Towards a Unified Framework for Learning from Observation

Santiago Ontañón
IIIA - CSIC
Bellaterra (Spain)
santi@iiia.csic.es

José L. Montaña
University of Cantabria
Santander (Spain)
montanjl@unican.es

Avelino J. Gonzalez
University of Central Florida
Orlando, FL (USA)
gonzalez@mail.ucf.edu

Abstract

This paper discusses the recent trends in machine learning towards learning from observation (LfO). These reflect a growing interest in having computers learn as humans do — by observing and thereafter imitating the performance of a task or an action. We discuss the basic foundation of this field and the early research in this area. We then proceed to characterize the types of tasks that can be learned from observation and how to evaluate an agent created in this manner. The main contribution of this paper is a joint framework that unifies all previous formalizations of LfO.

1 Introduction

Learning by watching others do something is a natural and highly effective way for humans to learn. It is also an intuitive and highly promising avenue for machine learning. It provides a way for machines to learn how to perform tasks in a more natural fashion. When we teach other humans, we often tell them "Here, watch how I do it." This is because for many tasks, this is better than providing static examples that explicitly contain the solution, as in the traditional supervised learning approach. Humans simply execute the task and trust that the observer can figure out how to successfully imitate the performance of the task.

Fernlund et al [Fernlund *et al.*, 2006] define learning from observation (LfO) as:

> The agent shall adopt the behavior of the observed entity solely from interpretation of data collected by means of observation.

However, this is a superficial characterization of LfO that merely dictates how the data are to be collected for learning. It makes no effort to formalize the concept of LfO, which algorithms to use, or what sets it apart from supervised and unsupervised learning.

We argue that LfO is distinguishable from conventional unsupervised learning because the examples (demonstrations) contain indication of correct behavior. LfO can be more readily likened to supervised learning, but here as well, distinguishing features exist. For one, the learning examples are time-based, continuous and inseparable through the duration of the exercise. Furthermore, no explicit linkage between cause and effect is provided, but must be extracted automatically by the learning algorithm as part of learning through observation. Our analysis shows, however, that some restricted types of learning from observation tasks can be addressed with supervised learning, although to solve the general problem LfO problem, supervised learning algorithms are not sufficient. Finally, compared to reinforcement learning, LfO learns from a collection of traces (or trajectories), rather than from a reinforcement signal.

Works reported in the literature also refer to *learning from demonstration*, *learning by imitation*, *programming by demonstration*, or *apprenticeship learning*, as largely synonymous to learning from observation. In learning from demonstration, a human purposely demonstrates how to perform a task or an action, expressly to teach a computer agent how to perform the same task or mission. We consider learning from demonstration to be a specialization of LfO and define the latter as a more general learning approach, where the actor being observed need not be a willing participant in the teaching process. This could include opponents being observed in team game competition or the enemy tactics being learned through observation (and thereafter modeled).

The remainder of this paper is organized as follows. Section 2 briefly summarizes previous research in the field. After that, Section 3 introduces a common framework and vocabulary for learning from observation. Section 4 then presents a classification of all the different tasks to be attempted using LfO. Finally, Section 5 presents a statistical formalization of the problem, which offers some interesting insights on the kind of algorithms required to solve different LfO tasks.

2 Learning from Observation Background

Work in learning from observation can be traced back to the early days of AI. For instance, Bauer [1979] proposed in 1979 to learn programs from example executions, which basically amounts to learning strategies to perform abstract computations by demonstration, and it was especially popular in robotics [Lozano-Pérez, 1983]. Another early mention of learning from observation comes from Michalski et al. [1983] who define it merely as unsupervised learning. Gonzalez et al [Gonzalez *et al.*, 1998] discussed learning from observation at length, but provided no formalization nor suggested an approach to realize it algorithmically.

More recent work on the more general LfO subject came nearly simultaneously but independently from Sammut et al [Sammut *et al.*, 1992] and Sidani [Sidani, 1994]. Fernlund et al. [2006] used learning from observation to build agents capable of driving a simulated automobile in a city environment. Pomerleau [Pomerleau, 1989] developed the ALVINN system that trained neural networks from observation of a road-following automobile in the real world. Moriarty and Gonzalez [Moriarty and Gonzalez, 2009] further the use of neural networks to carry out learning from observation for computer games.

