5-2009

Automatic Weight Learning for Multiple Data Sources when Learning from Demonstration

Brenna Argall
Carnegie Mellon University

Brett Browning
Carnegie Mellon University

Manuela M. Veloso
Carnegie Mellon University

Published In

Proceedings of the IEEE International Conference on Robotics and Automation (ICRA09) , 226-231.

Automatic Weight Learning for Multiple Data Sources when Learning from Demonstration

Brenna D. Argall and Brett Browning and Manuela Veloso

***Abstract*— Traditional approaches to programming robots are generally inaccessible to non-robotics-experts. A promising exception is the *Learning from Demonstration* paradigm. Here a *policy* mapping world observations to action selection is learned, by generalizing from task demonstrations by a teacher. Most Learning from Demonstration work to date considers data from a single teacher. In this paper, we consider the incorporation of demonstrations from multiple teachers. In particular, we contribute an algorithm that handles multiple data sources, and additionally reasons about *reliability* differences between them. For example, multiple teachers could be inequally proficient at performing the demonstrated task. We introduce *Demonstration Weight Learning (DWL)* as a Learning from Demonstration algorithm that explicitly represents multiple data sources and learns to select between them, based on their observed reliability and according to an adaptive *expert learning* inspired approach. We present a first implementation of DWL within a simulated robotic domain. Data sources are shown to differ in reliability, and weighting is found impact task execution success. Furthermore, DWL is shown to produce appropriate data source weights that improve policy performance.**

I. INTRODUCTION

As robots become more prevalent within general society, the need for programming techniques that are accessible to non-experts increases. To develop a control *policy*, or mapping from world observation to action selection, traditional approaches first model world dynamics and then derive the policy mathematically. Though theoretically well-founded, these approaches depend heavily on the accuracy of the world model, which requires considerable expertise to develop. Most other approaches to policy development are similarly restricted in use to robotics-experts.

One potential exception is policy development through *Learning from Demonstration (LfD)*, e.g. [4], [7]. In this paradigm, a teacher first demonstrates a desired behavior to the robot. The robot then generalizes from these examples to derive a policy. Demonstration has the attractive feature of being an intuitive communication medium for humans, who already use demonstration to teach other humans. Since it does not require expert mathematical knowledge of the system dynamics, demonstration also opens policy development to non-robotics-experts.

In this work, we explicitly consider LfD policy development that incorporates demonstration data from multiple sources. We argue that this consideration is particularly relevant for LfD robotics applications in general society, where multiple users of a robot system is likely. Multiple users easily translates into multiple policy developers, and therefore into multiple LfD teachers.

We further posit that all LfD data sources might not be equally reliable. Multiple teachers may differ in their respective abilities to perform a given task. Another possibility is that sources produce data in fundamentally different ways. For example, in [1] demonstration data is produced from a non-traditional method, by synthesizing it from learner executions modified by advice from a human teacher.

We introduce *Demonstration Weight Learning (DWL)* as an algorithm that incorporates data from multiple demonstration sources. DWL considers reliability, by assigning a weight to each data source. The weight is automatically determined and updated by the algorithm, based on learner performance using the LfD policy. Though the algorithm is general to any sort of LfD domain, we focus the scope of this work to motion control tasks.

In this paper, we introduce an algorithm that explicitly addresses reliability differences between multiple data sources. DWL weights each data source, and adaptively updates these weights automatically. A first implementation of this algorithm within a simulated robotic domain is presented, and data source weighting is found impact task execution success. Furthermore, DWL is shown to produce appropriate data source weights that improve policy performance.

The remainder of the paper is organized as follows. Section II motivates weighting multiple LfD data sources, while providing related work. The DWL algorithm is presented in Section III, followed by implementation details within a simulated robotic domain in Section IV. Empirical results are presented and discussed within Section V, and in Section VI we conclude.

II. MOTIVATION AND RELATED WORK

We begin with a discussion of Learning from Demonstration and the related work that motivates our consideration of multiple weighted demonstration sources.

