Incremental Learning of Subtasks from Unsegmented Demonstration

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

***Abstract*— We propose to incrementally learn the segmentation of a demonstrated task into subtasks and the individual subtask policies themselves simultaneously. Previous robot learning from demonstration techniques have either learned the individual subtasks in isolation, combined known subtasks, or used knowledge of the overall task structure to perform segmentation. Our infinite mixture of experts approach instead automatically infers an appropriate partitioning (number of subtasks and assignment of data points to each one) directly from the data. We illustrate the applicability of our technique by learning a suitable set of subtasks from the demonstration of a finite-state machine robot soccer goal scorer.**

I. Introduction

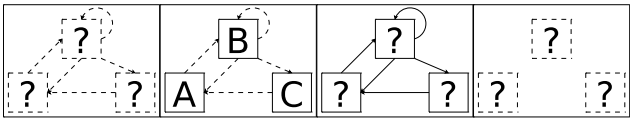
We consider Robot Learning from Demonstration (RLfD) as seeking to enable users to instantiate desired autonomous control policies onto robots without requiring explicit coding or procedural analysis of the task itself. The idea is that the user demonstrates the task and the robot collects data which when processed results in a control policy that can operate the robot autonomously in novel situations. A particularly compelling branch of RLfD is interactive-RLfD, where learning occurs as the demonstration is performed, enabling active learning and rapid appraisal of the learned skills [1].

There have been many techniques advanced for performing RLfD (see [2] for an overview), based both on ideas from Reinforcement Learning (RL) [3] and Direct Policy Approximation (DPA) [4]. Both classes of techniques seek to infer a control policy that maps the robot's state into actions. A main difference between the two is that RL considers a reward function underlying the policy, while DPA does not.

Both approaches have achieved success when the policy mapping is one- or many-to-one, when the robot's state uniquely determines the correct action to perform [5], [6]. However, when this is not the case, and the underlying policy is one-to-many (a multimap or multiple-valued function[1]), such techniques are unsuitable. Such policies are known to occur in Finite-State-Machine (FSM) controllers [7].

To be able to deal with FSM controllers, current RLfD techniques require additional information to disambiguate the outputs. Learning FSMs is thus related to the problem of Perceptual Aliasing (PA) [8], where two states that *should* lead to different actions generate the same perception. One solution is to augment the robot's state with the required additional information, by perhaps including a direct temporal dependency, incorporating history, operating in a belief space, or including higher-level features. Another technique is to first segment the task into subtasks (the FSM states), each of which is a unimap and can be learned in isolation. The overall task can then be performed by transitioning between these subtasks in an appropriate fashion.

[1]Consider $\sqrt{x}$ as a common multimap, where $\sqrt{4} \rightarrow 2, -2$. Unimap regression techniques average possible results and return 0.



(a) Full FSM (b) Transitions (c) Policies (d) Proposed

Fig. 1: FSM learning (a) requires us to discover the number of subtasks, their policies, and the transitions between them. Current techniques either use known subtasks and learn the transitions between them (b), or use a known segmentation and learn the individual policies (c). We propose to learn the segmentation and subtask policies simultaneously (d).

We propose to instead perform segmentation and subtask learning simultaneously using only the demonstration data, without any additional analysis on the part of the demonstrator. To be applicable to the interactive learning paradigm, our technique is incremental, updating the learned policy after each observed data point, and provides controls for setting the runtime speed of policy inference. We illustrate here that our approach can discover appropriate subtasks by manually combining those subtasks to perform the overall task. For complete FSM learning, our technique will need to be extended to simultaneously infer the transitions.

II. Background and related work

Our view is that in order to learn an FSM, we must learn both the necessary subtasks (including their number and individual policies), and the transitions between them, as illustrated in Figure 1a. As described in [7], this is not possible by approximating the full policy with standard techniques (either RL or DPA based), due to the fact that the underlying mapping is one-to-many. Instead, current approaches each address only part of the overall problem.

For example, given the transitions, the data can be segmented into subtasks and each subtask learnt in isolation (Figure 1c) [9]. Alternatively, if the individual subtasks are already known, transitions between them can be inferred and they can be sequenced properly (Figure 1b) [10].

The relation to PA arises when considering the identity of the currently-active subtask as a form of hidden state.

