to combine them to achieve the goals.

2.2 Difficult aspects of imitation learning

When applying imitation to learning robotic behavior, one has to find a mapping from the teacher's coordinate frame to the learner's coordinate frame. This means we need a model of how the teacher's behavior can be imitated. Another problem related to this is the problem that the teacher's variables might only be partially observable. For example, if trying to imitate a human with a robot, of course we can only measure and relate the movement itself, but are ignorant as to how the robot motors need to be controlled to achieve the same movement. Then a model is needed for inferring the missing variables. They can, for example, be learned in an additional learning step with common techniques like the EM algorithm or just be computed through inverse kinematics algorithms.

[Argall et al., 2008] divide the imitation problems among other criteria according to

1. The mapping between the teacher's actions / observations and the recorded training data (record mapping)

2. The mapping between the stored training data and the learner's actions / observations (embodiment mapping)

The distinction between record and embodiment mapping is irrelevant when training only one specific robot, but becomes interesting when the problem makes it necessary to train physically different robots from the same training data. An example of a mapping (record or embodiment depends on the specific problem) is that when measuring the joint angles of the teacher throughout the demonstration, we might have to map it to motor torques.

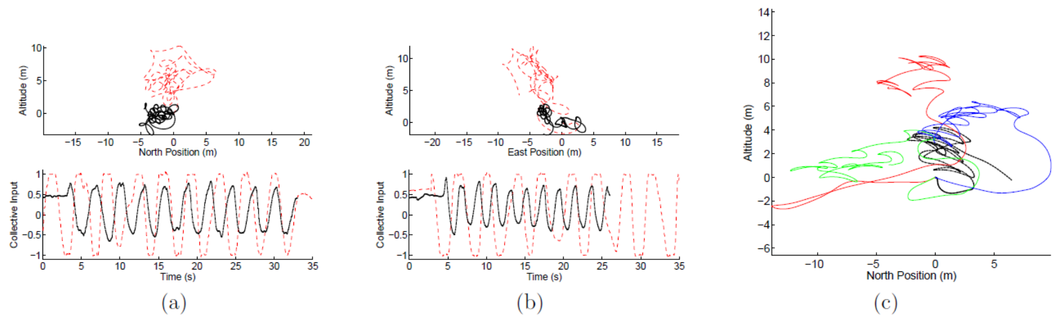

Figure 2: Flight trajectories: (a),(b), Solid black: [Coates et al., 2008], dashed red:[Abbeel et al., 2007]. (c) Dotted black: autonomous tic-toc. Solid colored: expert demonstrations

Another problem arises if state transitions are not deterministic and/or there are variables that can not be controlled or only indirectly controlled. Then it is intractable to show enough demonstrations to cover the possible behavior of the so-called external variables ([Kober et al., 2010]). Instead, imitation can be used to learn the basic behavior, and a model for the external variables and their relation to the internal variables of the agent then needs to be learned through other methods, for example reinforcement learning. Later, we will discuss the work of [Kober et al., 2010] as an example of this problem.

3 Case study I: Helicopter

In the following, we will discuss the work of [Coates et al., 2008], a case study using imitation learning - namely an autonomous helicopter learning aerobatic figures from a set of imperfect demonstrations.

For an autonomous helicopter, one is in the advantageous situation that the desired movements can be demonstrated using a remote-controlled model, maybe even the same model as the autonomous helicopter itself. This means it is possible to capture and measure all relevant variables, including the control variables. This makes learning easier as we do not need to infer the control variables from the positional variables.

However, helicopter aerobatics is a field where it is difficult, if not impossible, to obtain perfect demonstrations of a desired trajectory. Therefore, an approach is chosen to infer the optimal trajectory from a set of suboptimal demonstrations.

Multiple demonstrations are combined to obtain a better trajectory. Note however that because the demonstrations may not be aligned on the time axis, it is necessary to treat the time indices of the points on the trajectory as unobserved random variables, that have a certain probability to differ a little from the ones of the demonstrations. The observed variables are treated as noisy observations of the one "true" trajectory (the noise is assumed to be gaussian). Then the EM-algorithm is applied to learn the missing parameters, namely the time-shifts and the parameters of the trajectory distributions, which in fact means that the "true" trajectory is learned. For computing the time-shifts, dynamic time warping is applied alternatingly with the EM algorithm steps.

