Sparse Incremental Learning for Interactive Robot Control Policy Estimation

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

***Abstract*— We are interested in transferring control policies for arbitrary tasks from a human to a robot. Using interactive demonstration via teloperation as our transfer scenario, we cast learning as statistical regression over sensor-actuator data pairs. Our desire for interactive learning necessitates algorithms that are incremental and realtime. We examine Locally Weighted Projection Regression, a popular robotic learning algorithm, and Sparse Online Gaussian Processes in this domain on one synthetic and several robot-generated data sets. We evaluate each algorithm in terms of function approximation, learned task performance, and scalability to large data sets.**

I. Introduction and Related Work

In this paper we address the problem of **Policy transfer**, how a control policy (π) for some unknown task, latent in the mind of a human user, can be transitioned onto a robot. The robot's resulting policy, $\hat{\pi}$, should match the user's decision making as closely as possible. Commonly, policy transfer is effected via programming: the user, perhaps with the help of an expert coder, explicitly writes a program that controls the robot to accomplish the task. A different approach would be to use **Learning from Demonstration** (LfD) [1,2]. Sometimes called programming by demonstration, LfD can enable nontechnical users to teach robots to perform arbitrary tasks. By using interaction and feedback, users can implicitly train and evaluate the robot in realtime without needing to write any code [3,4].

Reinforcement Learning (RL) can be applied to LfD and has been used to teach mobile robots navigation and obstacle avoidance tasks, among others [5]. In RL, users provide rewards when a task is successfully completed and a robot's policy is to choose actions that maximize expected reward. One can, instead, update the policy directly, as in Policy Gradient approaches [6]. Both of these techniques operate under the assumption that rewards arrive as a task is being completed. However, it is possible that a demonstrator can give information as to what the robot should be doing at every point during the task. Kinesthetically [7], or via teleoperation [8], a robot can be physically guided through the task, generating many pairs of matched sensor-actuator data. Learning the task then becomes an issue of learning the mapping from sensors to actuators, which may be performed with supervised **statistical regression**.

Regression in interactive teaching by demonstration scenarios puts severe constraints on the learning algorithm, demanding that it be able to update a learned policy as new data arrives in realtime. Locally Weighted Projection Regression (LWPR) [9] is a popular machine learning algorithm that attempts to meet these requirements, seeking to provide incremental, realtime inference and prediction for high-dimensional input-output function approximation. Since its introduction, it has been used in a variety of robot learning tasks such as operational space control of a 7DOF arm [10] and switching between multiple dynamical models [11].

LWPR was originally compared with Gaussian Process Regression (GPR) [12], a statistical learning technique that requires batch processing, $O(N^2)$ memory, and $O(N^3)$ time. These high costs made it unacceptable for long-term realtime robot learning, even though GPR performs excellently on small data sets and has been used successfully to teach simple motor acts to a humanoid robot [13]. Many approaches in Gaussian Process approximation [14] have addressed the memory and runtime issues, and an incremental formulation [15] allows data to be processed as it arrives. Thus GPR is now poised to compete with LWPR, for they both provide incremental, sparse function approximation. A remaining fundamental difference between the two is that LWPR fits its approximation locally, while GPR works globally.

Local fitting is advantageous in that one set of parameters need not be used over the entire input space of the problem. Different regions can have different parameters, allowing a collection of simpler models, or experts, to approximate a complex mapping [16]. A major issue with this approach is that of **model selection**, determining the number and locations of the experts required to accurately capture the target function. Infinite mixture models avoid making a hard decision and instead allow the number of experts to grow in a nonparametric manner. Infinite Mixture of Gaussian Process Experts [17] may combine the regressive strength of GPR with the local model formulation of LWPR.

In this paper, we compare a sparse incremental approximation to GPR with LWPR in the domain of interactive robot learning from human demonstration. We first test both algorithms as function approximators and task learners and then examine them with respect to speed of training and ability to learn from noisy human-generated data. Preliminary experiments with non-functional tasks lead us to consider combining aspects of both algorithms in our future work.