A. Learning from Demonstration

During an LfD teacher execution, world state observations and action selections are recorded. Formally, our world consists of states S and actions A, with the mapping between

B. Argall and B. Browning are with the Robotics Institute, and M. Velsoso the Computer Science Department, at Carnegie Mellon University, Pittsburgh, PA 15213, USA. <bargall,brettb,mveloso>@cs.cmu.edu

This research was sponsored in part by the Boeing Company under Grant No. CMU-BA-GTA-1. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied of the Boeing Company. The authors Brenna Argall and Brett Browning were partially supported by Carnegie-Mellon University in Qatar and the Qatar Foundation.

states by way of actions being defined by the probabilistic transition function $T(s'|s,a) : S \times A \times S \rightarrow [0,1]$. We assume that state is not fully observable. The learner instead has access to observed state Z, through the mapping $M : S \rightarrow Z$. A teacher demonstration $d_j \in D$ is represented as t_j pairs of observations and actions such that $d_j = \{(\mathbf{z}_j^i, \mathbf{a}_j^i)\} \in D, \mathbf{z}_j^i \in Z, \mathbf{a}_j^i \in A, i = 0 \cdots k_j$. Within the typical LfD paradigm, the set D of these demonstrations are provided to the learner. A policy $\pi : Z \rightarrow A$, that selects actions based on observations of the world state, is then derived.

LfD has found success on a variety of robot platforms and applications. Key design decisions include the demonstration approach that produces the data, and then how a policy is derived from that data.

Approaches for executing and recording teacher demonstrations range from teleoperating a passive robot platform [9] to recording sensors worn by a human [5]. Our simulated robot is teleoperated while recording from its own sensors, as this minimizes correspondence issues between demonstrator and learner on real platforms. We restrict our scope to human demonstrators, though the algorithm itself is general to any teacher.

The most popular approaches for deriving a policy from demonstration data are to (i) directly approximate the underlying function mapping observations to actions [4], (ii) use the data to determine the world dynamics model $T(s'|s,a)$ [8] or (iii) provide a planner with a learned model of action pre- and post-conditions [7]. Our work derives a policy using the first mapping function approximation approach.

B. Multiple Weighted Data Sources

In this work, we introduce an algorithm that reasons explicitly about demonstration data from multiple sources. One motivating factor for this is our contention that multiple human teachers is a real possibility for LfD systems used within general society. Previous work within LfD considers multiple demonstration teachers for the purposes of isolating salient characteristics of the task execution [9]. To speed up learning, demonstration information is solicited from multiple other agents in the work of Oliveira and Nunes [8], and Argall et al. employ additional data sources to address circumstances under which teacher execution is difficult or inefficient [1].

We additionally consider the unequal reliability of different data sources. For example, human teachers may differ in their respective abilities to perform a given task, or sources may produce data in fundamentally different ways. We further consider that the reliability of a data source is not static. For example, data based on learner executions [1] may become more reliable as the learner becomes more proficient at reproducing the target behavior.

Along a similar vein to our consideration of data source reliability, previous LfD work considers the worth of demonstrated executions. For example, Kaiser et al. actively identify and remove unnecessary or inefficient elements from a teacher demonstration [6].

Our work is unique in combining these two LfD considerations of multiple data sources and reliability. We make use of demonstration data produced through traditional means (e.g. teacher execution of a behavior), as well as through non-traditional means (e.g. modification of a learner execution). Our DWL algorithm addresses the issues of data source reliability and stochasticity by assigning a weight to each data source and dynamically updating that weight as the policy develops.

III. ALGORITHM

In this section we present the Demonstration Weight Learning algorithm. DWL is an LfD approach characterized by the explicit incorporation and weighting of multiple demonstration data sources. Details of the algorithm execution are first presented, followed by those of the dynamic weight update.

A. Execution Overview

DWL incorporates demonstration data from multiple sources c. The algorithm automatically determines a set of weights $\mathbf{w}$ over these data sources, to reflect the respective reliability of each source. These weights are then considered when deriving a policy π from the provided demonstration data D, and are dynamically updated in response to learner execution performance. Psuedo-code for the algorithm is presented in Figure 1.

```
00  Given  D, Z
01    init w ← 1
02    init π ← policyDerivation(D, w)
03    Given  z^goal ∈ Z
04      Execute
05        while z^t != z^goal
06          select   a^t ← π(z^t)
07          execute  a^t
08          record   tr ← { z^t, a^t, c^t }
09        end
10      Advisor Feedback
11        provide { d̂ }
12      Update
13        adapt    w
14        update   D ← { d̂ }
15        rederive π ← policyDerivation(D, w)
```

Fig. 1. Psuedo-code for the DWL algorithm.

