Dogged Learning for Robots

Daniel H Grollman and Odest Chadwicke Jenkins
Brown University Department of Computer Science
Providence, RI 02912
{dang, cjenkins}@brown.edu

Abstract— **Ubiquitous robots need the ability to adapt their behaviour to the changing situations and demands they will encounter during their lifetimes. In particular, non-technical users must be able to modify a robot's behaviour to enable it to perform new, previously unknown tasks. Learning from Demonstration is a viable means to transfer a desired control policy onto a robot and Mixed-Initiative Control provides a method for smooth transitioning between learning and acting. We present a learning system (Dogged Learning) that combines Learning from Demonstration and Mixed Initiative Control to enable lifelong learning for unknown tasks. We have implemented Dogged Learning on a Sony Aibo and successfully taught it behaviours such as mimicry and ball seeking.**

I. Motivation

If robots are ubiquitous then the average user will likely be non-technical, but will have the desire to alter robot behaviour. That is, he or she will be unwilling or unable to sit down and program a robot in a traditional manner, either due to lack of time or relevant training, but would still like to modify the robot's actions. To perform this customization, users will rely upon whatever flexibility has been programed into the robot. Unfortunately, determining in advance what tasks users will want a robot to perform is difficult due to many social and economic factors. There are some fixed, limited autonomy tasks such as floor cleaning that can be universally assumed and robots can be and have been designed to perform these tasks well. However, we would like robots to move beyond the realm of single tasks toward more general functionality. Ideally we would like a robot to be able to perform any task asked of it that is physically possible given its embodiment. For robots that measure their lifetimes in years instead of minutes it is likely that many different tasks will be asked of it during operation, including tasks not thought of during the original designing and programming phases. We assume that it is impractical to program a robot in advance to execute any possible task in any situation that it might face. Therefore, we must utilize learning to enable a robot to change its behaviour after leaving the factory. Of particular interest is how a robot can learn to perform **"The Unknown Task"**, a task that it has previously never seen.

Learning from Demonstration (LfD) is an approach especially suited for robotic interaction with non-technical users. In this paradigm the robot is shown examples of the desired behavior and infers the demonstrator's latent control policy. The demonstration itself can take many forms, from watching a demonstrator to being physically guided through the task. Likewise, the demonstrator can be realized in multiple fashions. Possibilities include humans, other robots, or custom-designed algorithms such as planners. Once a control policy is learned, the robot then applies it to new situations, generalizing from the demonstration.

In a real world lifelong learning scenario the distinction between learning sessions and acting sessions (so necessary to laboratory research) becomes blurred. Not only does a robot need to learn new tasks during "learning time", but it should be able to modify those tasks on the fly while acting. This adjustment ability is particularly important when a robot has learned a task incorrectly and needs to be corrected, or when new situations are encountered that the robot was not adequately trained for. **Mixed Initiative Control** (MIC) is one method for enabling this constantly tunable learning. In a MIC setup, the robot's autonomous behaviours (learned or preprogrammed) share control of the robot with another agent. This agent can be a human, or for the purposes of our discussion, any demonstrator. By smoothly transitioning control between the robot and the demonstrator, MIC allows for quick and easy transitions in and out of training sessions.

Dogged Learning (DL) embraces all of these approaches by providing the ability to reprogram a robot for a previously unknown task using learning from demonstration in a mixed initiative manner. The ultimate desire is to have a robot that can be shown a new task by a human demonstrator, learn the task, acknowledge gaps in its abilities, and be amenable to future tuning as the human's needs and wants change.

II. Background

Robotic Learning from Demonstration is an approach to robot programming where a control policy is estimated from demonstrated examples of a task. The robot assumes the demonstration is consistent with "correct" behavior for the task and attempts to learn to recreate it. The robot's observations are the perceptual inputs and action outputs representative of the demonstrator's control policy. The primary challenge is to estimate this latent policy such that it can be used to drive the robot autonomously. This learning problem can be considered as related to supervised learning (i.e., regression) and reinforcement learning [1], [2], [3].