II. Learning Methodology

We seek to learn unknown tasks from interactive demonstration, and thus cast robot control policy estimation as a form of supervised regression. That is, our robot performs a mapping, $\hat{\pi}(\mathbf{s}) \rightarrow \mathbf{a}$, from perceived state to desired action and the goal of learning is to make $\hat{\pi}$ match π, the control policy latent in the demonstrator from observations are of the

Algorithm 1 LWPR

Training: Start with no receptive fields ($K = 0$)
for all $(\mathbf{x}_i, \mathbf{y}_i) \in (\mathbf{X}, \mathbf{Y})$ **do**
  for all RFs, $k = 1 : K$ **do**
    Calculate activation via (1)
    Update the RF model parameters, θ_k
  end for
  if no RF was activated by more than $w_{\texttt{gen}}$ **then**
    create a new RF, $\mathbf{c}_k = \mathbf{x}_i$, $\mathbf{D}_k$ = $\mathbf{D}^*$, K++
  end if
end for

Prediction on query point x'
for all RFs, $k = 1$ to K **do**
  calculate output $\mathbf{y}_k$ and activation w_k
end for
return $\hat{\mathbf{y}}$ via (2)
Confidence bounds (σ^2) are given by (3)

Algorithm 2 SOGP

Training: Start with empty parameters and basis set
for all $(\mathbf{x}_i, \mathbf{y}_i) \in (\mathbf{X}, \mathbf{Y})$ **do**
  calculate novelty (γ) according to (5)
  if $\mathbf{x}_i$ is sufficiently novel ($\gamma > \epsilon_{\texttt{tol}}$) **then**
    perform full update, increasing size of BV by 1
  else
    perform a sparse update, BV set remains same size
  end if
  if size$(BV) > P$ **then**
    find point with lowest score (ϵ_i) and remove it
  end if
  Remove any points whose novelty is now below $\epsilon_{\texttt{tol}}$
end for

Prediction on query point $\mathbf{x}'$
compute k'_i by (4) for all $\mathbf{b}_i \in BV$
return $\hat{\mathbf{y}}$, σ^2 via (6) and (7)

form $(\mathbf{s}, \mathbf{a})$ gathered as the robot is teleoperated to perform the task. As we are concerned with interactive teaching we look at incremental approaches, where $\hat{\pi}$ is updated after each pair is observed. In addition, we are interested in techniques that scale well to large, high-dimensional data sets. Lastly, we seek a general algorithm, capable of learning many different types of functions, and thus look at nonparametric approaches.

In keeping with the statistical literature, we denote the multivariate inputs (perceived states) $\mathbf{X} = \{\mathbf{x_i}\}_{i=1}^N$, and the outputs (desired actions) $\mathbf{Y} = \{y_i\}_{i=1}^N$.

A. LWPR

Locally Weighted Projection Regression (LWPR) is an incremental algorithm that performs piecewise linear function approximation using the concept of receptive fields. The input space is divided into a set of K (possibly overlapping) fields, defined by center point ($\mathbf{c}_k$) and a Gaussian area of influence parameterized by $\mathbf{D}_k$. Each Receptive Field (RF) fits a set of univariate regressions (θ_k) to the data within it. These regressors are chosen efficiently using Partial Least Squares Regression (PLS), thus dealing with redundant and extraneous dimensions. Receptive fields are created as needed based on a tunable threshold value, $w_{\texttt{gen}}$.

During training, all RFs calculate their *activation*, or weight, measuring how close the new input is to $\mathbf{c}_k$.

$$w_k = \exp(-\frac{1}{2}(\mathbf{x} - \mathbf{c_k})^\top \mathbf{D}_k(\mathbf{x} - \mathbf{c}_k)) \tag{1}$$

Each RF then performs (weighted) incremental PLS to update θ_k, possibly increasing the number of regressors, and leave-one-out cross-validation to adjust $\mathbf{D}_k$, which is set initially to a default value, $\mathbf{D}^*$. All update equations are local to the model and only require sufficient statistics to be kept.

