Predictive Learning from Demonstration

Erik A. Billing, Thomas Hellström, and Lars-Erik Janlert

Department of Computing Science, Umeå University, 901 87 Umeå, Sweden
billing@cs.umu.se, thomash@cs.umu.se, lej@cs.umu.se

Abstract. A model-free learning algorithm called Predictive Sequence Learning (PSL) is presented and evaluated in a robot Learning from Demonstration (LFD) setting. PSL is inspired by several functional models of the brain. It constructs sequences of predictable sensory-motor patterns, without relying on predefined higher-level concepts. The algorithm is demonstrated on a Khepera II robot in four different tasks. During training, PSL generates a hypothesis library from demonstrated data. The library is then used to control the robot by continually predicting the next action, based on the sequence of passed sensor and motor events. In this way, the robot reproduces the demonstrated behavior. PSL is able to successfully learn and repeat three elementary tasks, but is unable to repeat a fourth, composed behavior. The results indicate that PSL is suitable for learning problems up to a certain complexity, while higher level coordination is required for learning more complex behaviors.

1 Introduction

Recent years have witnessed an increased interest in computational mechanisms that will allow robots to *Learn from Demonstrations (LFD)*. With this approach, also referred to as *Imitation Learning*, the robot learns a behavior from a set of good examples, *demonstrations*. The field has identified a number of key problems, commonly formulated as *what to imitate*, *how to imitate*, *when to imitate* and *who to imitate* [3]. In the present work, we focus on the first question, referring to which aspects of the demonstration should be learned and repeated.

Inspiration is taken from several functional models of the brain and prediction is exploited as a way to learn state definitions. A novel learning algorithm, called *Predictive Sequence Learning (PSL)*, is here presented and evaluated. PSL is inspired by *S-Learning* [42,43], which has previously been applied to robot learning problems as a model-free reinforcement learning algorithm [40,41].

The paper is organized as follows. In Sect. 2 a theoretical background and biological motivation is given. Section 3 gives a detailed description of the proposed algorithm. Section 4 describes the experimental setup and results for evaluation of the algorithm. In Sect. 5, conclusions, limitations and future work are discussed.

2 Motivation

One common approach to identify what in a demonstration that is to be imitated is to exploit the variability in several demonstrations of the same behavior. Invariants among the demonstrations are seen as the most relevant and selected as essential components of the task [3,17]. Several methods for discovering invariants in demonstrations can be found in the LFD literature. One method presented by Billard et al. applies a time-delayed neural network for extraction of relevant features from a manipulation task [4,5]. A more recent approach uses demonstrations to impose constraints in a dynamical system, e.g. [16,25].

While this is a suitable method for many types of tasks, there are also applications where it is less obvious which aspects of a behavior should be invariant, or if the relevant aspects of that behavior is captured by the invariants. Since there is no universal method to determine whether two demonstrations should be seen as manifestations of the same behavior or two different behaviors [10], it is in most LFD applications up to the teacher to decide. However, the teacher's grouping of actions into behaviors may not be useful for the robot. In the well known imitation framework by Nehaniv and Dautenhahn [34], it is emphasized that the success of an imitation is observer dependent. The consequence of observer dependence when it comes to interpreting sequences of actions has been further illustrated with Pfeifer and Scheier's argument about the *frame of reference* [35,36], and is also reflected in Simon's parable with the ant [45]. A longer discussion related to these issues can be found in [6].

Pfeifer and Scheier promotes the use of a *low level specification* [36], and specifically the *sensory-motor space* $I = U \times Y$, where U and Y denotes the *action space* and *observation space*, respectively. Representations created directly in I prevents the robot from having memory, which has obvious limitations. However, systems with no or very limited memory capabilities has still reached great success within the robotics community through the works by Rodney Brooks, e.g., [13,14,15,12], and the development of the *reactive* and *behavior based* control paradigms, e.g., [1]. By extending the definition of I such that it captures a certain amount of temporal structure, the memory limitation can be removed. Such a temporally extended sensory-motor space is denoted *history information space* $I^\tau = I_0 \times I_1 \times I_2 \times \ldots \times I_\tau$, where τ denotes the temporal extension of I *[10]*. With a large enough τ, I^τ can model any behavior. However, a large τ leads to an explosion of the number of possible states, and the robot has to generalize such that it can act even though the present state has not appeared during training.

In the present work, we present a learning method that is not based on finding invariants among several demonstrations of, what the teacher understands to be "the same behavior". Taking inspiration from recent models of the brain where prediction plays a central role, e.g. [22,23,27,32], we approach the question of what to imitate by the use of prediction.

2.1 Functional Models of Cortex

During the last two decades a growing body of research has proposed computational models that aim to capture different aspects of human brain function, specifically the cortex. This research includes models of perception, e.g., Riesenhuber and Poggio's hierarchical model [38] which has inspired several more recent perceptual models [23,32,37], models of motor control [26,42,48,47,46] and learning [22]. In 2004, this field reached a larger audience with the release of Jeff Hawkins's book On Intelligence [28]. With the ambition to present a unified theory of the brain, the book describes cortex as a hierarchical memory system and promotes the idea of a *common cortical algorithm*. Hawkins's theory of cortical function, referred to as the *Memory-Prediction framework*, describes the brain as a prediction system. Intelligence is, in this view, more about applying memories in order to predict the future, than it is about computing a response to a stimulus.