[Coates et al., 2008] successfully trained an autonomous helicopter to perform in-place rolls and flips, as well as "tic tocs" and a complete airshow. They showed that their approach significantly improved the helicopter's performance over previous works. In Figure 2(a) and (b), a comparison is shown between the earlier work of [Abbeel et al., 2007] and [Coates et al., 2008]. The figures compare the X-Z position of the helicopter in the top image, as well as the control inputs in the bottom image. Figure 2(a) shows the results for in-place rolls, while (b) shows the results for the in-place flips. It can be seen that the latter work significantly improves the performance of the learned trajectory, in the sense that (i) the overall positioning is more stable, and (ii) the control inputs are generally lower (which roughly corresponds to energy consumption during the maneuver).

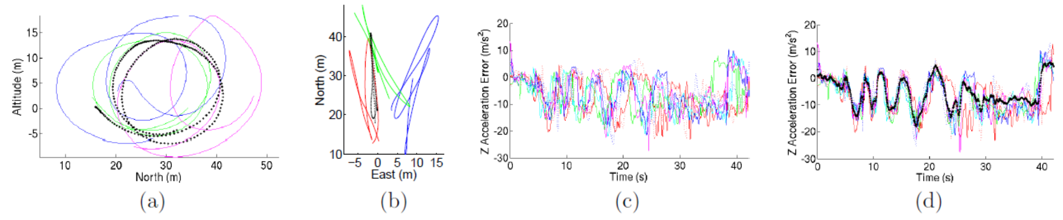

Figure 3: [Coates et al., 2008] Colored Lines: demonstrations. Black dotted line: inferred trajectory

One of the main questions was whether, given multiple suboptimal trajectories, a trajectory can be inferred with a better performance than the original demonstrations. Figures 3 (a) through (d) compare the original demonstration trajectories with the inferred final trajectory of the helicopter for two loops, figure 2 (c) shows the same for an autonomous tic-toc. The colored lines show the demonstrations, while the black dotted line shows the inferred ideal trajectory produced by the algorithm. Figure 3 (a) shows that the inferred trajectories are more rounded than the demonstrations. Figure 3 (b) shows a top-down view on the (three-dimensional) trajectory. The inferred loops lie in a plane like expected, while the demonstrations deviate from lying in a plane. Figure 2m (c) shows that the inferred tic-toc is also more stable than the demonstrations. Therefore one can say that using multiple demonstrations of a suboptimal teacher can raise the performance of the learner above that of the teacher.

4 Case study II: Ball-in-a-cup

The work of [Kober et al., 2010] is a second example of imitation learning, which shows different aspects of that method. [Kober et al., 2010] train a robot to play the Ball-in-a-cup game. This game consists of holding a cup, to which a ball is attached via a string, and then flinging the ball into the cup. This is already a complex task when the initial conditions (that is, ball position and velocity) are always the same. When the initial conditions vary largely, even skilled human players would be challenged.

[Kober et al., 2010] use an approach that they feel is similar to a child learning the game - the basic movement is learned, and then refined through trial-and-error (which means reinforcement learning in the realm of machine learning). The problem is that (i) actions can be noisy, and (ii) there are external variables that cannot be controlled directly (the ball position and velocity), but still affect the outcome of the action. These external variables relate to the inner variables (joint positions etc.), but the relation, also called perceptual coupling, needs to be learned. This could be done by imitation, but it would need intractably many demonstrations to capture a significant amount of possible initial conditions.

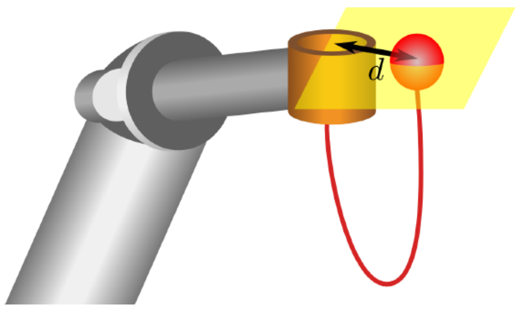

Figure 4: [Kober et al., 2010] Illustration of the reward calculation for the reinforcement learning part of the learning. When the ball passes the plane of the upper rim of the cup from above, the distance of the ball center and the rim center is penalized.

For this reason [Kober et al., 2010] split the learning into two parts: They first learn a model for unperturbed initial conditions and actions by imitation. Then this model is refined through reinforce-