For prediction, each RF produces an estimate of the output and outputs from all models are combined according to their activation weights. That is, for a query point $\mathbf{x}'$,

$$\hat{\mathbf{y}} = \frac{\sum_{k=1}^K w_k * \mathbf{y}_k}{\sum_{k=1}^K w_k} \tag{2}$$

where y_k is the output from the kth RF. An alternative formulation simply returns $\hat{\mathbf{y}} = \mathbf{y}_k, k = \text{argmax}(w_k)$. This formulation prevents RFs from blending with each other, which may or may not be desired.

LWPR also provides confidence bounds on the predicted outputs. The standard deviation is given by:

$$\sigma^2 = \frac{1}{(\sum_k w_k)^2} \sum_{k=1}^K w_k[(\hat{\mathbf{y}} - \hat{\mathbf{y}}_k)^2 + f(\mathbf{x}', \theta_k)] \tag{3}$$

The first term in the sum compares each RF's output with the global output and the second term measures the projection error of the query point onto the local projetion directions.

In addition to being local and incremental, LWPR is sparse, in that it does not need to remember all of the training data. Instead, once an input-output pair has been incorporated into a model (θ_k are updated), the data can be discarded. This feature makes it well suited for long-term robot learning, where memory constraints may be a factor.

An overview of the LWPR algorithm can be seen in Alg. 1, for full details we refer the reader to [9].

B. SOGP

Gaussion Process Regression (GPR) is a global nonparametric function approximation technique that derives a posterior distribution over functions in a data-driven manner. From a Gaussian Process (GP) Prior ($GP = p(\mathbf{f_x})$) distribution over functions, we determine the posterior $p(\mathbf{f_x}|\mathbf{X}, \mathbf{Y})$. As in LWPR, a Gaussian kernel with width σ_k^2 is used to compare points, although the formulation differs slightly:

$$k_{ij} = \exp\left(\frac{e||\mathbf{x}_i - \mathbf{x}_j||^2}{2d\sigma_k^2}\right) \tag{4}$$

where d is the dimensionality of the input space. Unlike LWPR, one kernel width is used over the entire input space, hence the global nature of GPR. In addition, there is a global noise parameter σ_0^2 that represents how noisy the function is in general. In LWPR, this is estimated from the data.

Incorporating a new datapoint is done by changing the output coefficients (α) and posterior kernel ($\mathbf{C}$) functions of the distribution. Algorithmically, this is accomplished by updating a Gram matrix (all pairs kernel distance), and storing the data for future comparison. Thus, GPR requires O(N^2) storage and is unsuitable for our needs. Sparse approximations to GPR store only a subset of the data and their associated kernel distances. Points stored are chosen to minimize the Kullback-Leiber (KL) divergence between the exact posterior distribution and the the approximation.

The P points, or basis vector set (BV), can be chosen incrementally by projecting each new point onto the space spanned by the points in the BV

$$\gamma = k^* - \mathbf{k}^\top \mathbf{Q} \mathbf{k} \tag{5}$$

where k^* is the kernel distance between a point and itself, $\mathbf{k}$ is the kernel distance between the new point and all points currently in the basis set, and $\mathbf{Q}$ is the inverted gram matrix of the basis set. γ then represents the magnitude of the residual vector after projection.

If the new point is sufficiently novel, it is added to the BV and the GP update proceeds as normal. If not, information from the new point is still incorporated into the posterior by updating α and $\mathbf{C}$, although $\mathbf{Q}$ does not change. Sufficiency is defined to prevent numerical instabilities as a cutoff value $\epsilon_{\texttt{tol}}$. Points are removed from the BV by calculating their leave-one-out error, or 'score', $\epsilon_i = \alpha_\mathbf{i}/(\mathbf{C}_{ii} + \mathbf{Q}_{ii})$. The point with the lowest score, that introduces the least error by being removed, is selected for removal.

