# Learning from Demonstration using a Multi-valued Function Regressor for Time-series Data

Jesse Butterfield Sarah Osentoski Graylin Jay Odest Chadwicke Jenkins

***Abstract*— Using data collected from human teleoperation, our goal is to learn a control policy that maps perception to actuation. Such policies are potentially multi-valued with regard to perception with a single input mapping to multiple outputs depending on the user's objective at a particular time. We propose a multi-valued function regressor to learn a larger class of robot control policies from human demonstration and extend the Hierarchical Dirichlet Process Hidden Markov Model to discover latent variables representing unknown objectives in the demonstrated data and the transitions between these objectives. Each of these objectives requires only a single-valued policy function, and thus can be learned with a Gaussian process function regressor. The learned transitions between these objectives determine the correct actuation where the complete policy function is multi-valued. We present the results of experiments conducted on the Nao humanoid robot platform.**

## I. Introduction

We propose a method for multi-valued function regression for time-series data to directly learn robot policies from human demonstration data. More broadly, we are interested in how learning from demonstration (LfD) can be used to create autonomous robot controllers. LfD is an active area of research within the robotics community [1]. However, much progress remains to be made before policies learned from demonstration perform as well as hand-coded controllers. In this paper, we examine extending LfD techniques to simultaneously learn both low level primitive tasks in a continuous space as well as the transitions between the primitive tasks when faced with perceptual aliasing, when a single perception can map to more than one action.

J. Butterfield, S. Osentoski, G. Jay, and O.C. Jenkins are members of the Department of Computer Science, Mathematics and Computer Science, {jbutterf | sosentos | tjay | cjenkins }@cs.brown.edu

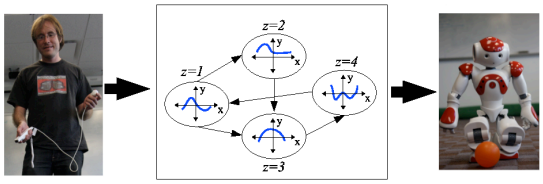


Fig. 1. This figure visualizes our proposed approach, in which a user demonstrates a desired policy through teleoperation. The robot learns a model consisting of single-valued function regressors and transitions between these regressors. The robot uses the learned model to do the behavior autonomously.

When a robot is shown a task, its perceptual information alone may be insufficient to determine its next action due to **perceptual aliasing**. For example, a robot that is opening a door may find itself with the same perceptual information (similar joint angles, visual information, and tactile sensing) as it does when closing a door. Information from onboard perception alone is not sufficient for the robot to decode whether it should push the door further closed or pull it further open. The robot must disambiguate the context by keeping some state information about whether it is currently opening or closing the door.

Our approach to creating robot control policies, visualized in Fig. 1, is to directly learn policy functions, mappings from perception to actuation, using data from human demonstration. These policy functions are potentially **multi-valued mappings**, with the same input value mapping to multiple output values, as opposed to typical single-valued functional mapping where each input maps to exactly one output.

We describe a multi-valued function regressor designed for time-series data to directly learn robot policies from human demonstration data. More formally, given a robot with perception and actuation, represented by the vectors $x$ and $y$ respectively, we want to learn a potentially multi-valued function, $\pi : x \rightarrow y$. The function regressor is trained using a demonstration of a task, which consists of one or more time series of perception-action pairs, $(x_t, y_t)$.

Objectives, or goals of the demonstrator that are not provided to the robot, are often a cause of perceptual aliasing. Our model considers objectives to be portions of the task that require only a single-valued policy function. Thus, a mathematically well-defined function, $\pi : x, z \rightarrow y$ where $z$ corresponds to the index of a particular single-valued policy function, can be learned. In the door example there are two separate objectives, opening and closing the door. When the objective is to open the door, the proper behavior is to continue pushing the door open. When the objective is to close the door, the proper behavior is to continue to pull the door closed. In this paper we extend the Hierarchical Dirichlet Process Hidden Markov Model (HDP-HMM) [2] to discover discrete latent variables $z$, which represent hidden objectives in the demonstrated data, and to learn the transitions between the objectives. The learned transitions between these objectives determine the correct single-valued policy where the complete policy, as a function of just the perception, is multi-valued. When the robot takes the action returned by this policy, it autonomously produces behaviors similar to those executed by the user during demonstration.