The first phase of DWL consists of teacher demonstration. From this data an initial policy is derived (line 02). The single data source weight is initialized to 1, since initially data exists from only one source (the demonstration teacher).

The second phase of DWL consists of learner practice. For each practice run, an observed goal state $z^{goal} \in Z$ is provided (line 03). To begin the robot performs the task (*Execute*, lines 04-09). The advisor then provides new data in response to this performance (*Advisor Feedback*, lines 10-11). Lastly, the learner updates the data source weights, the

data set D and finally the policy π in response to its execution performance and the data feedback of the advisor (*Update*, lines 12-15).

During the *Execute* portion of a practice run, the learner first executes the task, producing execution trace tr. The learner executes until achieving goal state z^{goal}. At each timestep the learner selects action $\mathbf{a}^t$ according to $\pi(\mathbf{z}^t)$ (line 06), the details of which are provided in Section III-B. This action is recorded in the execution trace tr, along with observation $\mathbf{z}^t$ and the data source c^t that recommended the action (line 08).

During the *Advisor Feedback* portion of a practice run, the human advisor responds to execution performance by providing new data $\hat{d}$ [1]. This data may be generated by any data source. Our implementation employs both traditional and non-traditional data sources; further details are provided in Section IV-B. Alternatively, should the advisor determine that the policy execution was satisfactory, she may elect to provide no new data, in which case $\hat{d} \rightarrow \{\}$.

During the *Update* portion of a practice run, the data source weights and policy are updated. First, the data source weights $\mathbf{w}$ are adapted based on the learner execution performance (line 13), the details of which are provided in Section III-C. This occurs independently of whether or not new data will be introduced into the system, i.e. even if $\hat{d} \rightarrow \{\}$. Second, the set $\hat{d}$ of new data is added to the dataset D (line 14). A new policy π is then derived from the updated set D and adapted weights $\mathbf{w}$ (line 15).

For succinctness, we define $\phi_i^t \equiv I(c^t = c_i)$, as an indicator on whether source c^t is equal to the source c_i.

B. Policy Prediction and Data Source Selection

At each time step, an action is predicted by the policy. A data source is sampled at random from a uniform distribution weighted by source weights $\mathbf{w}$. The action prediction of the selected data source is the policy output.

To make a prediction, data sources may employ any sort of policy derivation technique appropriate to the underlying data, for example classification or regression techniques. Our specific implementation (Sec. IV) considers continuous-valued predictions and employs a form of Locally Weighted Learning [2].

Given current observation $\mathbf{z}^t$, the action $\mathbf{a}^t$ for source c_i is predicted through an averaging of the data points in D produced by c_i, $d_{\phi_i^t} \in D$, weighted by their kernelized distance to $\mathbf{z}^t$. Thus,

$$\mathbf{a}^t = \sum_{(\mathbf{z}_j, \mathbf{a}_j) \in D} h_j^t \cdot \mathbf{a}_j, \quad h_j^t = e^{\left(\mathbf{z}_j - \mathbf{z}^t\right)\Sigma^{-1}\left(\mathbf{z}_j - \mathbf{z}^t\right)^T} \tag{1}$$

where the weights h_j^t have been normalized over j, and Σ^{-1} is a constant parameter scaling observations within $[0, 1]$ for each dimension. Regression tuning is external to DWL and tied to policy derivation. In our implementation the distance computation is Euclidean, and the kernel is Gaussian.

C. Expert Learning and the Selection Weight Update

In DWL, data sources are selected according to source weights $\mathbf{w}$. These weights are dynamically updated (line 13) after every execution and in response to learner performance. To update these weights, DWL takes an *expert learning* approach.

Originally proposed by [10] to solve the *k*-armed bandits problem, expert learning addresses the issue of choosing between multiple action recommenders, or *experts*. In this paradigm, executing an action receives reward, for which the recommending expert is credited. An expert's selection probability is determined by its accumulated reward, such that high reward, and therefore good performance, increases its probability of being selected. DWL treats data sources as experts.