A core issue related to the idea of a common cortical algorithm is what sort of bias the brain uses. One answer is that the body has a large number of reward systems. These systems are activated when we eat, laugh or make love, activities that through evolution have proved to be important for survival. However, these reward systems are not enough. The brain also needs to store the knowledge of how to activate these reward systems.

In this context, prediction appears to be critical for learning. The ability to predict the future allows the agent to foresee the consequences of its actions and in the long term how to reach a certain goal. However, prediction also plays an even more fundamental role by providing information about how well a certain model of the world correlates with reality.

This argument is supported not only by Hawkins's work, but by a large body of research investigating the computational aspects of the brain [8]. It has been proposed that the central nervous system (CNS) simulates aspects of the sensorimotor loop [29,31,33,47]. This involves a modular view of the CNS, where each module implements one *forward model* and one *inverse model*. The forward model predicts the sensory consequences of a motor command, while the inverse model calculates the motor command that, in the current state, leads to the goal [46]. Each module works under a certain *context* or bias, i.e., assumptions about the world which are necessary for the module's actions to be successful. One purpose of the forward model is to create an estimate of how well the present situation corresponds to these assumptions. If the prediction error is low the situation is familiar. However, if the prediction error is high, the situation does not correspond to the module's context and actions produced by the inverse model may be inappropriate.

These findings have inspired recent research on robot perception and control. One example is the *rehearse, predict, observe, reinforce* decomposition proposed by [18,20,44] which adapts the view of perception and action as two aspects of a single process. Hierarchical representations following this decomposition have also been tested in an LFD setting [19] where the robot successfully learns sequences of actions from observation. In work parallel to this, we also investigates

PSL as an algorithm for behavior recognition [11], exploring the possibilities to use PSL both as a forward and an inverse model. The present work should be seen as a further investigation of these theories applied to robots, with focus to learning with minimal bias.

2.2 Sequence Learning

PSL is inspired by *S-Learning*, a dynamic temporal difference (TD) algorithm presented by Rohrer and Hulet, [42,43]. S-Learning builds sequences of passed events which may be used to predict future events, and in contrast to most other TD algorithms it can base its predictions on many previous states.

S-Learning can be seen as a variable order Markov model (VMM) and we have observed that it is very similar to the well known compression algorithm LZ78 [49]. This coincidence is not that surprising considering the close relationship between loss-less compression and prediction [2]. In principle, any lossless compression algorithm could be used for prediction, and vice verse [21].

S-Learning was originally developed to capture the discrete episodic properties observed in many types of human motor behavior [39]. Inspiration is taken from the *Hierarchical Temporal Memory* algorithm [24], with focus on introducing as few assumptions into learning as possible. More recently, it has been applied as a model-free reinforcement learning algorithm for both simulated and physical robots [40,41]. We have also evaluated S-Learning as an algorithm for behavior recognition [9]. However, to our knowledge it has never been used as a control algorithm for LFD.

The model-free design of S-Learning, together with its focus on sequential data and its connections to human motor control makes S-Learning very interesting for further investigation as a method for robot learning. With the ambition to increase the focus on prediction, and propose a model that automatically can detect when it is consistent with the world, PSL was designed.

3 Predictive Sequence Learning

PSL is trained on an *event sequence* $\eta = (e_1, e_2, \ldots, e_t)$, where each *event* e is a member of an alphabet $\sum$. η is defined up to the current time t from where the next event e_{t+1} is to be predicted.

PSL stores its knowledge as a set of hypotheses, known as a *hypothesis library* H. A *hypothesis* $h \in H$ expresses a dependence between an event sequence $X = (e_{t-n}, e_{t-n+1}, \ldots, e_t)$ and a target event $I = e_{t+1}$:

$$h : X \Rightarrow I \tag{1}$$

X_h is referred to as the *body* of h and I_h denotes the *head*. Each h is associated with a *confidence* c reflecting the conditional probability $P(I|X)$. For a given η, c is defined as $c(X \Rightarrow I) = s(X, I)/s(X)$, where the *support* $s(X)$ describes the proportion of transactions in η that contains X and (X, I) denotes

Algorithm 1 Predictive Sequence Learning (PSL)

Require: an event sequence $\eta = (e_1, e_2, \ldots, e_n)$

1: $t \leftarrow 1$
2: $H \leftarrow \emptyset$
3: $M \leftarrow \left\{h \in H \mid X_h = \left(e_{t-|h|+1}, e_{t-|h|+2}, \ldots, e_t\right)\right\}$
4: **if** $M = \emptyset$ **then**
5: **let** $h_{new} : (e_t) \Rightarrow e_{t+1}$
6: **add** h_{new} to H
7: **goto** 20
8: **end if**
9: $\hat{M} \leftarrow \{h \in M \mid |h| \geq |h'| \ for\ all\ h' \in M\}$
10: **let** $h_{max} \in \left\{h \in \hat{M} \mid c(h) \geq c(h') \ for\ all\ h' \in \hat{M}\right\}$
11: **if** $e_{t+1} \neq I_{h_{max}}$ **then**
12: **let** h_c be the longest hypothesis $\{h \in M \mid I_h = e_{t+1}\}$
13: **if** $h_c = null$ **then**
14: **let** $h_{new} : (e_t) \Rightarrow e_{t+1}$
15: **else**
16: **let** $h_{new} : \left(e_{t-|h_c|}, e_{t-|h_c|+1}, \ldots, e_t\right) \Rightarrow e_{t+1}$
17: **end if**
18: **add** h_{new} to H
19: **end if**
20: update the confidence for h_{max} and $h_{correct}$ as described in Sect. 3
21: $t \leftarrow t + 1$
22: **if** $t < n$ **then**
23: **goto** 2
24: **end if**

the concatenation of X, and I. A transaction is defined as a sub-sequence of the same size as X. The length of h, denoted $|h|$, is defined as the number of elements in X_h. Hypotheses are also referred to as *states*, since a hypothesis of length $|h|$ corresponds to VMM state of order $|h|$.