Robotic **Lifelong Learning** [4], deals with learning multiple tasks over time. In our approach, we focus on using a general regression mechanism with human intervention to learn and adapt the robot's control policy over its lifetime. Our aims are similar to those of reinforcement learning,

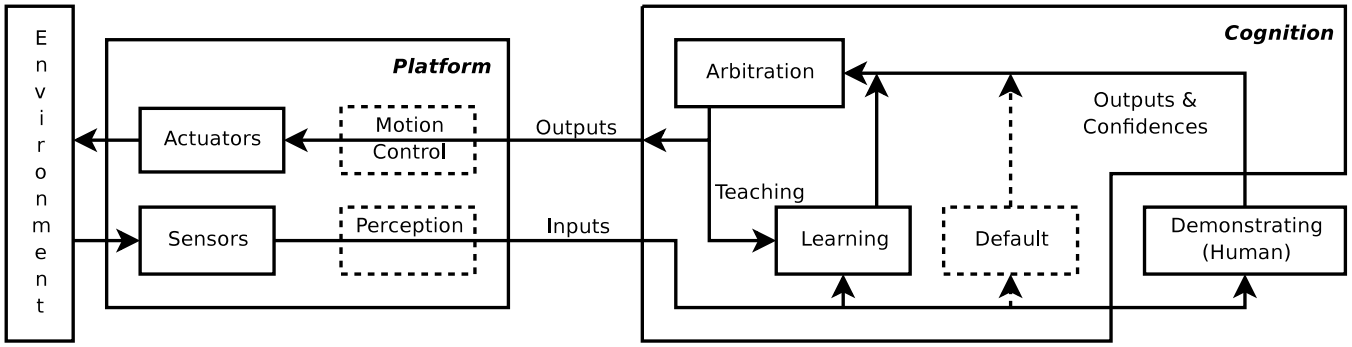


Fig. 1. The Dogged Learning system; dotted lines indicate optional portions. The platform (robot) provides inputs in the form of sensor data that has perhaps been put through some perception process. These inputs are passed to both the demonstrator and the cognition system (a learner and an optional default controller). Each box generates an output and a confidence value. The arbitrator receives these outputs and confidences and generates (chooses) an output to pass back to the platform and the learner. The learner then updates its I-O mapping and the platform interprets and acts upon the output.

except that we use human-robot interaction rather than autonomous exploration to drive the learning process. Our use of regression emphasizes generic input-output pairs and therefore we do not want to use an algorithm that is tied to particular data types. In addition, we would like to learn the mapping in an incremental, online way. We expect data to arrive one at a time and would like new data to cause an update in our learned method as it arrives. This is in contrast to a batch process, where large amounts of data are processed simultaneously. Locally Weighted Projection Regression (LWPR) [5] is a widely used regression technique that is well suited to our approach. It instantiates the concept of receptive fields to form a piecewise approximation of the input-output mapping and update it incrementally. Note that the system we propose is not tied to this specific regressor. In fact, other learning algorithms can be utilized in its place (such as Support Vector Regression [6] or Gaussian Processes [7]), and DL can be used as a means of comparing the abilities of these different learning algorithms.

Mixed Initiative Control is a rapidly growing area of research, studying the sharing of robot control between a human operator and an autonomous controller. Similar to Actor-Critic learning [8], MIC allows a human to guide a robot's learning process. Most work, however, deals with static autonomy - autonomous behaviors that are either preprogrammed, or learned once, and do not change over the course of operation [9]. Recently [10] showed a system that used sliding autonomy to enable one user to control three robots. In contrast, the autonomous behavior in robots utilizing DL can change over time to react to and incorporate what the human user does.

We also use MIC as a means to prompt the demonstrator for additional examples of the correct control policy in action. In contrast to traditional reinforcement learning, we consider demonstrator feedback a crucial aspect of a tractable and practical learning process. Our approach is to endow the robot with a sense of "confidence", essentially whether its previous experiences are applicable to its current situation, that allows it to ask for instructive interactions which may or may not occur. This confidence-based arbitration is similar to the use of social cues in the examples of [11] and the "responsibility estimator" of [12]. Note the robot's internal notion of confidence is a compliment to externally generated learning interventions (which also occur in our system), such as in the approach to feedback used by [13].