Könik and Laird [2006] further introduced learning from observation in complex domains with the SOAR system by using inductive logic programming techniques.

Other significant work done under the label of learning from demonstration has emerged recently in the case-based reasoning community. Floyd et al. [2008] present an approach to learn how to play RoboSoccer by observing the play of other teams. Ontañón et al. [2010] use learning from demonstration in the context of real-time strategy games in the context of case-based planning. The main difference between the work based on CBR and the previous work presented in this section is that CBR methods are related to lazy machine learning techniques, which do not require any form of generalization during learning. In CBR, thus, learning becomes only memorization of new cases, and any kind of generalization is delayed until problem solving time.

Finally, a related area is that of *inverse reinforcement learning* [Ng and Russell, 2000], where the focus is on reconstructing the reward function given optimal behavior (i.e. given a policy, or a set of trajectories). One of the main problems here is that different reward functions may correspond to the observed behavior, and heuristics need to be devised to only consider families of reward functions that are interesting.

We can see that while a significant amount of work has been on-going over the last 20 years (see [Argall *et al.*, 2009] for a more in-depth recent overview), the problem does not enjoy a modicum of formalization, or even of agreement in terminology. We hope to provide these in this paper.

3 A Unified Framework for LfO

In this section we propose a formalization that attempts to be a unification of previous work in the area. Let us start by introducing the different elements appearing in LfO:

- There is a *task* T to be learned, which can be either achieving a condition (such as building a tower, or defeating an enemy), maintaining some process over time (such as keeping a car on the road), or maximizing some value (e.g. a reward function)
- There is an *environment* E.
- There is one *actor* (or trainer, expert, or demonstrator) C, who performs the task T in the environment (there can in principle be also be more than one actor).
- There is a *learning agent* A, whose goal is to learn how to achieve T in the environment E by observing C.

In Learning from observation, the learning agent A typically first observes one or several actors performing the task in the environment, and records their behavior in the form of *traces*. Then, those traces are used to learn how to perform T. The environment in which the learning agent observes the actor perform the task and the one in which he later tries to perform do not necessarily have to be exactly the same. For instance, in the context of a computer game, the learning agent might observe the actor play the game in a particular map, and then try to perform in a different map.

Inputs to the learning process are temporally based. In supervised learning we have problem/solution pairs, in unsupervised learning, only problems, and in reinforcement learning we have a reward signal. In LfO we have traces that contain the evolution of the environment over time while the actor is executing actions to achieve a task.

Let B_C be the behavior of an actor C. By *behavior*, we mean the control mechanism, policy, or algorithm that an actor or a learning agent use to determine which actions to execute over time. In a particular execution, a behavior B_C is manifested as the series of actions that the actor executes over time, which we call a *behavior trace*, or BT. Depending on the nature of the task to be learned, the actions of the actor can vary in nature. These could be atomic actions, durative actions, or just a collection of control variables that are adjusted over time. In general, we will assume that the actor can control a collection of variables, and thus, a behavior trace is defined as the variation of these variables over time (in case of atomic actions, there would be a single variable containing the action executed at each time step, if any):

$$BT_C = [(t_1, y_1), ..., (t_n, y_n)]$$

where, t_i are time points, and y_i are the control variables. We assume here that the learning agent can execute the same set of actions as the actor. Otherwise, in addition to a LfO problem, the agent would have to solve an action mapping problem. The evolution of the environment is captured into an input trace:

$$IT = [(t_1, x_1), ..., (t_n, x_n)]$$

The combination of actor behavior plus input trace, constitutes a learning trace in learning from observation:

$$LT = [(t_1, x_1, y_1), ..., (t_n, x_n, y_n)]$$

Notice that this formalization of a trace is general enough for both discrete and continuous time domains, since typically, LfO algorithms which operate in domains with continuous time sample the environment with a given frequency (e.g. [Bentivegna and Atkeson, 2001]), and thus effectively also have a trace that matches our previous definition.

The goal of learning from observation is thus, given:

- A collection of learning traces $LT_1, ..., LT_k$
- An environment E (characterized by a set of input variables x and a set of control variables y).
- Optionally, a target task T

Learn a behavior B that behaves in the same way as the actors do in the environment E, given the same inputs (while achieving the task T, if specified). Notice that in the case where no target task T is specified, the LfO problem is equivalent to learning to predict the teacher's actions.