3.1 Detailed Description of PSL

Let the library H be an empty set of hypotheses. During learning, described in Alg. 1, PSL tries to predict the future event e_{t+1}, based on the observed event sequence η. If it fails to predict the future state, a new hypothesis h_{new} is created and added to H. h_{new} is one element longer than the longest matching hypothesis previously existing in H. In this way, PSL learns only when it fails to predict.

For example, consider the event sequence $\eta = ABCCABCCA$. Let $t = 1$. PSL will search for a hypothesis with a body matching A. Initially H is empty and consequently PSL will create a new hypothesis $(A) \Rightarrow B$ which is added to H. The same procedure will be executed at $t = 2$ and $t = 3$ so that $H = \{(A) \Rightarrow B; (B) \Rightarrow C; (C) \Rightarrow C\}$. At $t = 4$, PSL will find a matching hypothesis $h_{max} : (C) \Rightarrow C$ producing the wrong prediction C. Consequently, a

Algorithm 2 Making predictions using PSL

Require: an event sequence $\eta = (e_1, e_2, \ldots, e_{t-1})$
Require: the trained library $H = \left(h_1, h_2, \ldots, h_{|H|}\right)$

1: $M \leftarrow \left\{h \in H \mid X_h = \left(e_{t-|h|}, e_{t-|h|+1}, \ldots, e_{t-1}\right)\right\}$
2: $\hat{M} \leftarrow \{h \in M \mid |h| \geq |h'| \ \textit{for all}\ h' \in M\}$
3: **let** $h_{max} \in \left\{h \in \hat{M} \mid c(h) \geq c(h') \ \textit{for all}\ h' \in \hat{M}\right\}$
4: **return** the prediction $\hat{e}_t = I_{h_{max}}$

new hypothesis $(C) \Rightarrow A$ is added to H. The predictions at $t = 5$ and $t = 6$ will be successful while $h : (C) \Rightarrow A$ will be selected at $t = 7$ and produce the wrong prediction. As a consequence, PSL will create a new hypothesis $h_{new} : (B, C) \Rightarrow C$. Source code from the implementation used in the present work is available online [7].

3.2 Making Predictions

After, or during, learning, PSL can be used to make predictions based on the sequence of passed events $\eta = (e_1, e_2, \ldots, e_t)$. Since PSL continuously makes predictions during learning, this procedure is very similar to the learning algorithm (Alg. 1). The prediction procedure is described in Alg. 2.

For prediction of a suite of future events, $\hat{e}_t$ can be added to η to create η'. Then repeat the procedure described in Alg. 2 using η' as event history.

3.3 Differences and Similarities between PSL and S-Learning

Like PSL, S-Learning is trained on an *event sequence* η. However, S-Learning does not produce hypotheses. Instead, knowledge is represented as *Sequences* ϕ, stored in a *sequence library* κ [43]. ϕ does not describe a relation between a body and a head, like hypotheses do. Instead, ϕ describes a plain sequence of elements $e \in \eta$. During learning, sequences are "grown" each time a matching pattern for that sequence appears in the training data. Common patterns in η produce long sequences in κ. When S-Learning is used to predict the next event, the beginning of each $\phi \in \kappa$ is matched to the end of η. The sequence producing the longest match is selected as a winner, and the end of the winning sequence is used to predict future events.

One problem with this approach, observed during our previous work with S-Learning [9], is that new, longer sequences, are created even though the existing sequence already has Markov property, meaning that it can predict the next element optimally. To prevent the model from getting unreasonably large, S-Learning implements a maximum sequence length m. As a result, κ becomes unnecessarily large, even when m is relatively low. More importantly, by setting the maximum sequence length m, a task-dependent modeling parameter is introduced, which may limit S-Learning's ability to model η.

PSL was designed to alleviate the problems with S-Learning. Since PSL learns only when it fails to predict, it is less prune to be overtrained and can employ an unlimited maximum sequence length without exploding the library size.

4 Evaluation

The PSL algorithm was tested on a Khepera II miniature robot [30]. In the first evaluation (Sect. 4.1), the performance of PSL on a playful LFD task is demonstrated. In a second experiment (Sect. 4.2), the prediction performance during training of PSL is compared to the performance of S-Learning, using recorded sensor and motor data from the robot. During both experiments, the robot is given limited sensing abilities using only its eight infrared proximity sensors mounted around its sides.

One important issue, promoted both by Rohrer et al. [41,40] and ourselves [10], is the ability to learn even with limited prior knowledge of what is to be learned. Prior knowledge is information intentionally introduced into the system to support learning, often referred to as *ontological bias* or *design bias* [10]. Examples of common design biases are pre-defined state specifications, pre-processing of sensor data, the size of a neural network or the length of a temporal window. While design biases help in learning, they also limit the range of behaviors a robot can learn. A system implementing large amounts of design bias will to a larger extent base its decisions not on its own experience, but on knowledge of the programmer designing the learning algorithm, making it hard to determine what the system has actually learned.