Formally, at each decision cycle, t, each of the n experts makes a recommendation. The algorithm selects a single expert and executes the corresponding action, resulting in a payoff of $r^t \in \Re$. After d decision cycles, a sequence of $r^1, r^2, \cdots, r^d$ payoffs have been rewarded. The aim of expert learning is to select the best expert over all decision cycles. We formulate this learning objective in terms of regret, where the regret at decision cycle k is the difference between the observed reward r^t of the selected action, and the reward r_b^t we would have received from the best expert. Summed over all decision cycles,

$$Regret = \sum_{t=1}^{d} r_b^t - \sum_{t=1}^{d} r^t. \tag{2}$$

The goal of expert learning is to minimize this total regret.

When executing physical actions, however, only the reward of the recommending expert is observed. Learning thus becomes a partial information game. Algorithm Exp3 [3] handles partial information games by scaling reward inversely with selection probability. The effective reward $\hat{r}_i^t$ earned by expert i from reward r_i^t is thus

$$\hat{r}_i^t = \frac{r_i^t}{P(\phi_i^t)} \tag{3}$$

where $P(\phi_i^t)$ is the probability of selecting expert i at decision cycle t. This compensates for the fact that experts with low probability are infrequently chosen, and therefore have fewer observed rewards. The selection weight of an executed expert i is then updated according to

$$w_i^t = e^{\hat{r}_i^t} w_i^{t-1} \tag{4}$$

and weights are initialized to be equal across experts. Note that the exponent product is equivalent to adding $\hat{r}_i^t$ to $\sum_k \hat{r}_i^{t-1}$, and thus represents the cumulative reward received by expert i up to trial t.

D. The Dynamic Weight Update in DWL

To update data source selection weights, DWL models its approach on the Exp3 expert learning algorithm [3]. Similar to Exp3, reward r_i^t is scaled inversely with data source selection probability $P(\phi_i^t)$ (Eq. 3). Further considerations

key to DWL are how to distribute individual rewards, and how to determine selection probabilities.

1) Assigning Individual Reward: A single reward r^t is received by DWL in response to a given learner execution. The learner execution consists of a sequence of timesteps. At each timestep, a single data source, or expert, is selected and its recommended action executed. Across the *sequence* of timesteps, therefore, multiple experts may have been selected. The single reward thus must be distributed amongst all contributing experts.

The contribution of expert i to the execution trace tr is computed as the fractional number of execution points for which i was the recommending expert. This contribution then combines with the reward r^t, so that the individual reward r_i^t received by expert i is computed according to

$$r_i^t = r^t \left(\frac{k_{tr_i}}{k_{tr}} \right) \tag{5}$$

$$k_{tr_i} = \sum_t \phi_i^t, \;\; k_{tr} = \sum_t 1$$

where k_{tr} is the number of execution timesteps in tr and k_{tr_i} the number of those which selected data source i.

2) Determining Selection Probabilities: In DWL, the selection probability of experts is governed by two factors. The first is data source weight. The second is data source *distribution* within the state space. This is a consequence of the LfD data acquisition paradigm in DWL. Task proficiency aside, sources are not trusted to predict in state-space areas where they have no support. This means that the probability of selecting a data source decreases in areas where it has not provided demonstration data, independent of its weight value.

Data sources are selected based on a combination of weight and distance to the query point. Concretely, given a current observation $\mathbf{z}^t$, the probability of selecting data source i is given by

$$p\left(\phi_i^t | \mathbf{z}^t\right) = w_i \cdot \min_j h_j^t, \;\; d_{\phi_i^j} \in D \tag{6}$$

where h_j^t is a function of query point distance, computed as in Equation 1, and $w_i \in \mathbf{w}$ is the weight of source i. To then determine the probability $P(\phi_i^t)$ of selecting expert i over the entire execution tr, an average is taken

$$P\left(\phi_i^t\right) = \frac{1}{k_{tr}} \sum_j p\left(\phi_i^j | \mathbf{z}^j\right), \;\; j = 1 \cdots k_{tr}. \tag{7}$$

Using Equations 5 and 7, the scaled reward required for the data source weight update (Eq. 4) may now be computed.

IV. EXPERIMENTAL SETUP

Here we present the details of a first implementation of the DWL algorithm, within a simulated robotic driving domain.

A. Racetrack Driving Domain

Empirical validation of the DWL algorithm is performed within a simulated robotic driving domain, reflecting our focus on motion control policies. The robot is tasked with driving along a racetrack while staying within the track bounds. Task execution ends when the robot either completes the track (10.34 m length) or drives out of bounds.