III. Dogged Learning

Dogged Learning is the cognitive learning setup shown in Figure 1. It deals with abstract perceived inputs and action outputs, so that very few constraints can be made upon the implementation. In particular, DL does not depend on what the platform is, nor what occupies the learning, demonstrating, arbitration and default boxes. We wish for DL to be a general cognitive engine that can be used for multiple purposes, having drawn inspiration from Monte-Carlo Localization, which often serves as a "black box" for higher level planning.

All DL requires from the various boxes is that:

- The platform generates inputs.
- The demonstrating, learning (and default) boxes generate an output and confidence given an input.[1]
- The arbitrator accepts outputs and confidences and generates an output.
- The learning box accepts an output and learns to associate it with the corresponding input.
- The platform accepts outputs.

It is important to note that DL is doubly agnostic as to the nature of its inputs and outputs. Not only does it not care what hardware it is running on, but it also does not care what pre- or post- processing is performed on the inputs and outputs that it receives and sends. It is only necessary that the information provided be sufficient for the desired task to be learned. This is easily tested for by seeing if the demonstrator is able to demonstrate the desired task based on the inputs and outputs. This agnosticism allows the same DL system to be applied at multiple levels of sensory input from raw sensor readings to heavily processed and manipulated

[1]We refer to the items in the demonstrating box as the "demonstrator" or "teacher" and items in the learning box as the "learner" or "student".

Algorithm 1 Predictive Learning

Input:
`threshold`, the error threshold,
P, a machine that generates `inputs`,
D, a machine that generates the correct `outputs`
L, the learning predictor.
Output:
L, performing well.

1: `error` $\leftarrow \infty$
2: **while** `error` $>$ `threshold` **do**
3: Get an `input` from P
4: Query D with `input` to get the correct `output`
5: Query L on `input` to get $\hat{\texttt{out}}$
6: Train L on the pair `input,output`
7: `error` $\leftarrow (\hat{\texttt{out}} - \texttt{output})^2$
8: **end while**

Algorithm 2 Dogged Learning

Input:
P, a machine that generates `inputs` and accepts `outputs`,
D, a demonstrator that generates the correct `outputs`
L, the learning predictor.
Output:
None, infinite loop

1: **while** ∞ **do**
2: Get an `input` from P
3: Query D on `input` to get `out`, `con`
4: Query L on `input` to get $\hat{\texttt{out}}$, $\hat{\texttt{con}}$
5: Arbitrate between `out` and $\hat{\texttt{out}}$ using `con` and $\hat{\texttt{con}}$ to determine `output`
6: Train L on the pair `input,output`
7: Send `output` to P
8: **end while**

data. Likewise, outputs can range from direct control of the actuators of the hardware to higher-level control signals. The complexity of the inputs and outputs are not limited by DL, but rather by the algorithms resident in the learning (and demonstrating and default) boxes as they are the ones which must process the inputs to produce outputs.

Our discussion of DL starts with a basic predictive learning loop, shown in Algorithm 1. This loop is used to train a learner to predict what the demonstrator will do, previous to using the learner to control some system. When prediction error falls below a pre-set threshold, the loop terminates and returns the trained learner. Then the system transitions into an acting state and behaves autonomously. Note that this system is a form of batch learning, once the system stops learning, it never returns to the learning state. Our first modification of this system is to add the lifelong component, so that the learner never stops learning. This is easily done by turning the loop into an infinite loop, thereby eliminating the need for a pre-determined termination condition. Now, as we no longer wish to train and use the predictor separately, we must connect the loop with the platform. Thus, after receiving an input from the platform, and both the learner and the demonstrator have been queried for an appropriate output, we must generate an output to send back to the platform (or machine). A first pass attempt would simply say that the teacher is always right - if the teacher returns an output, that response is sent to the platform and the input-output pair is given to the learner to learn. If the teacher does not generate an output, or generates a null output, then the student's output is instead passed to the hardware. Given a patient teacher and a good learning algorithm, this scenario will teach the system to mimic the instructor.