In addition to design bias, there are many limitations and constraints introduced by other means, e.g., by the size and shape of the robot including its sensing and action capabilities, structure of the environment and performance limitations of the computer used. These kinds of limitations are referred to as *pragmatical bias* [10]. We generally try to limit the amount of ontological bias, while pragmatical bias should be exploited by the learning algorithm to find useful patterns.

In the present experiments, the robot has no previous knowledge about its surroundings or itself. The only obvious design bias is the thresholding of proximity sensors into three levels, *far*, *medium* and *close,* corresponding to distances of a few centimeters. This thresholding was introduced to decrease the size of the observation space Y, limiting the amount of training required. An *observation* $y \in Y$ is defined as the combination of the eight proximity sensors, producing a total of 3^8 possible observations.

An *action* $u \in U$ is defined as the combination of the speed commands sent to the two motors. The Khepera II robot has 256 possible speeds for each wheel, producing an action space U of 256^2 possible actions. However, only a small fraction of these were used during demonstration.

The event sequence is built up by alternating sensor and action events, $\eta = (u_1, y_1, u_2, y_2 \ldots, u_k, y_k)$. k is here used to denote the current stage, rather than the current position in η denoted by t. Even though events is categorized into

observations and actions, PSL makes no distinction between these two types of events. From the perspective of the algorithm, all events $e_t \in \sum$ are discrete entities with no predefined relations, where $\sum = Y \cup U$.

In each stage k, PSL is used to predict the next event, given η. Since the last element of η is an observation, PSL will predict an action $u_k \in U$, leading to the observation $y_k \in Y$. u_k and y_k are appended to η, transforming stage k to $k+1$. This alternating use of observations and actions was adopted from S-Learning [42]. A stage frequency of 10 Hz was used, producing one observation and one action every 0.1 seconds.

4.1 Demonstration and Repetition

To evaluate the performance of PSL on an LFD problem, four tasks are defined and demonstrated using the Khepera II robot. *Task 1* involves the robot moving forward in a corridor approaching an object (cylindrical wood block). When the robot gets close to the object, it should stop and wait for the human teacher to "load" the object, i.e., place it upon the robot. After loading, the robot turns around and goes back along the corridor. *Task 2* involves general corridor driving, taking turns in the right way without hitting the walls and so on. *Task 3* constitutes the "unloading" procedure, where the robot stops in a corner and waits for the teacher to remove the object and place it to the right of the robot. Then the robot turns and pushes the cylinder straight forward for about 10 centimeters, backs away and turns to go for another object. *Task 4* is the combination of the three previous tasks. The sequence of actions expected by the robot is illustrated in Fig. 1. The robot starts by driving upwards in the figure, following the dashed line. until it reaches the object at the loading position. After loading, the robot turns around and follows the dashed line back until it reaches the unload position. When the cylinder has been unloaded (placed to the left of the robot), the robot turns and pushes the object. Finally, it backs away from the pile and awaits further instructions. The experimental setup can be seen in Fig. 2. Even though the setup was roughly the same in all experiments, the starting positions and exact placement of the walls varied between demonstration and repetition.

All tasks capture a certain amount of temporal structure. One example is the turning after loading the object in Task 1. Exactly the same pattern of sensor and motor data will appear before, as well as after, turning. However, two different sequences of actions is expected. Specifically, after the teacher has taken the cylinder to place it on the robot, only the sensors on the robot's sides are activated. The same sensor pattern appears directly after the robot has completed the 180 degree turn, before it starts to move back along the corridor. Furthermore, the teacher does not act instantly. After placing the object on the robot, one or two seconds passed before the teacher issued a turning command, making it more difficult for the learning algorithm to find the connection between the events. Even Task 2 which is often seen as a typical reactive behavior is, due to the heavy thresholding of sensor data, temporally demanding. Even longer temporal structures can be found in Task 3, where the robot must push the

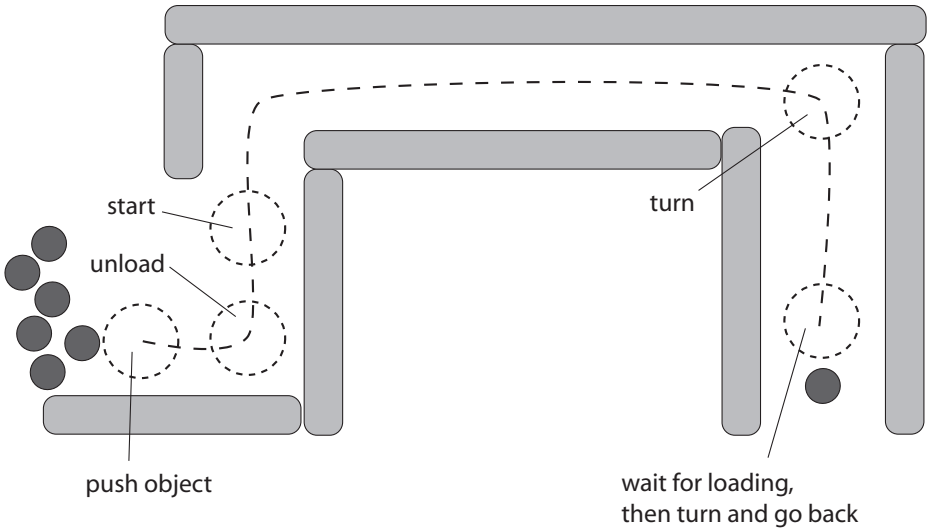

Fig. 1. Schematic overview of the composed behavior (*Task 4*). Light gray rectangles mark walls, dark gray circles mark the objects and dashed circles mark a number of key positions for the robot. See text for details.