3.1 Measuring Performance in LfO

Measuring the success of a LfO system is not trivial [Argall *et al.*, 2009]. In supervised learning, a simple way is to leave some training examples out, and use them as test. The equivalent in LfO would be to leave some learning traces out, and use them to verify the learnt behavior B. However, in LfO we have to compare behavior traces, which is not trivial.

If a task T is specified, the accomplishment of such task should also be taken into account in the performance measure, and not just resemblance with the actions executed by the actor. There are thus two main additional issues with respect to evaluating performance in a supervised learning setting: a) there are two different variables to measure: task performance and resemblance with actor, and b) comparing traces is not trivial.

Because of the previous reasons, there are three basic strategies to evaluate agents trained through LfO, all of which have been extensively used in the literature:

- **Evaluate Performance:** measure how well the learning agent A performs T, regardless of whether it is performing similar to the actor C or not. For example, in a car driving scenario, we could measure whether the learning agent can keep the car in the road. For example, the work of Ontañón et al. [2010] uses this strategy.
- **Evaluate Output:** compare the actions executed by A when performing task T against the actions executed by the actor for the same task T. This evaluation method is the most similar to supervised learning. The set of learning traces can be divided into a training set and a test set, and use the test set of traces to evaluate the learning agent. For example, Floyd et al. [2008] used this strategy to evaluate their techniques.
- **Evaluate Model:** compare the actual model (program, state machine, set of rules, probability distribution, policy, or any other decision procedure) that the agent has learned against the model the actor was using to generate the traces. This evaluation method is specially interesting to assess whether a given procedure can be recovered by learning from a set of learning traces. Early work on programming by demonstration [Bauer, 1979] used this paradigm to evaluate success.

Depending on the specific domain, one (or a combination) of the approaches above might be the best choice. This section has focused on formalizing learning from observation by specifying which are the inputs and expected outputs of LfO systems. In the next section we focus on creating a classification of the different tasks that could be attempted by LfO.

4 Levels of Learning from Observation

It is intuitively evident that not all learning algorithms will work for all problems in LfO. Therefore, we next attempt to categorize the types of problems that could be addressed through LfO. First we must state that the general type of problems LfO addresses are those that involve some type of control function. Such control functions may be low level, such as motor skills or higher level. Let us first identify the key factors that play a role in the difficulty of a LfO task. In addition to the common factors such as variables (like time) being continuous or discrete, there are three key factors that determine the complexity of an LfO task:

- *Does it require generalization?* Generalization is the process by which a general statement is obtained by inference from specific instances. Some tasks, as we will show below, do not require generalization.
- *Does it require planning?* In some tasks, the learning agent has to take into account events that occurred earlier in time, or even consider events that might happen in the future in order to take a decision. In such tasks, we say that planning is required.
- *Is the environment known?* the learner might or might not have a model of the environment, like the set of important variables to perceive in the environment, or the effects of the actions the learner can execute. Not having a model means that the learner, in addition to learning the task, might have to learn a model at the same time.

We define four levels of difficulty of tasks and/or actions that can be learned from observation, depicted in Table 1:

Level 1 - Strict imitation: strict imitation tasks do not require feedback from the environment, and thus neither generalization nor planning. The learned tasks are a strict function of time. For that reason, it does not matter whether the environment is known or unknown. The learning algorithms to solve these tasks only require pure memorization. One can think of these tasks as requiring open-loop control. Robots in factories are an instance of this type of learning, where the pieces are always in the exact same place at all times.

Level 2 - Reactive skills: reactive skills correspond to input to action mapping, without requiring planning. Generalization is required to learn this task. Since the task is just to learn a mapping from the perceived state of the environment and action, standard supervised machine learning techniques could be used to address this level. Learning how to play some of the old video games such as pong and space invaders would fit this category.

Level 3 - Tactical behavior in known environments: learning tactical behaviors requires generalization, and planning. This levels includes tasks in which the current state of the world is not enough to determine which actions to execute, previous states or potential future events might have to be taken into account. The control function to learn is not just a situation to action mapping, but might have internal state. For example, learning how to play a game like Stratego is a task of Level 3. The player has to remember and analyze past events in order to infer the piece types of the opponent.