### A. Related Work

Robot learning from demonstration techniques have been developed by applying a variety of techniques from machine learning, including reinforcement learning [3], [4], inverse reinforcement learning [5], [6], classification [7], and regression [8]. In general, this work only learns the single-valued primitive subtasks or is given the primitive subtasks and learns the mapping between these subtasks.

Some research has examined learning from demonstration with perceptual aliasing. Chernova and Velosa [9] introduce the concept of equivalent action choices for situations where multiple actions can be taken, any of which would be correct. This work recognizes these situations as "equivalent actions" and allows the robot to randomly select a next action from the set of valid actions. It does not handle situations where the actions are not equivalent and hidden information is required to distinguish between the actions. Grollman [10] proposed using the Infinite Mixture of Gaussian Process Experts (IMoGPE) function regressor [11] to model multi-valued policy functions. This work segments the data into experts, each with a single-valued policy function. This approach avoids the situation where traditional regressors would average the two separate but applicable actions to get an incorrect action. However, this model assumes the datapoints are independent and identically distributed (i.i.d.) and ignores the time-series nature of the data. To conduct tasks where the action choices are equivalent, this method requires data have a state or objective label that can guide switching between experts.

Previous work in activity recognition examined the Abstract Hidden Markov Model (AHMM), an extension of HMMs that models the relationship between state, action, and higher level objectives [12], [13]. However, these works considered discrete actions, predefined objectives, and used some a priori knowledge of the relationships between particular states, actions, and objectives.

## II. Multi-valued Function Regression

In this section, we introduce our extension of the HDP-HMM to multi-valued function regression. In section III, we will discuss specifics of the model for inference and expert assignment.

Similar to the work of Grollman [10], we propose to use an **infinite mixture of experts**, where each expert corresponds to an objective with a single-valued policy. The state-action data is segmented into single-valued policies by assigning each ordered pair of perception and actuation from the demonstration to an expert $z_t$. A function regressor is associated with each regression expert, mapping

a perception $x$ to an actuation $y$ with a well-defined function. Unlike the previous work by Grollman, we consider time-series nature of data by including transition probabilities between the experts assigned to consecutive ordered pairs (see Fig. 2).

Our model resembles a second order Hidden Markov Model (HMM), but with a likelihood function now defined as $P(x_t, y_t|z_t, z_{t-1})$. Substituting this term into the traditional equation for an HMM and expanding using Bayes rule gives

$$P(z_{1:T}, x_{1:T}, y_{1:T}) \propto P(z_0) \cdot \prod_{t=1}^{T} P(z_t|z_{t-1}, \mathcal{P}_{z_{t-1},z_t}) \cdot P(x_t|z_t, z_{t-1}, \theta_{z_t,z_{t-1}}) P(y_t|x_t, z_t, \phi_{z_t}), \quad (1)$$

where $\mathcal{P}_{z_{t-1},z_t}$ is the probability of transitioning from objective $z_{t-1}$ to $z_t$, $\theta_{z_t,z_{t-1}}$ are the parameters of the density estimator of $P(x_t|z_t, z_{t-1})$, and $\phi_{z_t}$ are the parameters of the function regressor associated with expert $z_t$. We define $P(y_t|x_t, z_t, \phi_{z_t})$ to be conditionally independent of $z_{t-1}$ because the function for expert $z_t$ is single valued, and therefore $y_t$ can depend only on $x_t$ and $z_t$.