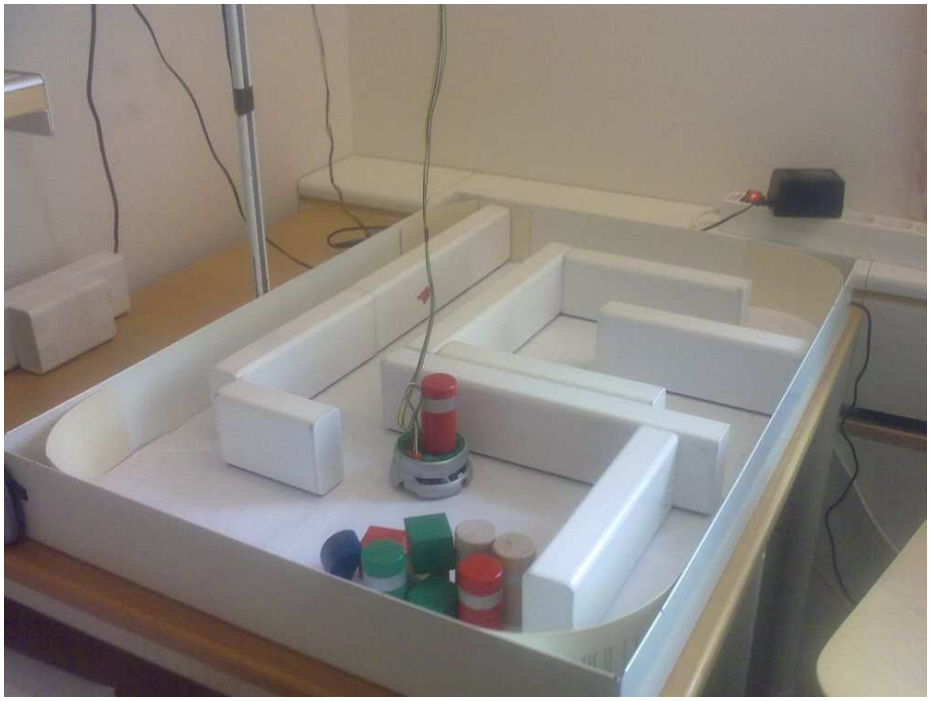

Fig. 2. Experimental setup.

object and remember for how long the object is to be pushed. This distance was not controlled in any way, making different demonstrations of the same task containing slightly conflicting data.

After training, the robot was able to repeat Task 1, 2 and 3 successfully. For Task 1, seven demonstrations were used for a total of about 2.6 min. Task 2 was demonstrated for about 8.7 min and Task 3 was demonstrated nine times, in total 4.6 min. The robot made occasional mistakes in all three tasks, reaching situations where it had no training data. In these situations it sometimes needed help to be able to complete the task. However, the number of mistakes clearly decreased with increased training, and mistakes made by the teacher during training often helped the robot to recover from mistakes during repetition.

For Task 4, the demonstrations from all three partial tasks were used, plus a single 2 min demonstration of the entire Task 4. Even after extensive training, resulting in almost 40 000 hypotheses in library, the robot was unable to repeat the complete behavior without frequent mistakes. Knowledge from the different sub-tasks was clearly interfering, causing the robot to stop and wait for unloading when it was supposed to turn, turning when it was supposed to follow the wall and so on. Detailed results for all four tasks can be found in Table 1.

Table 1. Detailed statistics on the four evaluation tasks. Training events is the number of sensor and motor events in demonstrated data. Lib. size is the number of hypotheses in library after training. Avg. $|h|$ is the average hypothesis length after training.

Task	Training events	Library size	Avg. $\|h\|$
Task 1	3102	4049	9.81
Task 2	10419	30517	16
Task 3	5518	8797	11
Task 4	26476	38029	15

PSL was trained until it could predict about 98% of the demonstrated data correctly. It would be possible to train it until it reproduces all events correctly, but this takes time and initial experiments showed that it did not affect the imitation performance significantly.

4.2 Comparison between S-Learning and PSL

In Sect. 3.3, a number of motivations for the design of PSL were given, in relation to S-Learning. One such motivation was the ability to learn and increase the model size only when necessary. S-Learning always learns and creates new sequences for all common events, while PSL only learns when prediction fails. However, it should be pointed out that even though S-Learning never stops to learn unless an explicit limit on sequence length is introduced, it quickly reduces the rate at which new sequences are created in domains where it already has extensive knowledge.

To evaluate the effect of these differences between PSL and S-Learning, prediction performance and library size were measured during training in three test cases. *Case 1* contained a demonstration of the loading procedure (*Task 1*) used in the LFD evaluation, Sect. 4.1. During the demonstration, the procedure was repeated seven times for a total of about 150 seconds (3000 sensor and motor events). *Case 2* encapsulated the whole composed behavior (*Task 4*) used in LFD evaluation. The behavior was demonstrated once for 120 seconds (2400 events). *Case 3* constituted 200 seconds of synthetic data, describing a 0.1 Hz sinus wave discretized with a temporal resolution of 20 Hz and an amplitude resolution of 0.1 (resulting in 20 discrete levels). The 4000 elements long data sequence created a clean repetitive pattern with minor fluctuations due to sampling variations.