Prediction of output for a query point x' is given by:

$$\hat{\mathbf{y}} = \mathbf{k}^\top \alpha \tag{6}$$

where k is as defined above. Confidence bounds are likewise:

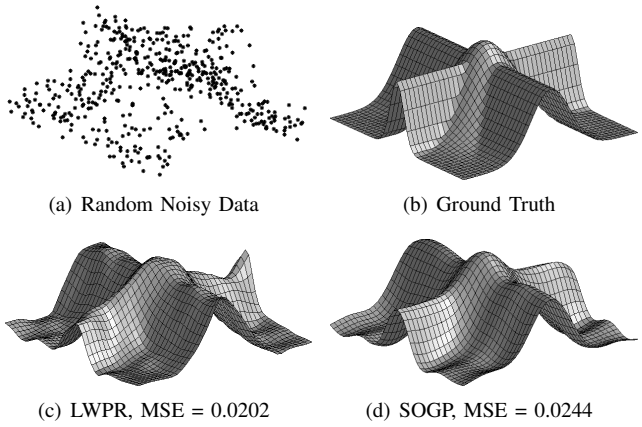
$$\sigma^2 = \sigma_0^2 + k^* + \mathbf{k}^\top \mathbf{C} \mathbf{k} \tag{7}$$

The size of the basis set, P, and thus the maximum capacity of the GP, can be set to limit the SOGP to O(P^2) memory and O(P^2) per data point training time (O(NP^2) overall). Prediction is likewise limited to O(P^2). P can thus be chosen to ensure realtime operation.

An overview of the SOGP algorithm can be seen in Alg. 2, and we refer the reader to [15] for extensive details.

C. Comparison

LWPR and SOGP perform comparably well on small data sets. In the case where $N < P$, SOGP is equivalent to a robust form GPR, protected against numerical instabilities. Key differences are that LWPR, by virtue of its local PLS models, can detect the intrinsic dimensionality of each region and adapt the local expert to match. SOGP, on the other hand, has one set of global parameters that are used over the entire input space. However, SOGP provides tighter control of the storage used by setting P. LWPR allows control over RF generation with $w_{\texttt{gen}}$ and similar variables enable RFs to be pruned, but it is more difficult to set a hard limit on total storage. Instead, it tends to grow as $O(N)$. Both techniques provide statistically sound notions of 'confidence' as a measure of how good the predictive output is.



(a) Random Noisy Data (b) Ground Truth

(c) LWPR, MSE = 0.0202 (d) SOGP, MSE = 0.0244

Fig. 1. SOGP and LWPR with default parameters compared on the cross function. We limit SOGP's capacity to the number of RFs used by LWPR (22). Both algorithms have low MSE, which can be further decreased by parameter tuning. If SOGP capacity is infinite, $\texttt{MSE}_{\texttt{SOGP}} = 0.0056$. The local model nature of LWPR is visible as 'lumpiness'.

Both LWPR and SOGP can be seen as addressing the model selection problem. In LWPR, this is explicit as receptive fields and local models are created. The number and location of the RFs, and their parameters, greatly influence the resulting approximation. SOGP, on the other hand, does not vary parameters across models, nor can it grow the number of models greater than P. However, the basis set implicitly defines a set of model centers.

To initially compare expressive capability, we ran both SOGP and LWPR on the 'cross' dataset seen in Fig. 1. This dataset was used to demonstrate LWPR's functionality in [9] and exhibits a mapping that is difficult to capture with parametric models. 500 randomly distributed points are used to train each algorithm and a regularly spaced grid is used to test. Out-of-the-box LWPR, with the default parameters of $\mathbf{D}^* = 25, w_{\texttt{gen}} = .2$, produces 22 RFs and $\texttt{MSE}_{\texttt{LWPR}} = 0.0202$. Using 22 as the capacity of SOGP, and $\sigma_0^2 = .1$, $\sigma_k^2 = .1$, we obtain $\texttt{MSE}_{\texttt{SOGP}} = 0.0244$. Visually examining the plots, the local nature of LWPR becomes visible as it appears 'lumpy'. By increasing $\mathbf{D}^*$ to 100, we can decrease MSE by an order of magnitude, to $\texttt{MSE}_{\texttt{LWPR}} = 0.0067$ and $\texttt{MSE}_{\texttt{SOGP}} = 0.0073$, but at the cost of more models, $K = 72$. It is important to note that we only process one pass, or 'epoch', over the training data in keeping with our realtime robot learning formulation.