Level 4 - Tactical behavior in unknown environments: Tasks that fall in this category do require generalization, and potentially also planning. Additionally, the agent does not understand the environment. Thus, the learning algorithms might need to address both the problem of learning the task and learning the environment. For

Generalization?	Planning?	Known environment?	Level
no	no	-	Level 1: Strict Imitation
yes	no	yes	Level 2: Reactive skills
yes	yes	yes	Level 3: Tactical in known environment
yes	-	no	Level 4: Tactical in unknown environment

Table 1: Characterization of the different levels of difficulty of learning from observation.

example, any of the previously mentioned tasks for levels 2 and 3 would be Level 4 if the environment is unknown.

Let us now formalize the intuitive descriptions of each of the 4 levels by using a statistical formulation of LfO.

5 Towards a Statistical Formulation of Learning from Observation

Imagine some actor C exhibiting a certain behavior B_C. The learning trace $LT = [(t_1, x_1, y_1), ..., (t_n, x_n, y_n)]$ observed by the learning agent A can be seen as the realization – also trajectory or sample path – of a stochastic process ([Papoulis and Pillai, 2002]) with state space $I = X \times Y$, where X denotes the representation of the environment and Y denotes the representation of the space of actions. Each component of the learning trace LT satisfies, $x_k \in X$ and $y_k \in Y$.

Accordingly, the behavior B_C can be interpreted as the stochastic process $I = \{I_k : k \in T\}$, where $T = \{1, ..., n\}$ and $I_k = (X_k, Y_k)$ is the random variable whose components have the following obvious meaning: X_k and Y_k represent respectively the state of the environment E at time t_k and the action performed by actor C at time t_k. In the following we assume that the random variables $I_k = (X_k, Y_k)$ are multidimensional variables that can be either continuous or discrete, that is, $X = \mathbb{R}^p$, for some p (or X is some discrete set) and $Y = \mathbb{R}^q$ for some q (or Y is a discrete set). Examples in which the environment is a continuous space and the actions are described inside a discrete space are found quite often in the literature (see for instance Pomerleau [1989]).

We also assume, as usual in stochastic process theory, that variables I_k are defined over the same probability space (Ω, F, ρ), where Ω is a nonempty set, F is the σ-algebra of events and ρ is an unknown probability measure that governs the behavior of the observed agent C. To simplify notation, the joint probability distributions $\rho_{1,...,k}$ of the joint random variables $(I_1, ..., I_k)$ will also be denoted by ρ if not confusion may arise.

Under this formalization, the LfO problem is to estimate some features of the unknown probability measure ρ, taking as input data a set of trajectories $\{LT_j : 1 \leq k \leq k\}$ of the stochastic process $I = \{I_k : k \in T\}$. The features to be learned regarding the probability distribution ρ depend on the level of the problem under consideration.

In a quite general situation, the learning agent A has to decide next action Y_{k+1} assuming the current state $X_{k+1} = x_{k+1}$ is known and the subtrace generated until the present moment, $I_k = (x_k, y_k), ..., I_1 = (x_1, y_1)$. This LfO scenario can be formulated as the problem of estimating the conditional probability distribution of Y_{k+1} given X_{k+1} and variables $I_k = (x_k, y_k), ..., I_1 = (x_1, y_1)$:

$$\rho(Y_{k+1}|X_{k+1} = x_{k+1}, \{I_j = (x_j, y_j))_{j=1}^{k}\}) \qquad (1)$$

In order to simplify, we shall denote the conditional probability in Equation 1 by

$$\rho(Y_{k+1}|x_{k+1}, i_k, i_{k-1}, \ldots, i_1)$$

However the scenario defined in Equation 1 may not cover all cases considered in a particular LfO problem. Imagine for instance that we have to imitate a behavior that has a task as goal. The task may specify some given future states and actions at some particular instants. In such situations it may be necessary to take into account knowledge regarding future events, that is, at times $j > k$. This suggests that in this scenario the LfO problem consists in estimating some conditional probability measure of the form:

$$\rho(Y_{k+1}|x_{k+1}, i_{k+j1}, i_{k+j2}, ..., i_{k+jk}, i_k \ldots, i_1) \qquad (2)$$

with $j1 > j2 > ... > jk \geq 1$.