If the instructors are infinitely patient, then this scenario is all we need. We would like, however, for the instructors to be humans and the whole process human-usable. Therefore, we cannot expect a response to every input, as very often the inputs arrive at framerates faster than a human can process. We also expect the human to be lazy and only wish to respond if necessary to improve the system's behaviour or teach a new task. For this reason we have the requirement that along with an output, the learning and demonstrating boxs must generate a confidence value for each input query. Many learning algorithms, including LWPR, already have this capability, where confidence is a measure of how well the query input is supported by previous experience.

This confidence value can be used in many ways. Firstly, it can be used to recognize when the system has adequately learned a task by operating as the inverse of the error value in the simple algorithm. When the confidence value associated with queries is high, it means that the system has enough information to make a good prediction of the appropriate output. Demonstration may then cease and the platform can proceed to act autonomously.

Conversely, when the confidence value falls, it means that the learning algorithm is operating in space that it is unfamiliar with and perhaps its predicted outputs should not be trusted. In this case, signals can be sent to the demonstrator requesting more education (giving initiative to the teacher), or a default controller can be activated to prevent damage to the system.

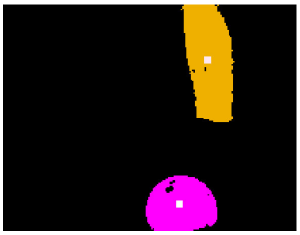
Furthermore, as we ask the demonstrator to return a confidence value as well, both of these values can be used to arbitrate control of the system. It is possible that the demonstrator returns an output but is, for some reason, unsure as to how correct it is while the learner has greater confidence. Such a scenario might arise if a new teacher has arrived and the student can leverage its previous education. In such a case we would like the learner to control the system until the demonstrator gains confidence.

The full DL algorithm with these modifications is shown in Algorithm 2.

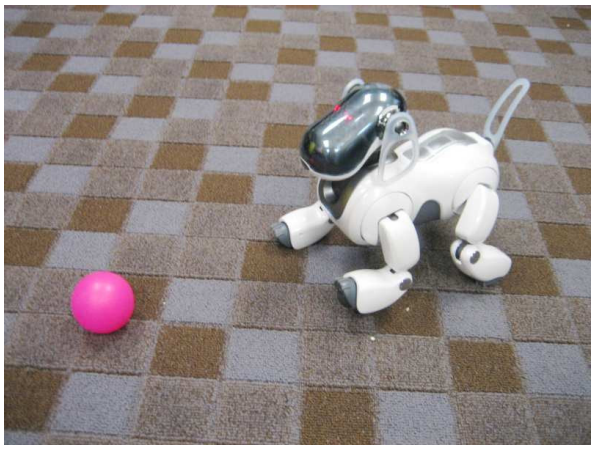
IV. Implementation

We have implemented DL for a Sony Aibo, pictured in Figure 2. Computational processing (aside from sensor and motor IO) is performed off-board and information is exchanged with the robot via wireless UDP communication.

(a) JPEG



(b) Segmented



(c) Dog and Ball

Fig. 2. Our robot, a Sony Aibo, and its ball. 2(a) shows the image obtained from the nose camera and 2(b) shows what our perceptual system perceives.

The robot generates the positions of its 18 motors as well as a JPEG image from a nose-mounted camera as sensor inputs and accepts settings for all 18 motors as actuator outputs.

Our DL implementation is built around LWPR in the learning box. In order to control for demonstrator error we have replaced the human demonstrator with hand coded controllers (described in experiments). However, we still have a human user in the loop who chooses when to activate the demonstrator and can do so for arbitrarily short or long periods of time. For arbitration we have adopted the simple strategy of always assuming that the demonstrator is correct (demonstrator confidence is always 100%). If the demonstrator returns a non-null output it is sent to the platform. Otherwise, the student-generated output is sent.

Feedback to the user was given via LEDs on the robot itself. The robot had 3 modes; 'teacher', 'student' and 'unsure' signalled by red, blue and green colors on the ears. These modes correspond to: 1) The teacher being in control of the robot, 2) The student being in control and fairly sure of what it was doing and 3) The student being in control, but unsure of its abilities and requesting more education. This correspondes to low confidence ($< 25\%$). Whether or not the education is provided depends on the human user.