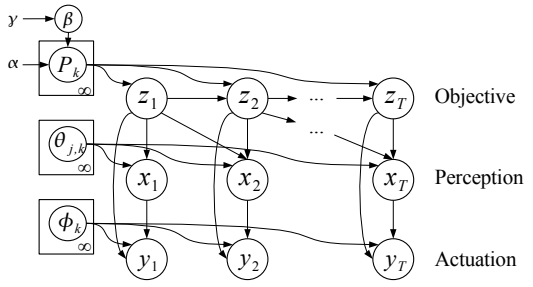


Fig. 2. The extension to the HDP-HMM. The state at time $t$ is $z_t \sim \mathcal{P}_{z_{t-1}}$ where $\mathcal{P}_k$ is $P(z_t|z_{t-1} = k)$. $\alpha$ and $\gamma$ are tunable parameters. $\beta \sim \text{GEM}(\gamma)$ is a prior on $\mathcal{P}$. $\theta_{j,k}$ are the learned parameters of the density estimator for $P(x_t|z_{t-1} = j, z_t = k)$ and $\phi_k$ are the learned parameters for the function approximator $P(y_t|x_t, z_t = k)$.

### A. Prediction

Once the demonstration data is assigned to experts and the associated parameters for each expert are learned, the trained model can be used for predicting actuation from perception for a single-valued policy subtask. Given perception $x_t$ and the previous expert $z_{t-1}$, the most likely expert and the associated actuation can be predicted by selecting the $\hat{z}_t$ and $\hat{y}_t$ that maximize

$$P(z_t, y_t|z_{t-1}, \mathcal{P}, x_t, \theta_{z_t}, \phi_{z_t}) \propto P(z_t|z_{t-1}, \mathcal{P}) P(x_t|z_t, \theta_{z_t}) P(y_t|x_t, z_t, \phi_{z_t}). \quad (2)$$

For any given $z_t$ this distribution will be maximized by $\hat{y}_t(z_t)$ defined as

$$\hat{y}_t(z_t) = \underset{y_t}{\text{argmax}}\, P(y_t|x_t, z_t, \phi_{z_t}). \quad (3)$$

There are a finite number of existing experts; thus we maximize the joint probability by searching over all possible $z_t$ to get

$$\hat{z}_t = \underset{z_t}{\text{argmax}}\, P(z_t|z_{t-1}, \mathcal{P}) P(\hat{y}_t(z_t)|x_t, z_t, \phi_{z_t}). \quad (4)$$

$$\hat{y}_t = \hat{y}_t(\hat{z}_t) \quad (5)$$

## III. Parameterization and Inference

In Section II, we proposed a model for learning from demonstration. In this section, we describe inference, parameterization, and implementation of this model, illustrated in Fig. 3.

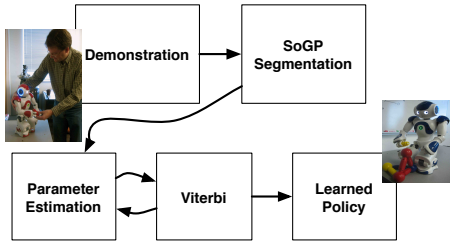


Fig. 3. Our implementation for learning from demonstration using multi-valued function regression.

### A. Inference

The model is initialized by sequentially and greedily assigning the demonstrated data to states according to

$$z_t = \underset{z}{\text{argmax}}\, P(y_t|x_t, z, \phi_z). \quad (6)$$

A new expert is created whenever the probability of assigning the datapoint to any of the existing experts is less than the probability of assigning it to an empty function regressor with mean zero and large standard deviation. The parameters for an expert are updated whenever a demonstration point is assigned to that expert.

The initial segmentation can then be used to estimate the model parameters, $\mathcal{P}$, $\theta$, and $\phi$, of the model. The Viterbi algorithm is used to calculate the posterior probability $P(z_t|x_{1:T}, y_{1:T}, \mathcal{P}, \theta, \phi)$ for every $z_t$ using the current model parameters, as well as including an empty function regressor to create a potential new expert. Each datapoint is assigned to the expert that maximizes that datapoint's posterior distribution. This new segmentation is used to update the parameters. This procedure is iterated until convergence when the segmentation remains the same on consecutive iterations or some maximum number of iterations is exceeded.