Next we analyze the simplifications that occur in Equation 1 at each level of the LfO hierarchy proposed in Section 4. The same simplifications can be translated into the scenario described by Equation 2 if necessary.

Level 1. Strict imitation of a given behavior B_C from a trace is a deterministic process. If what we want is to reproduce the set of actions defined by a behavior trace $BT = [(t_1, y_1), ..., (t_n, y_n)]$, we just consider the set $\Omega = (X \times Y)^n$, where Y is the space of actions, the σ-algebra F given by all subsets of Ω, and the joint probability measure ρ defined as follows: $\rho(I_1 = (X_1, Y_1), ..., I_n = (X_n, Y_n))(A) = 1$, if $(y_1, \ldots, y_n) \in A$ and 0 otherwise, for all subsets $A \subset \Omega$. It is easy to see that in this level, the conditional variable $\zeta_k = Y_{k+1}|(x_{k+1}, i_k, i_{k-1}, \ldots, i_1)$, takes value y_{k+1} with probability 1. Hence, the stochastic process defined above $I = \{Y_k : k \in T\}$, $T = \{1, ..., n\}$, represents the LfO problem at level 1.

Level 2. At this level, generalization is required, and the task is just to learn a mapping from the perceived state of the environment into the state of actions. This means essentially that in this kind of problems the action only depends on the current state of the environment and no other dependencies must be considered. This implies, in particular, that the probability measure in Equation 1 satisfies the following property:

$$\rho(Y_{k+1}|x_{k+1}, i_k, ..., i_1) = \rho(Y_{k+1}|x_{k+1})$$

In this case a learning trace LT can be seen as a sample of a supervised learning problem. Our LfO problem consists in this case in predicting the value of random variable $Y = Y_{k+1}$ known the value $X = x_{k+1}$. In this context we can factorize

the probability measure ρ as usual: ρ_X is the induced probability measure on X (marginal distribution) and the distribution $\rho(Y|X = x)$ is the conditional probability measure. The LfO problem then consists in estimating the conditional probability measure.

Level 3. At this level the state and/or the action at time $k+1$ depend on the environment states and/or actions at previous instants and no other dependencies must be considered. The amount of previous knowledge required for predicting variable Y_{k+1} reflects the memory requirements to learn the agent's behavior. If, for instance, we only need to know the previous action and state, the conditional distribution considered in Equation 1 follows a rule which is similar to that of a Semi-Markov process, that is:

$$\rho(Y_{k+1}|x_{k+1}, i_k, ..., i_1) = \rho(Y_{k+1}|x_{k+1}, i_k)$$

Note however that in many real world situations at level 3 we need knowledge of more situations than the previous one. For instance, if what we want is to imitate the behavior of a person that browses the net, it is possible that, apart from knowledge of the current page, we need to take into account at least the two or three pages previously visited. This yields to a simplification of the form:

$$\rho(Y_{k+1}|x_{k+1}, i_k, ..., i_1) = \rho(Y_{k+1}|x_{k+1}, i_n, ..., i_{k-l}) \quad (3)$$

Here l plays a similar role to the order of a Markov process and defines a sub-level of level 3 that captures the amount of previous knowledge required to predict the current action. We assume that only a finite amount of previous knowledge is required to perform any task, and thus, this can be captured with the finite number l.

Level 4 As in level 3, at this level the state and/or the action at time $k+1$ depend on the environment states and/or actions at previous times. However the learner might not have a complete description of the environment. For modeling this situation we can assume that variables X_k take values in some infinite dimensional space in which the known part is embedded. For this purpose, a quite general stochastic scenario is that provided by Hilbert-valued stochastic processes.

5.1 An example: Autonomous Land Vehicle

To clarify the model presented above we consider the example of statistical modeling of the task performed by an Autonomous Land Vehicle (ALV) learning to drive by observing a human doing it. We represent the environment variable X by a matrix $X = (X_{kl})$ of real valued random variables representing the input retina image obtained by the driver. Each variable X_{kl} is a real random variable with values in the interval $[0, 1]$ representing a pixel greyscale.