Robot sensing within the domain consists of three range-finders (oriented forward, left, and right of the robot body). Motion dynamics are governed by robot heading and speed. The system runs at 30Hz ($0.033s$), and changes in both heading and speed are bounded ($0.6m/s$ and $0.52rad$, respectively). Gaussian noise (5%) is added to observed range information, robot heading and executed speed.

During policy execution, the observations (continuous-valued) computed by the robot are 5-dimensional: left sensor range (lr), center sensor range (cr), right sensor range (rr), a ratio of the right and left ranges (lr/rr) and a binary flag indication of which track border (left or right) is observed by the center sensor. The actions (continuous-valued) predicted by the robot are 2-dimensional: target speed and change in heading. Teacher demonstrations are performed via teleoperation, with the teacher indicating to (in/de)crease speed or (in/de)crease heading.

B. Data Sources

Within the simulated racetrack driving domain, three data sources are available:

1) Teacher demonstration (Source T)
2) Confirmation of learner executions (Source C)
3) Advice-operator modified executions (Source A)

Data from source T is produced from task execution by a human teacher teleoperating the simulated robot. Source C data is produced from a learner execution. The teacher indicates a subset K of the execution which displayed good performance, and the learner adds recorded data $\{\mathbf{z}^t, \mathbf{a}^t\} \in K$ from this subset to the dataset D.

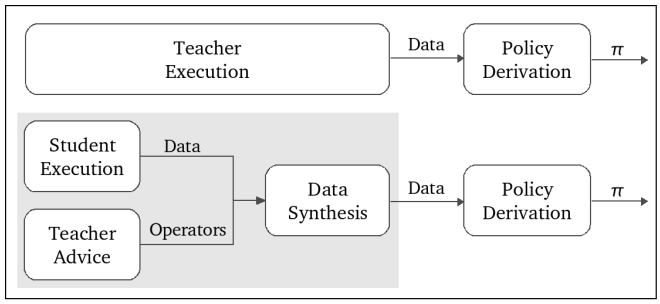

Fig. 2. Generating demonstration data. Traditional approaches provide demonstration data from teacher executions (top). One non-traditional approach generates data based upon learner executions and teacher advice (bottom, shaded box).

Source A is produced from a combination of learner executions and teacher advice (Fig. 2). The teacher corrects poor performance by selecting from a discrete number of advice-operators, and indicating a subset K of the execution

over which to apply the operator. *Advice-operators* perform simple mathematical computations that produce continuous-valued corrections for data point observations or actions. Applied to the learner execution trace, they result in synthesized, corrected, data. Three advice-operators were employed in this domain: *Turn* (less/more/none), *Change Speed* (less/more/none) and *Curve* (less/more). For a more complete discussion of advice-operators, we refer the reader to [1].

C. Evaluation Metrics

The evaluation metrics within this domain are defined as execution success and efficiency. *Success* is measured by the length of traversed track (m). *Efficiency* is measured by mean executed speed (m/s).

The weight update in DWL requires that the system reward learner executions. Note, however, that one reward is provided for the entire execution, and not at every execution timestep. The reward therefore only needs to evaluate overall performance, and be sufficiently rich to learn the data source *weights*; it is not necessary that it be sufficiently rich to learn the *task*. The reward in our system is formulated as a simple combination of success and efficiency, as defined above.

V. Empirical Results and Discussion

In this section empirical results from the DWL implementation are presented and discussed. During practice runs DWL successfully learns data source weights automatically, and execution success and efficiency are found to improve. These weights are shown to produce superior performance to equal data source weighting, and data sources are demonstrated to be unequally reliable in this domain.

A. Automatic Data Source Weighting

A practice run for the learner consists of a single track execution followed by teacher feedback. After each practice run, the data source weights are automatically updated. New data may additionally be available, depending on teacher feedback. If so, this data is also incorporated. After completing the practice runs, the result is a final dataset D, consisting of multiple data sources.

Figure 3 shows the results of weight updating and new data incorporation during these practice runs. Note that all data sources are not available at the start of the executions. In particular, initially only data source T (teacher demonstration) is present. Data from source A (advice-operator modified executions) next begins to be provided, since it is formulated from corrections on learner executions. Finally data source C (confirmation of learner executions) is introduced, only once the learner begins executing well enough to have its performance confirmed by the teacher. Note further that a given data source has zero weight before it is introduced into the data set.