### B. Model Parameterization

To accommodate a potentially infinite mixture of experts, the prior on the transition between two experts, $i$ and $j$, is modeled as an HDP-HMM. The *maximum a posteriori* (MAP) estimate of the transition matrix, $\mathcal{P}$ is then approximately

$$\mathcal{P}_{i,j} \propto \begin{cases} (n_{ij} + \alpha n_j) & \text{if } j \text{ is an existing expert} \\ \alpha\gamma & \text{if } j \text{ is a new expert,} \end{cases} \tag{7}$$

where $n_{ij}$ is the number of transitions from expert $i$ to expert $j$, and $n_j$ is the total number of datapoints assigned to expert $j$. $\alpha$ controls the prior probability of a previously unseen transition, and $\gamma$ controls the prior probability of creating a new expert.

$P(x_t|z_t, z_{t-1}, \theta_{z_t,z_{t-1}})$ is modeled with a multivariate kernel density estimator. The estimator uses Gaussian kernels and a max leave-one-out likelihood criterion to determine the kernel width. We used an implementation by Aihler [14] of the algorithm by Gray and Moore [15] for performing inference on kernel density estimators.

$P(y_t|x_t, z_t, \phi_{z_t})$, the single-valued function regressor, is modeled using Sparse Online Gaussian Processes (SOGP) [16]. The function regressor returns MAP estimates $\hat{y}_t$ and $\hat{\Sigma}_t$ which are used to estimate $P(y_t|x_t, z_t, \phi_{z_t})$ according to

$$P(y_t|x_t, z_t, \phi_{z_t}) = \frac{1}{(2\pi)^{\frac{D}{2}}|\hat{\Sigma}|^{\frac{1}{2}}} \cdot \exp\left( -\frac{1}{2}(y_t - \hat{y}_t)'\hat{\Sigma}^{-1}(y_t - \hat{y}_t)\right). \tag{8}$$

Note that any function regressor that produces $\hat{y}_t$ and $\hat{\Sigma}_t$ can be used. We use Sparse Online Gaussian Processes (SOGP) [16] because it has been used to learn robot policy functions from demonstration and is online [8]. The perception data is normalized to be between $-1$ and $1$ with a fixed Gaussian kernel width $w$ and noise prior $\sigma_0$.

## IV. Experimental Results

In this section we experimentally evaluate our approach on a ball tracking task and a manipulation task, both performed with an Aldebaran Nao robot.

### A. Ball Tracking

In our first experiment, the robot learns to find and track a brightly colored orange ball by moving its head. The demonstration is provided by a hand-coded finite state machine with 3 states, *scan left*, *scan right*, and *track ball*. This controller scans all the way left and right when no blob is in view and uses a proportional controller to center the blob on the screen when it appears. The ball is randomly moved into and out of the robot's viewable environment. This task requires the learning from demonstration algorithm be capable of dealing with multi-valued demonstrations. When there is no blob on screen, the current head position is insufficient to determine if the robot should be scanning left or right. Selecting randomly between these two actions will cause the head to erratically oscillate, while averaging the actions will produce no movement.

The perception space consists of the visual perception, the pixel area, and the $x$ and $y$ image coordinates of the center of the largest orange blob in the current image, and the joint perception, the current pan and tilt of the robot's head. The actuation space consists of the pan and tilt velocities of the head. Values for the variables controlling the priors were selected by hand. They were not

optimized for the particular task but were selected to match the scale and dimensionality of the data.

Fig. 4 shows the segmentation of the demonstration data as compared to the actual state of the hand-coded state machine. The hand-coded controller has three objectives: *seek right*, *seek left*, and *track ball*. The approach proposed in this paper segments the data into four objectives roughly corresponding to *seek left and track*, *seek right*, *look up*, and *look down*.