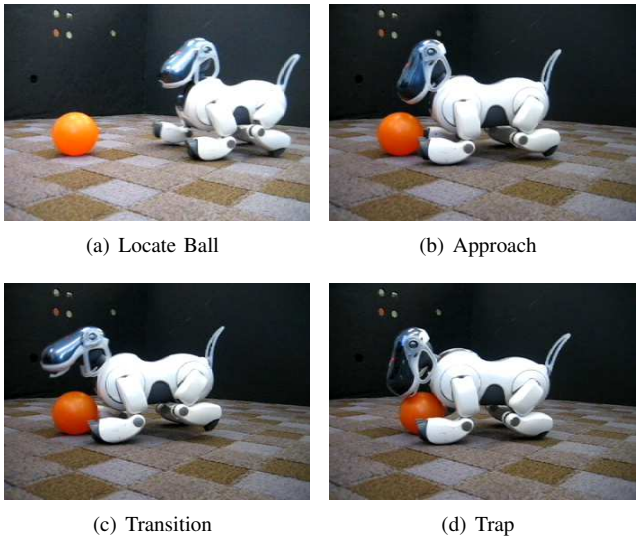
III. Experimental Results

We examine both LWPR and SOGP as applied to an interactive robot control-policy estimation problem which requires an incremental, realtime learning algorithm. Our interactive training paradigm, discussed in [18], is used to

TABLE I

THE TASKS USED IN OUR EXPERIMENTS, THEIR INPUT/OUTPUT DIMENSIONALITIES AND DATA SET SIZES.

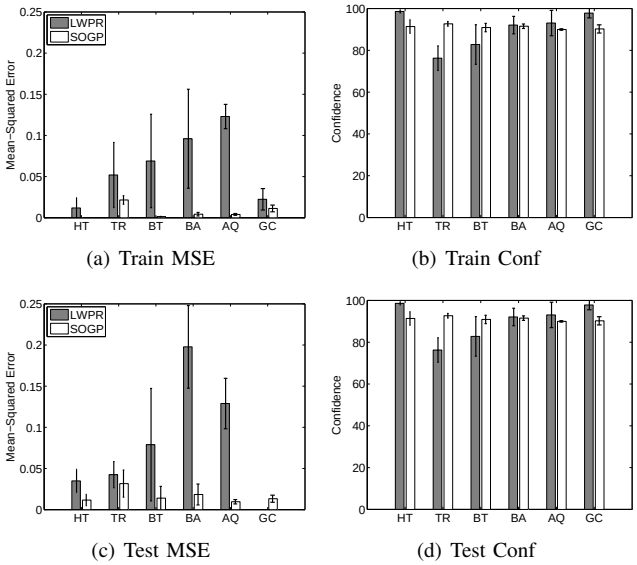
Task	d_{in}	d_{out}	Data Sets				
HT: Head-Tail test task, Robot's head mirrors its tail	8	10	494	629	525	825	981
TR: Trap, Robot captures ball under chin and signals success	3	4	236	185	517	536	373
BT: Ball-Track, Robot's head moves to keep ball centered in view	20	2	526	757	718	723	1684
BA: Ball-Approach, Robot walks to ball and stops	21	6	805	747	829	967	938
AQ: Ball-Acquire, A composition of BA and TR	22	7	2037	1329	1458	929	762
GC: Goal-Charge, Robot locates and walks into the yellow goal	30	23	882	826	958	1309	944



(a) Locate Ball (b) Approach

(c) Transition (d) Trap

Fig. 2. Our robot and its ball, performing the AQ task described below.



(a) Train MSE (b) Train Conf

(c) Test MSE (d) Test Conf

Fig. 3. MSE and confidence of each algorithm on the training data sets and on leave-one-out testing. All results are averaged over 5 folds and 1 standard deviation error bars are shown. LWPR tests on the GC task ran longer than the allocated time and are thus not reported.

attempt to teach a Sony Aibo robot dog (Fig. 2) a variety of tasks related to robot soccer. Briefly, the robot is equipped with rudimentary color-based vision capable of picking out relevant objects (ball, goal, etc) and a walking behavior suitable for locomotion. The system does *not*, however, have knowledge of the desired tasks. The tasks examined here, their input-output dimensionalities, and the sizes of the data sets described below can be seen in Table I.