The space of actions is represented by two random discrete variables $Y = (S, A)$, where A is the correct steering direction and A represents the correct accelerator position of the vehicle. Let us consider that S can take a finite set of values $S = \{s_1, ..., s_p\}$ where s_1 is the left most position and s_p is the corresponding right most position. Analogously A takes a finite set of values $A = \{a_1, ..., a_q\}$ where a_1 represents throttle up and a_q represents full throttle.

Level 1: ALV Problem If the road is always the same, there are no other vehicles and traffic signals, persons or agents interacting with the environment and the goal is merely to drive from a starting point to end point then the task is Level 1. This is because if the environment does not change then the agent will drive just by exactly imitating the actions of the human driver. This means in the statistical model that the behavior trace is a temporal series of actions representing the correct positions of the steering and the accelerator of the form $BT = [(t_1, (s_1^b, a_1^b)), ..., (t_n, (s_n^b, a_n^b))]$, with $(s_i^b, a_i^b) \in S \times A$, $\Omega = (X \times Y)^n$ is the space of events, where $Y = S \times A$ is the space of actions, the σ-algebra F given by all subsets of Ω, and the joint probability distribution ρ defined as follows: $\rho(I_1 = (X_1, X_1), ..., I_n = (X_n, Y_n))(B) = 1$, if $(s_i^b, a_i^b)_{1 \leq i \leq n} \in B$ and 0 otherwise, for any $B \subset \Omega$. It is straightforward to realize that he conditional variable $\zeta_k = Y_{k+1}|(x_{k+1}, i_k, i_{k-1}, \ldots, i_1)$, takes value (s_{k+1}^b, a_{k+1}^b) with probability 1, for $k = 0$ to $k = n-1$.

Level 2: ALV Problem If there are other agents on the road, like other vehicles driving with a given speed limit, and our autonomous agent is supposed to learn to slow down when crossed with other vehicles, then the task becomes Level 2, since generalization is required (the agent must learn from the particular instances when the human slowed down or stopped in the presence of other vehicles) and we must take into account the vehicle position, that is, the input retina image. In this case the learning trace is a sequence of pairs, $LT = (x_i, y_i)_{1 \leq i \leq n}$, where each x_i represents an input retina image (i.e. a matrix of pixels) and each y_i is a pair (s_i, a_i), where $s_i \in S$ and $a_i \in A$ are respectively the correct steering and accelerator positions obtained from a demonstration performed for instance by one or several human drivers.

We assume in this case, as in classical supervised learning, that the learning trace is a sample of random identically distributed variables (X_i, Y_i), generated according to an unknown probability measure ρ defined on the space $X \times Y$. This probability measure factorizes, as usual, in the marginal distribution ρ_X defined on the instance space X (environment in LfO) and the conditional probability measure $\rho(Y|X = x)$ defined on Y. The LfO problem is to estimate the probability that governs the random variable $Y|X = x$ whose meaning is "action y corresponding to input retina image x". We note that this approach is basically the initial formulation given in the ALVINN project by Pomerleau in [Pomerleau, 1989].

Level 3: ALV Problem If the task is to learn how to drive a car in several specified kinds of roads that may include other vehicles, traffic signals and maybe other agents like persons and animals crossing along the road, then the ALV task becomes Level 3, since the autonomous agent requires to take actions that depend on previous actions and situations. Note, for instance, that if the human driver sees a traffic signal like a stop signal at instant i, he will start to slow down and will stop at time $k = i + j$, $j > 0$. This means in particular, that action Y_k at time k depends of input retina image and actions at some previous instant $i < k$. This explains why we are interested in estimating the probability measure that governs variable $Y_{k+1}|x_{k+1}, i_k, ..., i_{k-l}$ as expressed in Equation 3 whose meaning is "action y corresponding to input retina image $x = x_{k+1}$ assuming that we know situations and actions

at the $l+1$ previous instants". Here l represents the number of previous images and actions that must be considered in order to predict action at present instant.

Note that this formulation seems to be more realistic than that provided by level 2. Note also that the generalization error of the steering and accelerator depends of the particular learning algorithm used to solve the problem, but for a given kind of algorithm –for instance, a neural network with an specified architecture. i. e., a fixed estimation error– the generalization error depends of the empirical error. We conjecture that a more realistic formulation of the problem, taking into account previous actions and situations and their dependencies can help to produce more robust algorithms to correct the steering and accelerator errors.