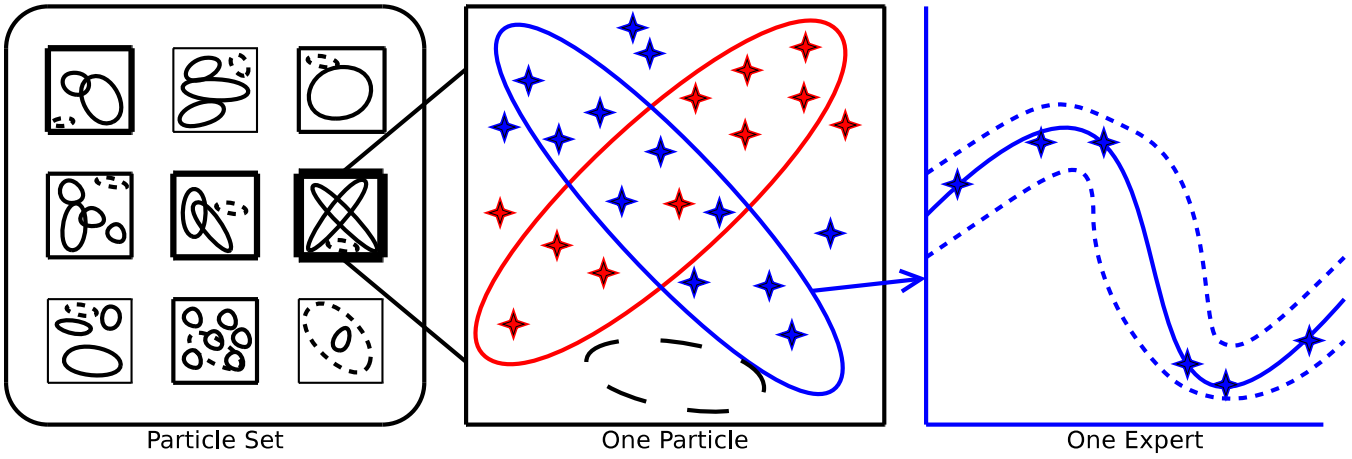


Fig. 2: ROGER, an infinite mixture of sparse Gaussian Process experts. A weighted particle set (left) approximates the distribution over possible partitions of the data. Each particle (center) contains a set of experts in input space (Gaussian Mixture Model) with assigned data (stars), and an empty expert (dashed). Each expert (right) is a nonparametric Sparse Online Gaussian Process regressor that maps from inputs to a Gaussian distribution over outputs using a small basis set.

Including this information in the robot perception effectively disambiguates which action should be performed in otherwise identical situations. The underlying policy is then a unimap (with respect to this extended state), and can be learned as usual.

One approach to inferring the existence of this hidden state is to examine the outputs associated with a particular perception and determine if they are *consistent* or not. The U-Tree algorithm [11] does this in the domain of discrete reinforcement learning by extending the belief space of an underlying POMDP. We act similarly here in the domain of continuous DPA by positing the existence of a new subtask.

Particularly, our technique always considers the possibility that an observed action has been generated from a previously unobserved subtask. By doing so we are able to grow the number of subtasks to fit the data, which is a form of *model selection*. Within each expert (hypothesized subtask) we utilize a Gaussian Process (GP) Regressor which approximates the distribution over all unimaps consistent with that expert's observed data. As the total number of experts is not fixed in advance, our model is an example of an Infinite Mixture of Gaussian Process Experts (IMoGPE) [12].

Previous work in this model has focused on batch inference techniques, where all training data is collected before learning. However, as we wish to operate in the interactive RLfD paradigm, we require an incremental variant. Further, we wish to be sparse to limit the amount of storage and computation required. Sparsity is achieved within each expert by minimizing the KL-divergence between the GP distribution based on all observed data, and one based on a smaller dataset. We use the sparse approximation (SOGP) of [13], because it has an incremental formulation.

We use our model, termed ROGER (Realtime Overlapping Gaussian Expert Regression), to estimate both the number of subtasks and their policies (Figure 1d) incrementally from unsegmented demonstration data. The overall learned policy is a multimap, where for a given perception multiple actions (corresponding to the different subtasks) can be generated. Proper action selection requires a transition matrix between the subtasks, which we do not learn here.

III. ROGER

An overview of the ROGER model is in Figure 2. Each expert is modeled as a Gaussian in robot perception (input) space, along with an SOGP regressor that approximates the subtask mapping from perception to action (Figure 2 right). Areas of overlapping expert regions (gates) correspond to multimaps, as outputs from each expert are applicable (Figure 2 middle). A single possible partitioning of the data into overlapping experts constitutes a particle, and we maintain a distribution over the possible partitioning of the data using a weighted set of particles (Figure 2 left). The partitioning itself is determined incrementally via the Chinese Restaurant Process (CRP) [14]. Essentially, as each new data point is considered, there is the possibility that it arises from (and is assigned to) a heretofore unseen subtask.