Of special interest is the AQ task, which involves locating and approaching the ball and then transitioning into a trap behavior. Initial attempts to learn this task failed due to the ambiguous state that occurs at transition time. While in the midst of transitioning (Fig. 2(c)), there is not enough data available to the robot to determine which subtask it should be performing. To solve this issue, we had to introduce a new variable into the environment, one that represents the robot's internal state. That is, the robot generates an output indicative of the current subtask being performed, and this output is read back in at the next time step, enabling the learning algorithm to make decisions based upon it.

In all of our experiments the inputs and outputs are normalized to be in the range [-1,1]. We use fixed parameters for both algorithms across all tests. For LWPR, we use $w_{\text{gen}} = .2$, an initial setting of $\mathbf{D}^* = 100$ and enable blending. For SOGP, we set $\sigma_k^2 = .1, P = 500, \sigma_0^2 = .1$. These values were chosen to give comparable performance on the cross data and to enable learning on the HT test task. We used the C++ LWPR library available from [19] and wrote our own SOGP library based on [15].

A. Mean Squared error

With the exception of the GC task which was human demonstrated and will be discussed in detail later, each task is demonstrated by a hand-coded controller that provides a functional mapping from inputs to outputs. We use coded controllers to allow for repeatable, consistent data generation without issues of human error and noise. Each behavior was run, without learning, 5 times, generating matched input-output pairs for analysis. For the experiments in this section, the interactive training ability of our system was not used.

We trained and tested each algorithm on each data set. Figure 3(a) shows the average mean squared error from each algorithm being tested on the same data it was trained on. In addition, the mean confidence level, scaled to the range [0-100], is shown in Fig. 3(b). As can be seen, SOGP consistently has lower MSE and standard deviation than LWPR, although both have high confidence. This may be a sign of SOGP overfitting the training data.

Generalizing to new data is key for robot learning. We therefore train each algorithm on 4 of the data sets from each task and test on the 5th. Figure 3(c) shows the resulting MSE and 3(d) the confidences, averaged over all 5 folds. Again,

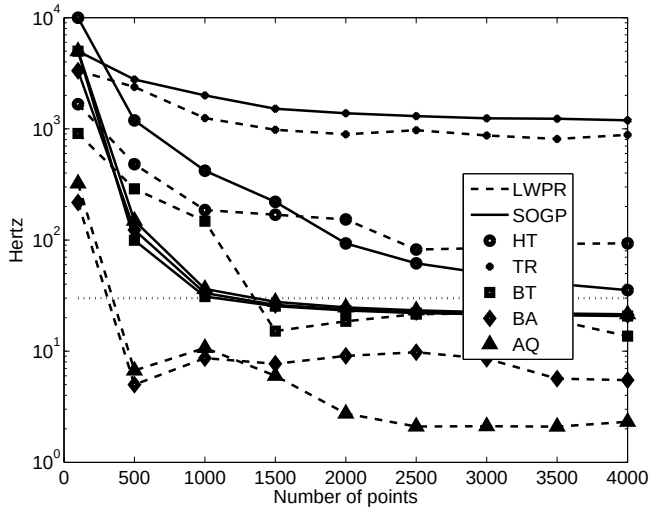


Fig. 4. The speed of the algorithms as a function of the amount of data trained on. Dotted line represents 30Hz, or realtime performance.