Level 4: ALV Problem Finally, the task could even become Level 4 if the agent has no information of which input variables in its sensors determine that there is a certain type of obstacle, or some of those variables are not even observable from the demonstrations provided by the human (for instance driving under rainy weather). In that case, the agent has to learn first, for example, which variables are relevant to learn the task and add these variables to the environment.

6 Conclusions

Despite the considerable amount of interest and work on learning from observation, the field lacks a unified framework for comparing different approaches. The main goal of this paper is to put forward a proposal to fill in that gap, and present a unified framework for learning from observation, consisting of three main components: a) a unified formalism, b) a classification of the different levels of difficulty of the tasks attempted by LfO, and c) a statistical model which sheds some light on the formal differences between LfO and other forms of learning. Finally, we have illustrated with an example how different LfO tasks fit in our formalism.

In our formalism, we have not discussed the different existing algorithms for LfO. We have instead focused on characterizing the task itself. As part of our future work we plan to characterize existing algorithms for LfO, and how do they fit under each of the levels presented in this paper.

References

[Argall *et al.*, 2009] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robot. Auton. Syst.*, 57:469–483, May 2009.

[Bauer, 1979] Michael A. Bauer. Programming by examples. *Artificial Intelligence*, 12(1):1–21, 1979.

[Bentivegna and Atkeson, 2001] Darrin C. Bentivegna and Christopher G. Atkeson. Learning from observation using primitives. In *In IEEE International Conference on Robotics and Automation*, pages 1988–1993, 2001.

[Fernlund *et al.*, 2006] H. K. G. Fernlund, Avelino J. Gonzalez, Michael Georgiopoulos, and Ronald F. DeMara. Learning tactical human behavior through observation of human performance. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 36(1):128–140, 2006.

[Floyd *et al.*, 2008] Michael W. Floyd, Babak Esfandiari, and Kevin Lam. A case-based reasoning approach to imitating robocup players. In *Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society (FLAIRS)*, pages 251–256, 2008.

[Gonzalez *et al.*, 1998] Avelino J. Gonzalez, Michael Georgiopoulos, Ronald F. DeMara, Amy Henninger, and William Gerber. Automating the cgf model development and refinement process by observing expert behavior in a simulation. In *Proceedings of The 7th Conference on Computer Generated Forces and Behavioral Representation*, 1998.

[Könik and Laird, 2006] Tolga Könik and John E. Laird. Learning goal hierarchies from structured observations and expert annotations. *Mach. Learn.*, 64(1-3):263–287, 2006.

[Lozano-Pérez, 1983] Tomás Lozano-Pérez. Robot programming. In *Proceedings of IEEE*, volume 71, pages 821–841, 1983.

[Michalski and Stepp, 1983] Ryszard S. Michalski and Robert E. Stepp. Learning from observation: Conceptual clustering. In Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, chapter 11, pages 331–364. Tioga, 1983.

[Moriarty and Gonzalez, 2009] Christopher Lawrence Moriarty and Avelino J. Gonzalez. Learning human behavior from observation for gaming applications. In *FLAIRS Conference*, 2009.

[Ng and Russell, 2000] Andrew Y. Ng and Stuart Russell. Algorithms for Inverse Reinforcement Learning. In *in Proc. 17th International Conf. on Machine Learning*, pages 663–670, 2000.

[Ontañón *et al.*, 2010] Santiago Ontañón, Kinshuk Mishra, Neha Sugandh, and Ashwin Ram. On-line case-based planning. *Computational Intelligence Journal*, 26(1):84–119, 2010.

[Papoulis and Pillai, 2002] Athanasios Papoulis and S. Unnikrishna Pillai. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill Series in Electrical and Computer Engineering. McGraw-Hill, 2002.

[Pomerleau, 1989] Dean Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems 1*. Morgan Kaufmann, 1989.

[Sammut *et al.*, 1992] Claude Sammut, Scott Hurst, Dana Kedzier, and Donald Michie. Learning to fly. In *ML*, pages 385–393, 1992.

[Sidani, 1994] T.A. Sidani. *Automated Machine Learning from Observation of Simulation*. PhD thesis, University of Central Florida, 1994.