In addition to PSL and S-Learning, a first order Markov model (*1MM*) was included in the tests. The Markov model can obviously not learn the pattern in any of the three test cases perfectly, since there is no direct mapping $e_t \Rightarrow e_{t+1}$ for many events. Hence, the performance of 1MM should be seen only as reference results.

The results from the three test cases can be seen in Fig. 3. The upper part of each plot show accumulated training error over the demonstration while lower parts show model growth (number of hypotheses in library). Since the Markov model does not have a library, the number of edges in the Markov graph is shown, which best corresponds to sequences or hypotheses in S-Learning and PSL, respectively.

5 Description

A novel robot learning algorithm called *Predictive Sequence Learning (PSL)* is presented and evaluated in an LFD setting. PSL is both parameter-free and model-free in the sense that no ontological information about the robot or conditions in the world is pre-defined in the system. Instead, PSL creates a state space (hypothesis library) in order to predict the demonstrated data optimally. This state space can thereafter be used to control the robot such that it repeats the demonstrated behavior.

In contrast to many other LFD algorithms, PSL does not build representations from invariants among several demonstrations that a human teacher considers to be "the same behavior". All knowledge, from one or several demonstrations, is stored as hypotheses in the library. PSL treats inconsistencies in these demonstrations by generating longer hypotheses that will allow it to make the correct predictions. In this way, the ambiguous definitions of *behavior* is avoided and control is seen purely as a prediction problem.

In the prediction performance comparison, PSL produces significantly smaller libraries than S-Learning on all three data sets. The difference is particularly large in Case 3 (Fig. 3), where both algorithms learn to predict the data almost perfectly. In this situation, S-Learning continues to create new sequences, while PSL does not.

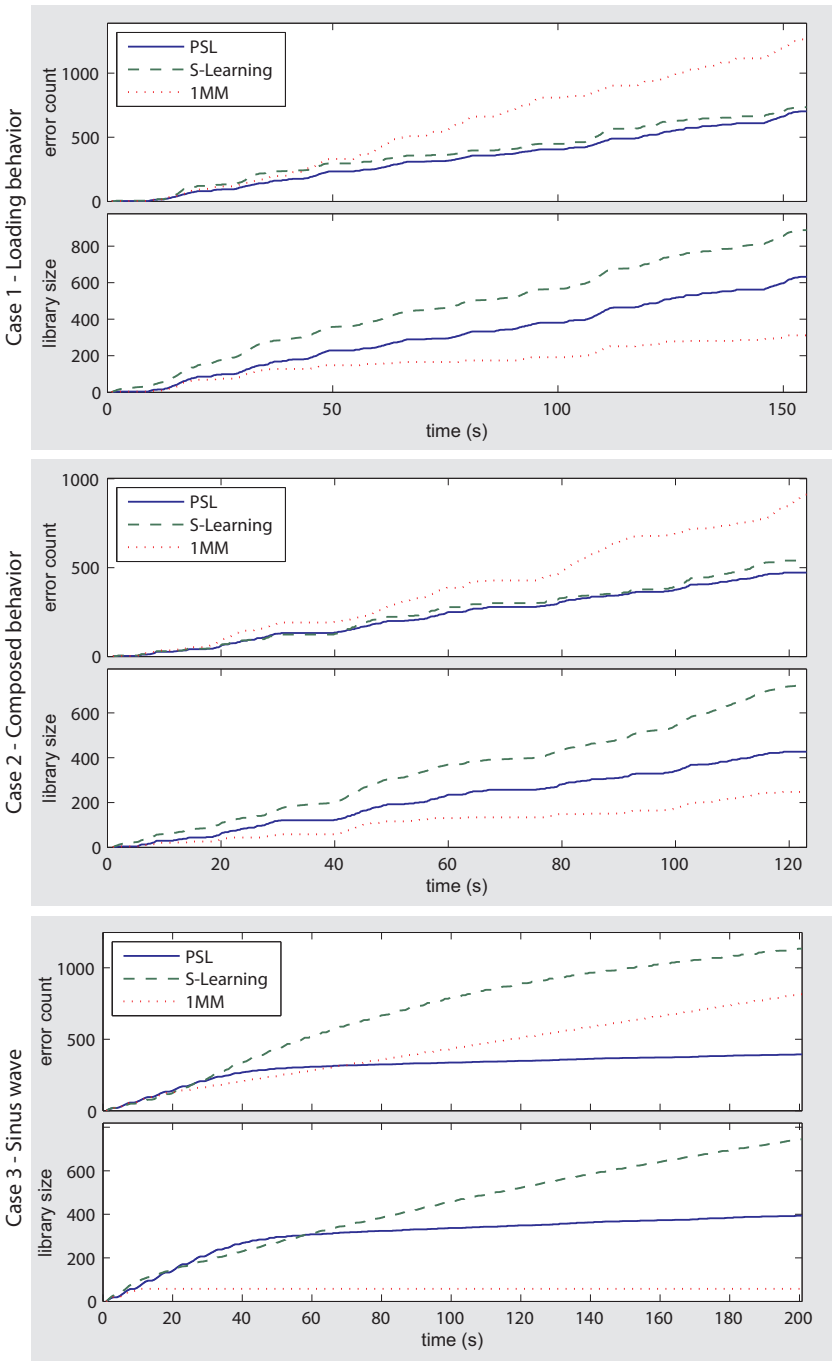

Fig. 3. Training results for all three test cases. See Sect. 4.2 for details.