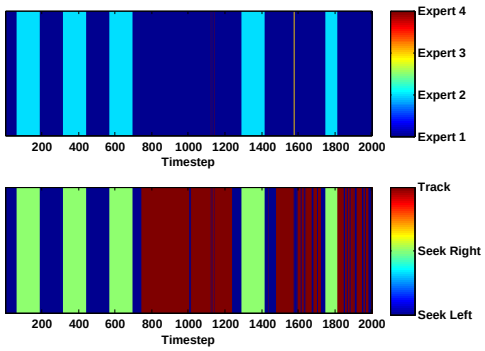


Fig. 4. (Top) The experts assigned by the algorithm for the first two thousand datapoints in the demonstration. (Bottom) The state of the hand-coded controller at each datapoint.

The learned controller performed well at both scanning and tracking on an actual robot. Fig. 5 visualizes the learned controller when the robot does not see the ball. The first two learned experts contain the seeking right and seeking left objectives. The transition between these two objectives differs from the ground truth transitions by less than .01 radians on the left and .03 radians on the right. Fig. 6 shows a comparison of the actuation executed by the learned controller during timesteps when the robot could see the ball with the actuation which would have been selected by the hand-coded controller. The learned controller selects similar actions to the hand-coded with a root mean squared error (RMSE) of less than 0.0032 and 0.0024 radians per second for the pan and tilt axes respectively.

We performed a comparison against the work of Grollman [10] and found that it segmented the demonstration data into six equally valid objectives with single-valued policies. However, to perform

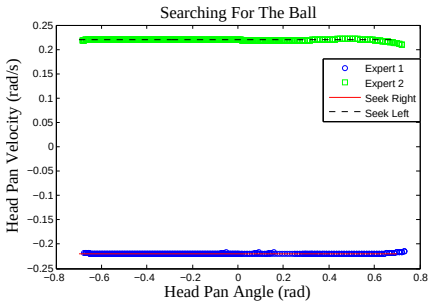


Fig. 5. Perceptual aliasing occurs when the ball is not found in the image. This graph shows the data collected while the robot was under autonomous control (blue and green datapoints) and the ground truth policies of the hand-coded controller (red and black lines). The autonomous function matches the ground truth very closely, with the robot panning slightly too far right and not quite far enough left compared to the hand-coded controller.

the task, Grollman's algorithm would require using a hand coded mechanism for selecting the correct expert at each timestep. A single-valued function regressor (SOGP) was able to learn to track the ball when it was on screen, though it was significantly more prone to error with an RMSE 40 times higher than our method. The higher rate of error suggests perceptual aliasing impedes the accuracy of the function regressor even in areas where the function is single-valued. SOGP also failed to learn the scanning behavior due to the perceptual aliasing. See the video available at `http://www.youtube.com/watch?v=XEPDMxbq-Vk` which shows our learned controller and the controller learned by SOGP.

### *B. Localize Reach*

In our second experiment, the robot learned to look for, reach towards, and touch a red cylinder on a table. This task was taught to the robot from human demonstration, with two human operators controlling the Nao. The arm of the learning robot is controlled by wirelessly mirroring the arm of a second robot which is being directly manipulated by an operator. The Naos head is controlled by a second operator using a wiimote. The perceptual space has eleven dimensions consisting of head pan and tilt, the two coordinates blob center, the blob area, and the six joint angles of the arm. The

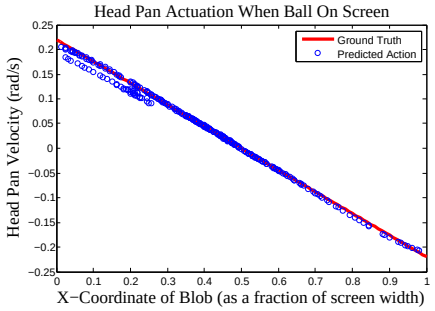


Fig. 6. Head pan actuation and blob perception data collected while the autonomous controller was running and the ball was on screen (blue datapoints) are compared to the hand-coded control policy (red line). Errors in the upper left correspond to an area where the head tilt values were further from those in the demonstration data, causing the zero mean prior to have a larger influence.

action space has eight dimensions consisting of the velocities of the six joints in the arm, and the pan and tilt velocities for the head.