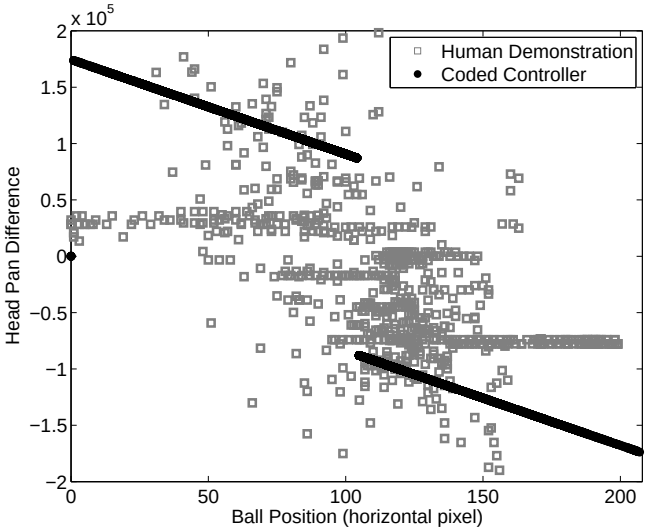


Fig. 5. The user generated data compared with that from a hand-coded controller. The user's data is noisy and exhibits non-functionalities. However, we are able to learn the task from it.

SOGP has lower MSE and standard deviation over all tasks. It is possible that LWPR's MSE could be bought down to the level of SOGP, or even lower, by changing the parameters. However, as seen in the next section, these error rates are sufficiently low for task learning in most cases.

B. Task performance

Mean squared error is a useful metric for evaluating these algorithms as function approximators, but we are more concerned with their abilities as task learners. That is, does the learned function allow a robot to track a ball when trained on BT data, trap the ball from TR data, etc. We evaluate algorithm performance in this way by using the learned function to drive a robot in real time. Each algorithm, trained on all 5 datasets for each task drives the robot and the task is evaluated by a human expert. The expert rates the system in a task specific manner from 0 (complete failure, i.e. robot doesn't move in response to stimuli) to 10 (task performed with no errors). Speed of performance is examined separately. Both algorithms were able to achieve at least a decent performance (5) on most tasks. A notable exception is when LWPR was applied to the AQ and GC tasks. For AQ, the learned policy did not transition from the approach to trap behavior. For GC, the robot simply exhibited random behavior, although some goal-directed movements were seen. Full results are shown in Table II.

TABLE II

THE RESULTS OF USING THE LEARNED FUNCTION APPROXIMATION TO DRIVE THE ROBOT TO PERFORM EACH TASK. SCORES ARE FROM 0 (TOTAL FAILURE) TO 10 (PERFECT TASK EXECUTION).

Task	HT	BT	BA	TR	AQ	GC
LWPR	8	4	6	10	3	3
SOGP	6	8	9	10	6	9

C. Speed

In addition to error in mapping and task performance, we are concerned with the speed of learning. Realtime interactive instruction requires that new data must be incorporated as fast as it arrives and outputs must be produced at the same rate. On our system, this corresponds to $\sim$30Hz.

We generated 4000 new data points for each task. Fig. 4 shows speed (average data points processed per second) versus N, the number of data points processed. It can be seen that speed depends on the dimensionality of the data as well. Both algorithms appear to approach an asymptotic limit, although LWPR's path is more erratic, suggesting data dependency. As new, sufficiently different data arrives we expect LWPR to slow down further, as new RFs are created. SOGP, on the other hand, ceases to slow down once its capacity is reached. We believe that by decreasing P, the speed of SOGP can be kept $>$ 30Hz indefinitely.

D. Interactive Training

In addition to non-interactive data gathering, we used human demonstration to train the GC task interactively. Human demonstrators teleoperated the robot to locate and walk into the goal. During training, the user could stop teaching and observe the robot's behavior, and resume generating output at will. Due to the lack of a full user study, this experiment provides more anecdotal results.

Results with SOGP were very good in terms of task performance and ease of training. In a small amount of time (5-10 seconds), users were able to teach the robot to perform the GC task. In contrast, coding a controller to perform this task takes on the order of minutes for an expert in the system. Of particular interest is that the interactive nature of training allowed the user to recognize when the task had been sufficiently learned, and stop demonstrating.

LWPR was unable to learn this task from initial demonstrations, although this may be due to poor parameters. However, as more data was generated by the user to further teach the task, the increasing processing time per point became a problem. The entire system slowed down and interaction became impractical. This experience agrees with the results of our speed analysis.

We posited that the noisy and ambiguous nature of human demonstration may have been causing additional problems. We thus compare human and controller-generated data for the same task (BT). As seen in Fig. 5, human generated data is very noisy, and does not, at first glance, appear to represent the correct function. However, SOGP successfully learned the BT task from this human demonstration as well.