In Case 3, PSL also shows the clearly fastest learning rates (least accumulated errors). The reason can be found in that PSL learns on each event where it fails to predict, while S-Learning learns based on sequence length. When the model grows, S-Learning decreases its learning rate even though the performance is still low. In contrast, the learning rate of PSL is always proportional to performance, which can also be seen in the plots for all three test cases (Fig. 3). However, even though PSL commits less accumulated errors than S-Learning in all three tests, the performance difference in Case 1 and 2 is small and how these results generalize to other kinds of data is still an open question.

In the demonstration-repetition evaluation, tasks 1, 2 and 3 were repeated correctly. Even though the robot made occasional mistakes, the imitation performance clearly increased with more demonstrations. However, in Task 4, which was a combination of the three first tasks, an opposite pattern could be observed. Despite the fact that PSL was still able to predict demonstrated data almost perfectly, knowledge from the three elementary tasks clearly interfered. The reason for this interference is that Task 4 requires much longer temporal dynamics than any of the elementary tasks did when learned separately.

One example of how this knowledge interference is manifested is the turning versus unloading. When the robot approaches the position marked as *turn* in Fig. 1, coming from the left and is supposed to take a right turn, it no longer sees the right wall behind it. Consequently, the situation looks identical to that of unloading. When the robot is to unload, it goes downward in Fig. 1 (position *unload*) but instead of turning it must wait for the cylinder to be placed to its right side. To make the right prediction, PSL has to base its decision on information relatively far back in the event history. Even though PSL has no problem to build a sufficiently large model from training data, the large temporal window produces a combinatorial explosion and the chance of the right patterns reappearing during repetition is small. As a result, PSL decreases the temporal window (i.e., uses shorter hypotheses), and the two situations become inseparable.

5.1 Conclusions and Future Work

The results show that the proposed algorithm is feasible for LFD problems up to a certain complexity. PSL implements very few assumptions of what is to be learned and is therefore likely to be applicable to a wide range of problems.

However, PSL also shows clear limitations when the learning problem increases and longer temporal dynamics is required. PSL is subject to combinatorial explosion and the amount of required training data increases exponentially with problem complexity. In these situations, some higher-level coordination is clearly necessary. One possible solution is to place PSL as a module in a hierarchical system. PSL learns both to predict sensor data as a response to action (forward model) and to select actions based on the current state (inverse model). In the present work, PSL is viewed purely as a controller and the forward model is consequently not considered. However, in work parallel to this, we show that PSL can also be used as an algorithm for behavior recognition [11], i.e., as a

predictor of sensor values. A big advantage of using PSL for both control and behavior recognition is that the forward and inverse computations are in fact based on the same model, i.e., the PSL library. This approach has several theoretical connections to the view of human perception and control as two heavily intertwined processes, as discussed in Section 2.1.

The present work should be seen as one step towards a hierarchical control architecture that can learn and coordinate itself, based on the PSL algorithm. The model-free design of PSL introduces very few assumptions into learning, and should constitute a good basis for many types of learning and control problems. Integrating PSL as both forward and inverse model to achieve a two-layer modular control system, is the next step in this process and will be part of our future work.

Acknowledgments

We would like to thank Brandon Rohrer at Sandia National Laboratories and Christian Balkenius at Lund University for valuable input to this work.

References

1. R. C. Arkin. *Behaviour-Based Robotics*. MIT Press, 1998.
2. R. Begleiter and G. Yona. On prediction using variable order markov models. *Journal of Artificial Intelligence Research*, 22:385–421, 2004.
3. A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*. Springer, 2008.
4. A. Billard, Y. Epars, G. Cheng, and S. Schaal. Discovering imitation strategies through categorization of multi-dimensional data. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2398–2403 vol.3, 2003.
5. A. Billard and M. J. Mataric. Learning human arm movements by imitation:: Evaluation of a biologically inspired connectionist architecture. *Robotics and Autonomous Systems*, 37(2-3):145–160, November 2001.
6. E. A. Billing. Representing behavior - distributed theories in a context of robotics. Technical report, UMINF 0725, Department of Computing Science, Umeå University, 2007.
7. E. A. Billing. Cognition reversed. http://www.cognitionreversed.com, 2009.
8. E. A. Billing. *Cognition Reversed - Robot Learning from Demonstration*. PhD thesis, Umeå University, Department of Computing Science, Umeå, Sweden, December 2009.
9. E. A. Billing and T. Hellström. Behavior recognition for segmentation of demonstrated tasks. In *IEEE SMC International Conference on Distributed Human-Machine Systems*, pages 228 – 234, Athens, Greece, March 2008.
10. E. A. Billing and T. Hellström. A formalism for learning from demonstration. *Paladyn: Journal of Behavioral Robotics*, 1(1):1–13, 2010.
11. E. A. Billing, T. Hellström, and L. E. Janlert. Behavior recognition for learning from demonstration. In *Proceedings of IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, May 2010.