In this experiment, the Nao's task was to touch the red cylinder and then to return to the starting position. This task also contains perceptual aliasing where the robot moved its arm forward and back along similar paths. The perceptual information alone is not sufficient to determine if the robot should move toward the cylinder or back toward the final position. The operators gave the robot approximately three minutes of training.

Inference produced two experts which roughly corresponded to the two objectives in the task. The video at `http://www.youtube.com/watch?v=XEPDMxbq-Vk` shows the teleoperation system and an example learned controller completing the localize reach task. Of seven trials, the robot correctly touched the cylinder in six, and reached the final position with its fist raised in five.

## V. Conclusions

Robot learning from demonstration is a promising approach to enable autonomous robot control policies to be created without programming. This paper focuses on extending LfD techniques to simultaneously learn both low level primitive tasks in a continuous space as well as the transitions between the primitive tasks when faced with perceptual aliasing. Our initial results demonstrate that the proposed model is viable when perceptual aliasing occurs within demonstration data.

## References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, 2009.

[2] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical dirichlet processes," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1566–1581, 2006.

[3] W. D. Smart and L. P. Kaelbling, "Effective reinforcement learning for mobile robots," in *International Conference on Robotics and Automation*, (Washington, D.C.), pp. 3404–3410, May 2002.

[4] J. Kober, B. Mohler, and J. Peters, "Learning perceptual coupling for motor primitives," in *International Conference on Intelligent Robots and Systems*, (Nice, France), Sept. 2008.

[5] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Neural Information Processing Systems*, (Whistler, CAN), pp. 1–8, Dec. 2006.

[6] C. Atkeson and S. Schaal, "Robot learning from demonstration," in *International Conference on Machine Learning*, (Nashville, TN), pp. 12–20, July 1997.

[7] D. C. Bentivegna, C. G. Atkeson, and G. Cheng, "Learning tasks from observation and practice," *Robotics and Autonomous Systems*, vol. 47, pp. 163–169, June 2004.

[8] D. H. Grollman and O. C. Jenkins, "Sparse incremental learning for interactive robot control policy estimation," in *International Conference on Robotics and Automation*, (Pasadena, CA, USA), pp. 3315–3320, May 2008.

[9] S. Chernova and M. Veloso, "Learning equivalent action choices from demonstration," in *International Conference on Intelligent Robots and Systems*, (Nice, France), Sept. 2008.

[10] D. Grollman, *Teaching old dogs new tricks: Robot Learning from Demonstration with Multimap Regression*. PhD thesis, Brown University, 2009.

[11] F. Wood, D. H. Grollman, K. A. Heller, O. C. Jenkins, and M. Black, "Incremental nonparametric bayesian regression," Tech. Rep. CS-08-07, Brown University Department of Computer Science, 2008.

[12] H. H. Bui, S. Venkatesh, and G. West, "Policy recognition in the abstract hidden markov model," *Journal of Artificial Intelligence Research*, vol. 17, p. 2002, 2002.

[13] L. Liao, D. Fox, and H. Kautz, "Learning and inferring transportation routines," in *AAAI*, 2004.

[14] A. Ihler, "Kernel density estimation toolbox for matlab," 2010.

[15] A. Gray and A. Moore, "Very fast multivariate kernel density estimation via computational geometry," in *Joint Stat. Meeting*, 2003.

[16] L. Csató and M. Opper, "Sparse online gaussian processes," *Neural Computation*, vol. 14, no. 3, pp. 641–669, 2002.