IV. Conclusion and Future Work

Policy transfer, as an area of research, will be increasingly important in the future of Human-Robot Interaction as nontechnical users begin to modify consumer robots to fit their needs. Such adaptable robots will require algorithms that meet the standards discussed here, namely realtime performance, incremental updates, and performance in resource constrained systems. After examining LWPR and SOGP in the domain of robot control policy estimation for unknown tasks from human demonstration, we conclude that for the tasks examined here, both algorithms provide good function approximation and are adequate to learn most of them. However, the hard memory and timing guarantees of SOGP make it more suitable for realtime interaction.

Two related issues that we plan to research in the future are those of model selection and internal state.

A. Model Selection

Both LWPR and SOGP perform a type of model selection to achieve sparseness in memory and thus speed up processing to realtime speeds. LWPR's receptive fields are the more versatile of the two, as they can change shape and parameters to fit data locally. SOGP's basis vectors instead all share the same parameters. Despite this drawback, SOGP performs better on our tasks in terms of both function approximation and task performance. This may be due to the fact that LWPR's RFs are immobile once created, while SOGP's basis set can change entirely.

In addition, both algorithms are nonparametric, in the sense that the number of models is not set a priori. Each also affords controls governing the creation/deletion of models. Modifications to LWPR that enable it to mimic SOGP's hard limit on memory usage may enable it to maintain realtime performance in the face of large amounts of data.

B. Internal State

First attempts to learn the AQ task with both algorithms failed. Believing that ambiguous outputs generated during transitioning were at fault, we exposed the internal state of the controller to the learning algorithm and achieved the results presented here. This approach is not generally applicable, especially to noisy human-generated data.

Ambiguous outputs can be dealt with by models that overlap in input space and produce distinct outputs, one of which is selected as the global answer. This is exactly the non-blending option of LWPR. We would like to merge this ability with the noisy learning capabilities of SOGP.

An Infinite Mixture of Gaussian Processes model has been presented that deals well with both ambiguous outputs and varying parameters across the input space [17]. This technique, modified to be incremental and sparse, may be what we require to combine the function approximation and memory guarantees of SOGP with the local model formulation of LWPR.

References

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

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

[3] T. Inamura, M. Inaba, and H. Inoue, "Acquisition of probabilistic behavior decision model based on the interactive teaching method," in *9th Intl. Conf. on Advanced Robotics (ICAR)*, 1999, pp. 523–528.

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

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

[6] R. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems 12 (NIPS)*, 2000.

[7] S. Calinon and A. Billard, "Incremental learning of gestures by imitation in a humanoid robot," in *2nd Conf. on Human-Robot Interaction (HRI)*, Arlington, Virginia, 2007, pp. 255–262.

[8] S. Chernova and M. Veloso, "Confidence-based policy learning from demonstration using gaussian mixture models," in *Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2007.

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

[10] J. Peters and S. Schaal, "Reinforcement learning for operational space control," in *2007 IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 2111–2116.

[11] G. Petkos and S. Vijayakumar, "Context estimation and learning control through latent variable extracion: From discrete to continuous contexts," in *2007 IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 2117–2123.

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

[13] A. P. Shon, J. J. Storz, and R. P. Rao, "Towards a real-time bayesian imitation system for a humanoid robot," in *2007 IEEE Intl. Conference on Robotics and Automation (ICRA)*, 2007, pp. 2847–2852.

[14] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[15] L. Csató, "Gaussian processes - iterative sparse approximations," Ph.D. dissertation, Aston University, Marck 2002.

[16] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, pp. 79–87, 1991.

[17] E. Meeds and S. Osindero, "An alternative infinite mixture of gaussian process experts," in *Advances in Neural Information Processing Systems (NIPS)*, 2006, pp. 883–890.

[18] D. H. Grollman and O. C. Jenkins, "Learning robot soccer skills from demonstration," in *6th IEEE International Conference on Development and Learning*, 2007.

[19] http://homepages.inf.ed.ac.uk/svijayak/software/LWPR/