The mean execution speed and distance traveled along the track during the practice runs are presented in Figure 4. Both measures are shown to improve as a result of learner practice.

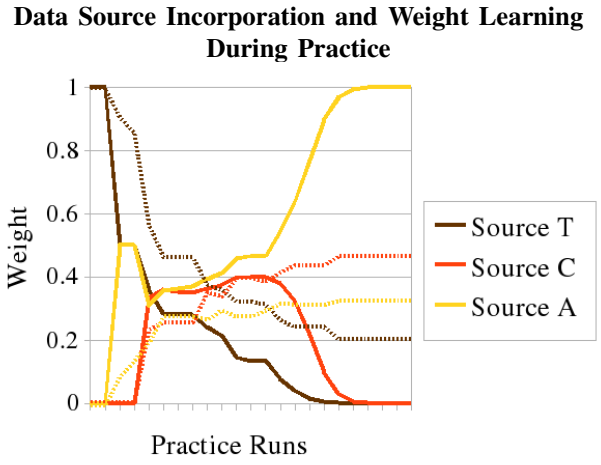

Fig. 3. Data source weight learning during the practice runs (solid lines). For reference, the fractional contribution (in number of data points) of each data source to the full data set D is additionally provided (dashed lines).

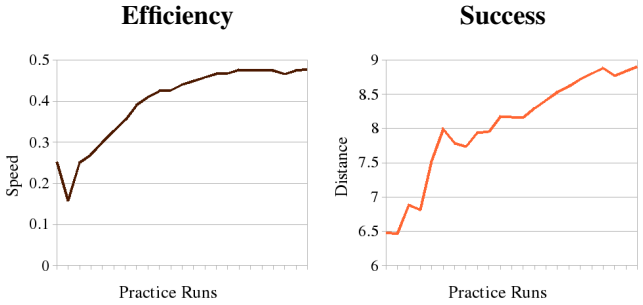

Fig. 4. Mean execution speed and distance traveled during the practice runs (running average, 23 practice runs).

B. Performance Improvement with Weighting

To examine the effect of data source weighting, executions on the test set were performed using all data sources and constant equal data source weights. The mean execution speed and distance results are presented in Figure 5 (green hashed bars, averaged over 20 runs, variance error bars). For comparison, test set execution using the final weights learned during practice is also presented (white hashed bars). In execution speed, the learned weighting displays superior performance over the equal weighting (top). In distance traveled, similar behavior is seen, with both giving near perfect performance (bottom).

To explore weight learning consistency, data source weights were relearned using the full final dataset D. No new data was added during these executions, and source weights were initialized to be equal. The weights learned during these executions are presented in Figures 6 (averaged over 50 runs, variance error bars).

C. Unequal Data Source Reliability

To further explore the reliability of each data source, test set executions were performed using exclusively one data source. The results of these executions are presented in Figure 5 (solid bars, averaged over 20 runs, variance error

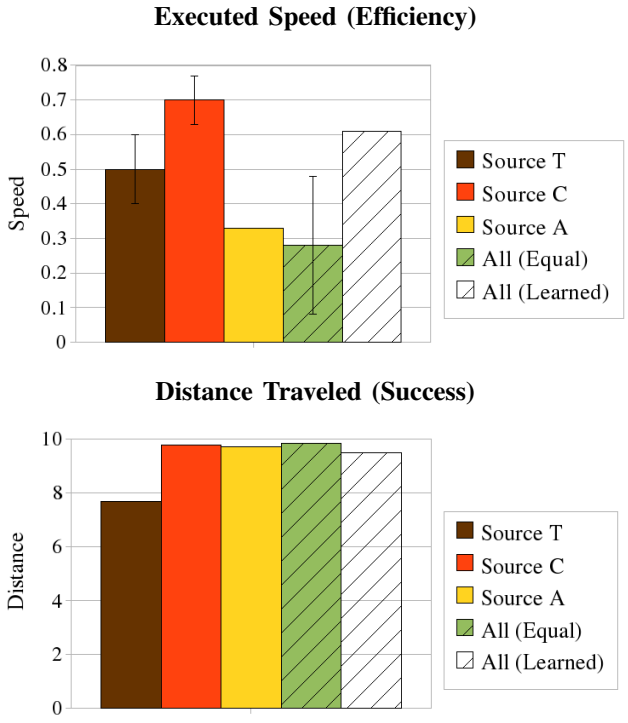

Fig. 5. Mean execution speeds and distances traveled during test set execution (mean of 20 test set executions, variance error bars). Executions with exclusively one data source (solid bars) are compared to executions using all data sources (hashed bars).

bars). The significance of the results are two-fold. First, they provide confirmation that in this domain the multiple data sources are indeed not equally reliable. Second, they reflect that the DWL algorithm is able to combine information from *all* of the data sources in such a manner as to perform similarly to the *best* performing data source.