While our implementation uses fixed-dimensionality inputs and outputs, this is not a requirement of the DL system, but instead is a limitation of our current implementation. We have nevertheless been able to use our setup in various configurations and do not see this limitation as impeding.

(a) Pan Left

(b) Pan Right

(c) Tilt Forward

(d) Tilt Back

Fig. 3. The first task we set our system was to learn to match the robot's head to its tail (mirroring). When we used one instructor to teach the pan portion (top row), and another to teach the tilt (bottom row), DL enabled the system to learn and merge both into full mirroring.

V. Experiments

Each of our experiments involved connecting the DL system to different platforms and demonstrators. This meant that the inputs and outputs of the system changed, but the DL system itself did not need to be modified. We held our learning box constant over the experiments presented here but, as previously mentioned, that is not required.

A. Mirroring

We first used DL to learn the simple cognitive task shown in Figure 3. The desired behavior is for the dog's head to 'mirror' its tail - as the tail was panned and tilted, the head should do the same. The setup follows:

- Inputs to the system were the 18 raw readings from the robot's motors.
- Outputs from the system were the 18 desired settings of the robot's motors.
- The demonstrator was a hand-coded controller that mapped tail readings to head settings.

This task was successfully learned after very little training. At startup, the student knows nothing and does not respond

to changing inputs. However, after less than 10 seconds of demonstrator activation, the student can perform the task on its own. Note that this task is not just to learn the tail-head mapping, but to also learn to ignore extraneous inputs. The readings corresponding to the other 16 motors are ignored and can be changed at will with no effect of the behaviour of the system.

We modified the above task slightly by replacing the one demonstrator with two teachers, each of which taught a different aspect of the desired task. One taught the panning portion of the mirror task and the other the tilting portion. Trained with either teacher alone the system is able to recreate that teacher's portion of the task. However, when trained with the two teachers successively, the student learns to merge their behaviours into the desired task. This demonstrates the ability of DL to learn from multiple instructors in an extended time scenario and combine the taught actions into novel behaviours.

B. Ball Seeking

We then applied DL to a very different task; that of ball seeking. This new behaviour, pictured in Figure 4 and shown in the accompanying video, is precursor to many desired skills such as foraging and goal scoring. We do not expect the system to be able to learn this task from raw sensor inputs, and so we have enabled some pre- and post- processing.

- The inputs to the system are locations of color blobs extracted from images from the robot's camera.
- The outputs of the system are movement vectors, which are transformed into robot walking behaviours.
- The demonstrator is a hand-coded algorithm that directs the robot to seek out and approach pink balls.

Our DL system was put *unmodified* into this setup. Again, as the system is agnostic as to the nature of the inputs and outputs, it does not matter what the platform provides. Therefore, no modification of DL was needed to apply it to this new task. The only changes required were to insert the pre- and post- processing algorithms between DL and the robot and to replace the demonstrator.

Like the basic task before it, this task was successfully learned by our system. After a short primary training session ($<$ 1 minute), the student began to exhibit ball-seeking behavior. Further short ($<$ 10 second) activations of the demonstrator serve to correct any deviations from this task. These shorter activations can be applied at will by the user, but are most helpful to the student when its confidence is low (the robot is in 'unsure' mode).

An interesting emergent property of the system appeared during this experiment - the system generalizes as much as possible from the inputs. What occurred is that there were 2 balls (in addition to much clutter) in the robot's environment, one pink and one orange. If the teacher has never been active when the orange ball is present, the student lacks information as to what to do when viewing it. In this case, the student approaches the orange ball in the same manner as it would the pink one. If, however, the teacher is activated, the student quickly learns to ignore the orange ball and respond only to the pink one.