12. R. A. Brooks. A robust layered control system for a mobile robot. In *IEEE Journal of Robotics and Automation RA-2*, volume 1, pages 14–23, 1986.
13. R. A Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.
14. R. A. Brooks. Intelligence without reason. *Proceedings, 1991 Int. Joint Conf. on Artificial Intelligence*, pages 569–595, 1991.
15. R. A. Brooks. New approaches to robotics. *Science*, 253(13):1227–1232, 1991.
16. S. Calinon, F. Guenter, and A. Billard. On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation*, 37(2):286–298, 2007.
17. N. Delson and H. West. Robot programming by human demonstration: The use of human inconsistency in improving 3D robot trajectories. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems '94. Advanced Robotic Systems and the Real World, IROS '94.*, volume 2, pages 1248–1255, Munich, Germany, September 1994.
18. J. Demiris and G. R. Hayes. Imitation as a dual-route process featuring predictive and learning components: a biologically plausible computational model. In *Imitation in animals and artifacts*, pages 327–361. MIT Press, 2002.
19. Y. Demiris and M. Johnson. Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning. *Connection Science*, 15(4):231–243, 2003.
20. Y. Demiris and G. Simmons. Perceiving the unusual: Temporal properties of hierarchical motor representations for action perception. *Neural Networks*, 19(3):272–284, April 2006.
21. M. Feder and N. Merhav. Relations between entropy and error probability. *IEEE Transactions on Information Theory*, 40(1):259–266, 1994.
22. K. J. Friston. Learning and inference in the brain. *Neural Networks: The Official Journal of the International Neural Network Society*, 16(9):1325–52, 2003. PMID: 14622888.
23. D. George. *How the Brain might work: A Hierarchical and Temporal Model for Learning and Recognition.* PhD thesis, Stanford University, Department of Electrical Engineering, 2008.
24. D. George and J. Hawkins. A hierarchical bayesian model of invariant pattern recognition in the visual cortex. In *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN'05)*, volume 3, pages 1812–1817 vol. 3, 2005.
25. F. Guenter, M. Hersch, S. Calinon, and A. Billard. Reinforcement learning for imitating constrained reaching movements. *RSJ Advanced Robotics, Special Issue on Imitative Robots*, 21(13):1521–1544, 2007.
26. M. Haruno, D. M. Wolpert, and M. Kawato. Hierarchical MOSAIC for movement generation. In *International Congress Series 1250*, pages 575– 590. Elsevier Science B.V., 2003.
27. M. Haruno, D. M. Wolpert, and M. M. Kawato. MOSAIC model for sensorimotor learning and control. *Neural Comput.*, 13(10):2201–2220, 2001.
28. J Hawkins and S Blakeslee. *On Intelligence.* Times Books, 2002.
29. M. Jordan and D. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science: A Multidisciplinary Journal*, 16(3):354, 307, 1992.
30. K-Team. Khepera robot. http://www.k-team.com, 2007.
31. M. Kawato, K. Furukawa, and R. Suzuki. A hierarchical neural-network model for control and learning of voluntary movement. *Biological Cybernetics*, 57(3):169–185, 1987. PMID: 3676355.

32. T. S. Lee and D. Mumford. Hierarchical bayesian inference in the visual cortex. *J Opt Soc Am A Opt Image Sci Vis*, 20(7):1448, 1434, July 2003.
33. R. C. Miall and D. M. Wolpert. Forward models for physiological motor control. *Neural Netw.*, 9(8):1265–1279, 1996.
34. C. L. Nehaniv and K. Dautenhahn. Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. In J. Demiris and A. Birk, editors, *Learning Robots: An Interdisciplinary Approach*, volume 24, pages 136–161. World Scientific Press, 2000.
35. R. Pfeifer and C. Scheier. Sensory-motor coordination: the metaphor and beyond. *Robotics and Autonomous Systems*, 20(2):157–178, June 1997.
36. R. Pfeifer and C. Scheier. *Understanding Intelligence*. MIT Press. Cambrage, Massachusetts, 2001.
37. T. Poggio and E. Bizzi. Generalization in vision and motor control. *Nature*, 431(7010):768–774, October 2004.
38. M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–25, November 1999. PMID: 10526343.
39. B. Rohrer. S-Learning: a biomimetic algorithm for learning, memory, and control in robots. In *CNE apos;07. 3rd International IEEE/EMBS Conference on Natural Engineering*, pages 148 – 151, Kohala Coast, Hawaii, 2007.
40. B. Rohrer. S-learning: A model-free, case-based algorithm for robot learning and control. In *Eighth International Conference on Case-Based Reasoning*, Seattle Washington, 2009.
41. B. Rohrer, M. Bernard, J. D. Morrow, F. Rothganger, and P. Xavier. Model-free learning and control in a mobile robot. In *Fifth International Conference on Natural Computation*, Tianjin, China, 2009.
42. B. Rohrer and S. Hulet. BECCA - a brain emulating cognition and control architecture. Technical report, Cybernetic Systems Integration Department, Univeristy of Sandria National Laboratories, Alberquerque, NM, USA, 2006.
43. B. Rohrer and S. Hulet. A learning and control approach based on the human neuromotor system. In *Proceedings of Biomedical Robotics and Biomechatronics, BioRob*, 2006.
44. S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 358(1431):537–547, March 2003. PMC1693137.
45. H. A. Simon. *The Sciences of the Artificial*. MIT Press, Cambridge, Massachusetts, 1969.
46. D. M. Wolpert. A unifying computational framework for motor control and social interaction. *Phil. Trans. R. Soc. Lond.*, B(358):593–602, March 2003.
47. D. M. Wolpert and J. R. Flanagan. Motor prediction. *Current Biology: CB*, 11(18):729–732, 2001.
48. D. M. Wolpert and Z. Ghahramani. Computational principles of movement neuroscience. *Nature Neuroscience*, 3:1212–1217, 2000.
49. J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.