D. Discussion

Data sources are shown to be unequally reliable within this domain (Fig. 5). Important to remember is that the learner is *not* able to choose the source from which it receives data. Furthermore, even a poor data source can be useful, if it is the only source providing data in certain areas of the state-space. DWL addresses both of these concerns. Though the learner is not able to choose the data source, it is able to favor data from certain sources through the DWL weighting scheme. Furthermore, when selecting a data source at prediction time, both this weight *and* the state-space support of the data source are considered.

For example, consider the low execution speed of data source A. This source generates data from correcting poor behavior. The developed advice-operators limit the magnitude of the correction, to keep the synthesized data close to what was actually executed and therefore close to the robot's known capabilities. Being closer to the poorly behaving data points, however, means that source A will not always have the greatest data. Nevertheless, the fact that the data addresses particular problem areas for the policy is important.

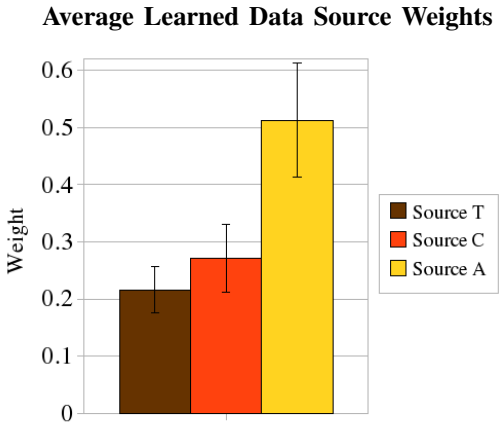

Fig. 6. Data source weights learned using the full combined data set D (averaged over 50 test set executions, variance error bars).

VI. CONCLUSION

In this work, we explore incorporating multiple data sources within a *Learning from Demonstration* paradigm. In particular, we consider *reliability* differences between multiple Learning from Demonstration data sources. *Demonstration Weight Learning (DWL)* is introduced as an Learning from Demonstration algorithm that explicitly reasons about multiple data sources, and through a weighting scheme leverages their reliability. Data source weights are automatically determined, and dynamically updated according to an adaptive *expert learning* inspired approach. A first implementation of DWL is presented within a simulated robotic driving domain. Data sources are found to differ in task execution reliability, and data source weighting is shown to impact task performance in both success and efficiency. Furthermore, DWL is shown to produce weights that enable similar performance to the best data source, and superior performance to no source weighting.

REFERENCES

[1] B. Argall, B. Browning, and M. Veloso. Learning robot motion control with demonstration and advice-operators. In *Proceedings of IROS '08*, 2008.

[2] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning for control. *Artificial Intelligence Review*, 11:75–113, 1997.

[3] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multiarm bandit problem. In *36th Annual Symposium on Foundations of Computer Science*, pages 322–331, Milwaukee, WI, 1995.

[4] D. C. Bentivegna. *Learning from Observation Using Primitives*. PhD thesis, College of Computing, Georgia Institute of Technology, 2004.

[5] S. Calinon and A. Billard. Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of HRI '07*, 2007.

[6] M. Kaiser, H. Friedrich, and R. Dillmann. Obtaining good performance from a bad teacher. In *Programming by Demonstration vs. Learning from Examples Workshop at ML'95*, 1995.

[7] M. N. Nicolescu and M. J. Mataric. Methods for robot task learning: Demonstrations, generalization and practice. In *Proceedings of AAMAS '03*, 2003.

[8] E. Oliveira and L. Nunes. *Learning by exchanging Advice*. Springer, 2004.

[9] P. K. Pook and D. H. Ballard. Recognizing teleoperated manipulations. In *Proceedings of ICRA '93*, 1993.

[10] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin American Mathematical Society*, 55:527–535, 1952.