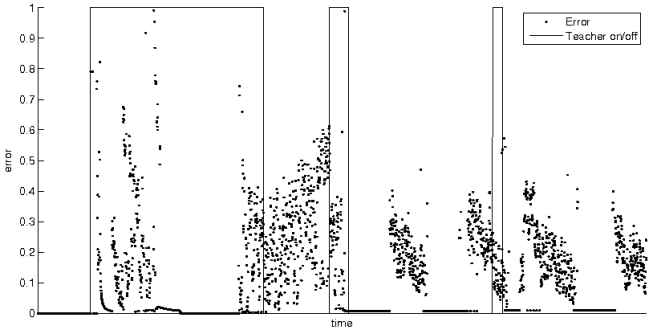


Fig. 5. Error versus time for a run of the system performing the ball-seeking task. Also plotted is an indicator variable showing when the teacher is active. When the teacher is first activated, error goes high and then quickly falls as the student learns the task. In the absence of instruction, error rises again but further education reigns it in. The system then stabilizes into a characteristic error profile over the various portions of the task. Time shown: 2 minutes.

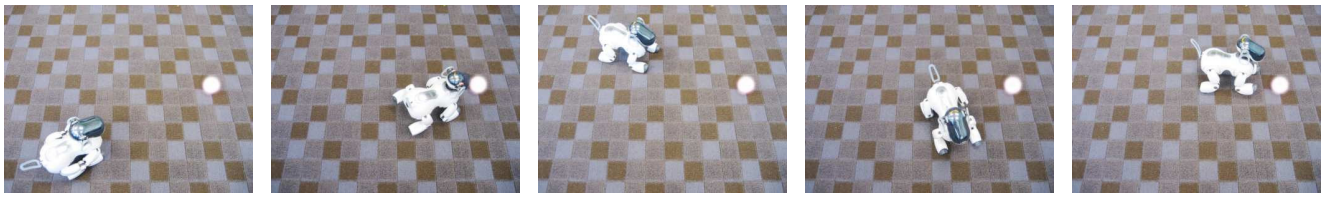
The learning abilities of the system were evaluated qualitatively to ensure that the desired behaviour was actually learned. Visual inspection of the student controlling the robot shows the robot performing the task that it was taught. We have also carried out a quantitative evaluation using our hand-coded controllers. To pin a number on how well our system learns, we ran the demonstrator program on all of the data seen by the student (both data that the student received instruction for and data that the student acted on by itself). With the outputs from the teacher ($\mathtt{out}$) and from the student ($\hat{\mathtt{out}}$) we can generate an error score, $\mathtt{error} = \sqrt{(\mathtt{out} - \hat{\mathtt{out}})^2}$. This error score is plotted over time in Figure 5, along with an indicator variable showing when the teacher was active.

VI. Discussion

DL has been designed to be *agnostic* to platforms, demonstrators, and learners, and thus applicable in many situations and to many different tasks. All three of these boxes warrant further discussion and deserve further research.

A. The Learning Box

DL is not tied to any particular learning algorithm. The current system is attached to a particular learning scenario (incremental real-time online learning), but the exact details of how that learning takes place is unspecified. For this reason we see DL as a means to directly compare different approaches to this learning problem. By training a system using one learning box and then retraining it using a different one and comparing the resulting behaviours, different learning algorithms can be pitted against each other. By operating on log files we can relax the need for real-time computation and try out more computationally intensive algorithms. We also believe that it is possible to relax the incremental requirement and run the system in a quasi-batch mode by performing batch learning after each 'training session'. We will explore this possibility in future work.



(a) Untrained Student (b) Teacher demonstrates (c) Student generalizes (d) Assistance Required (e) Student succeeds

Fig. 4. The Ball seeking task. At first (a), the new student knows nothing and does not react to the ball. After the teacher demonstrates the seek behavior (b), the student can perform it as well (c). When encountering a new situation (d), the student continues to behave but requests more education. After the user delivers more training, the student can successfully complete its task (e). (Ball highlighted for visibility)

B. The Demonstration Box

DL places very few bounds on what serves as a demonstrator. Conceptually, DL treats the demonstrator as a kind of intermittent oracle, an entity that has a latent control policy and gives outputs representing this policy in return for inputs, but not all of the time. In the experiments discussed here, all of our demonstrators have been hand-coded controllers that have been toggled on and off by human users. The use of compiled code has allowed us to provide demonstration outputs at the input framerate, as well as avoid some of the inconsistencies associated with human users. Our future goal is to allow human users to not only control the presence or absence of instruction, but to also dictate the content of the instruction, perhaps using teleoperation. In this manner we hope to allow non-technical users to teach our robots to perform various tasks.