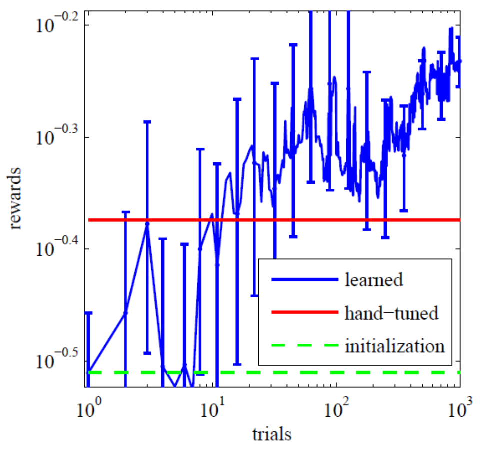

Figure 5: [Kober et al., 2010]: This figure shows the expected return for one specific perturbation of the learned policy in the Ball-in-a-Cup scenario (averaged over 3 runs with different random seeds and the standard deviation indicated by the error bars). Convergence is not uniform as the algorithm is optimizing the returns for a whole range of perturbations and not for this test case. Thus, the variance in the return as the improved policy might get worse for the test case but improve over all cases. The algorithm rapidly improves, regularly beating a hand-tuned solution after less than fifty trials and converging after approximately 600 trials. Note that this plot is a double logarithmic plot and, thus, single unit changes are significant as they correspond to orders of magnitude.

ment learning. Their reinforcement learning algorithm was specifically tailored to their problem and is based on the EM-algorithm. Their reward function was tied to the distance of the ball to the cup in the moment where the ball passed the cup plane from above (see Figure 4). For exploration, the actions where perturbed with gaussian noise based on the state of the system. In general, random perturbations of the actions can be dangerous for a physical robot. By taking into account the state of the system for the perturbation, regions of the state space that were dangerous for the robot could be avoided. In their experiments, the learned model regularly outperformed the hand-tuned solution after less than fifty trials, and converged after 600 trials.

It is interesting to note that the learning was performed in the cartesian space (meaning the position and velocity of the cup and ball), the motor controls were inferred through inverse kinematics. They also tried to learn directly in the joint-space of the robot (which had seven degrees of freedom), but were unable to achieve the same performance. This can be explained through the higher complexity of the learning task, as essentially the inverse kinematics needs to be implicitly learned. Also through perturbation of the initial conditions, the robot easily gets into regions where his kinematics model is inaccurate. By this, we see that the choice of state and action representation has a significant impact on learning performance, and should be treated with care. It is also interesting to see that a more complex embodiment mapping (here in the form of inverse kinematics) does not necessarily hinder the learning, but may even help it.

5 Comparison

As we have seen, imitation learning implementations differ as much as the problems differ. The two case studies use completely different methods: There is no time-warping for the Ball-in-a-cup problem, simply because there is only one demonstration. There is no reinforcement learning for the autonomous helicopter, because there is no additional knowledge that could be learned through trial and error, and everything is more or less deterministic and directly controllable.

What is common to both approaches is that the performance of the learner may rise above the performance of the teacher. In the helicopter case this is done by combining information from multiple demonstrations. In the ball-in-the-cup case, the initial learning experience is enhanced through additional experience gained from own execution of the learned behavior. In both cases, more information is learned than would have been available from a single demonstration itself.

6 Conclusion

Imitation learning is a tool for improving existing learning methods. We discussed how reinforcement learning suffers from the high dimensionality of state/action spaces in robotics. In such cases, reinforcement learning can become intractable because extensive exploration needs to be done before anything useful is learned. Imitation learning can help focus exploration (and thus learning) on parts of the state/action space that are actually relevant. We have seen that there has been significant success in applying this idea, and we showed two concrete example, namely an autonomous helicopter learning aerobatic maneuvers and a robotic arm learning the game ball-in-a-cup. In the autonomous helicopter case, we have seen that it is possible for a learner to show a higher performance than the teacher if the teacher provides multiple suboptimal demonstrations. In the robotic arm case, it has been shown that a learned policy can quickly and significantly outperform a handcrafted policy. Thus imitation learning not only is able to capture behavior demonstrated by an expert, but also to surpass the performance of the expert.

References

[Abbeel et al., 2007] Abbeel, P., Coates, A., Quigley, M., and Ng, A. Y. (2007). An application of reinforcement learning to aerobatic helicopter flight. *Education*, 19:1.

[Argall et al., 2008] Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2008). A survey of robot learning from demonstration.

[Coates et al., 2008] Coates, A., Abbeel, P., and Ng, A. Y. (2008). Learning for control from multiple demonstrations. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 144–151, New York, NY, USA. ACM.

[Kober et al., 2010] Kober, J., Mohler, B., and Peters, J. (2010). Imitation and reinforcement learning for motor primitives with perceptual coupling. In Sigaud, O. and Peters, J., editors, *From Motor Learning to Interaction Learning in Robots*, volume 264 of *Studies in Computational Intelligence*, pages 209–225. Springer Berlin / Heidelberg.

[Russell and Norvig, 2004] Russell, S. J. and Norvig, P. (2004). *Artificial Intelligence: A Modern Approach.* Pearson Education.

[Schaal, 1999] Schaal, S. (1999). Is imitation learning the route to humanoid robots?