C. The Platform Box

The inputs and outputs of the DL system are also left underdefined. This allows it to operate in many different perceptual and action spaces depending on how much pre- and post- processing occurs. By connecting DL directly to the sensors and motors of a robot, we can teach simple behaviours such as the head/tail mirroring discussed above. However, by inserting more advanced perceptual and action systems between the hardware and the DL system, learning can take place on these perceptual concepts (such as a pink ball or walking). It is important that the perception and action concepts provided be sufficient for the desired task, as we cannot ask the robot to learn to do a task that it does not have the tools to perform. For this reason we require that the demonstrator be able to perform the task given access to the same inputs and outputs. Furthermore, multiple copies of DL can interact with each other and a system that utilizes DL with various amounts of pre- and post- processing could learn cognitive tasks at multiple resolutions.

VII. Conclusion

If robots are to be ubiquitous and interact with non-technical users, they will require methods of modifying their behaviour over the course of their lifetime. Here we have presented a learning system that enables a robot to learn tasks from demonstration in a mixed-initiative manner. The system, Dogged Learning, deals with abstractions of learning and demonstration and is thus well suited for comparing different learning algorithms and demonstrative techniques. As implemented on a Sony Aibo with LWPR learning and hand-coded demonstration, DL has successfully been used to learn multiple cognitive behaviours.

Acknowledgements

This work was supported in part by the NSF (IIS-0534858) and the Brown Salomon Fund. The authors would also like to thank the members of Brown # (Ethan F. Leland, Brendan C. Dickinson and Mark L. Moseley) for their assistance.

References

[1] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *14th Intl. Conf. on Machine Learning (ICML)*, D. H. Fisher, Ed., Nashville, TN, 1997, pp. 12–20.

[2] W. D. Smart and L. P. Kaelbling, "Effective reinforcement learning for mobile robots," in *2002 IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 4, Washington, D.C., 2002, pp. 3404–3410.

[3] A. L. Thomaz and C. Breazeal, "Transparency and socially guided machine learning," in *5th Intl. Conf. on Developmental Learning (ICDL)*, 2006.

[4] S. B. Thrun and T. M. Mitchell, "Lifelong robot learning," University of Bonn, Tech. Rep. IAI-TR-93-7, Jan 1993.

[5] S. Vijayakumar, A. D'Souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[6] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and other kernelbased learning methods*. Cambridge, UK: Cambridge University Press, 2000.

[7] D. J. C. Mackay, *Neural Networks and Machine Learning*. Springer-Verlag, 1998, ch. Introduction to Gaussian Processes, pp. 84–92.

[8] M. Rosenstein and A. Barto, *Learning and Approximate Dynamic Programming: Scaling Up to the Real World*. New York: John Wiley and Sons, Inc., 2004, ch. Supervised actor-critic reinforcement learning, pp. 359–380.

[9] J. A. Adams, P. Rani, and N. Sarkar, "Mixed initiative interaction and robotic systems," Vanderbilt, Tech. Rep. WS-04-10, 2004.

[10] F. W. Heger and S. Singh, "Sliding autonomy for complex coordinated multi-robot tasks: Analysis and experiments," in *Robotics: Science and Systems II (R:SS)*, Philidelphia, PA, 2006.

[11] C. Breazeal, A. Brooks, J. Gray, G. Hoffman, C. Kidd, H. Lee, J. Lieberman, A. Lockerd, and D. Chilongo, "Tutelage and collaboration for humanoid robots," *International Journal of Humanoid Robotics*, vol. 1, no. 2, pp. 315–348, June 2004.

[12] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural Networks*, vol. 11, no. 7-8, pp. 1317–1329, 1998.

[13] M. Nicolescu and M. J. Matarić, "Natural methods for robot task learning: Instructive demonstration, generalization and practice," in *2nd Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Melbourne, AUSTRALIA, 2003, pp. 241–248.