Mathematically, the generative model underlying ROGER is summarized in the following equations:

$$\begin{array}{lcl} z_i & \sim & \text{CRP}(\alpha) \\ \Sigma'_k & \sim & \text{Inverse-Wishart}_{\nu_0}(\Lambda_0) \\ \mu'_k & \sim & \text{Multivariate-normal}(\mu_0, \Sigma_k/\kappa_0) \\ \mathbf{x}_i|z_i & \sim & \text{Multivariate-normal}(\mu'_{z_i}, \Sigma'_{z_i}) \\ \mathbf{y}_k|\mathbf{X}_k, \theta & \sim & \text{Multivariate-normal}(0, \mathbf{Q}_k) \end{array} \quad (1)$$

Where $\mathbf{X}$ and $\mathbf{y}$ are the observed perceptions and actuations and $\mathbf{z}$ are a set of latent indicator variables that indicate which expert generated each datapair. μ_k and Σ_k are the parameters of the input Gaussian gates, which we analytically integrate over using the conjugate inverse-Wishart and multivariate normal priors to produce the joint:

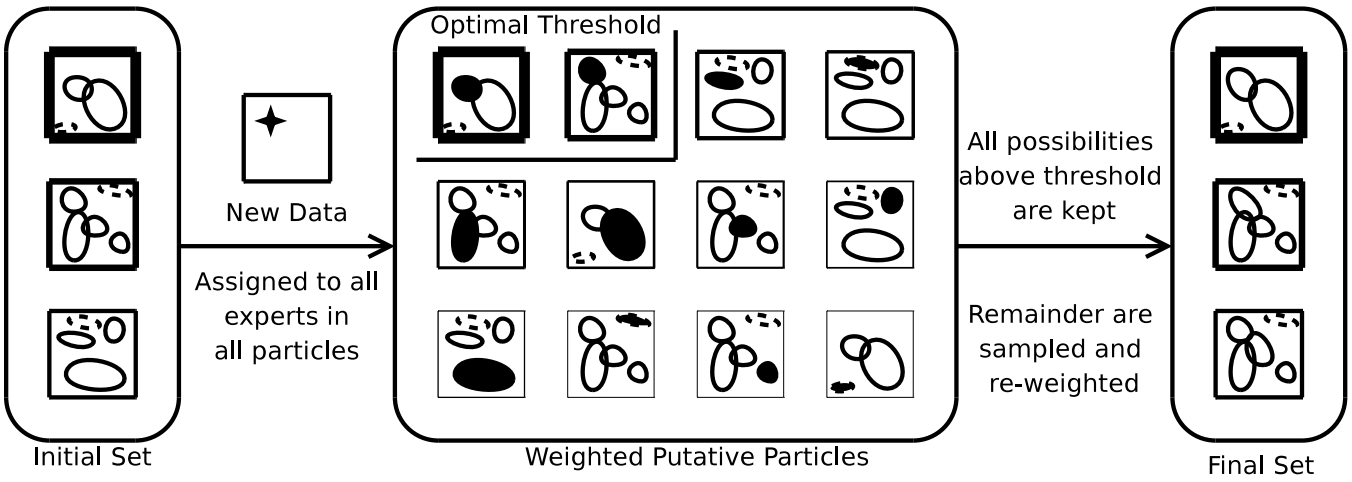


Fig. 3: Inference in ROGER. New data is provisionally assigned to all possible experts, including previously empty ones. Sampling from this set follows the optimal resampling technique of [15] and data is only incorporated after sampling.

$$P(\mathbf{X}, \mathbf{y}, \mathbf{z}; \Omega) = P(\mathbf{X}|\mathbf{z}; \Omega) P(\mathbf{z}|\Omega) \prod_{k=1}^{K} P(\mathbf{y}_k|\mathbf{X}_k, \Omega) \quad (2)$$

Here $\Omega = \{\alpha, \mu_0, \kappa_0, \Lambda_0, \nu_0, \theta\}$, is the collection of all parameters, and $(\mathbf{X}_k, \mathbf{y}_k) = (\mathbf{X}_i, \mathbf{y}_i) \forall i, z_i = k$ are the data associated with each expert. The GP parameters, $\theta = \{\theta_w, \theta_n\}$, are shared by all experts and used to generate the individual covariance matrices in each one, $\mathbf{Q}_k$.

Note that by tracking multiple particles, we are in effect performing Monte-Carlo integration over the possible partitions. Further, we have simplified somewhat from [12] by constraining all experts to use the same GP parameters. Doing so decreases the computational and storage requirements of ROGER (as individual GP parameters do not need to be tracked), but at the cost of some flexibility.

A. Inference

At the heart of ROGER is the inference technique, who's goal is to find the $\mathbf{z}$ that maximizes Equation 2. At all times we track a set of P particles, each with their own set of experts, with its partitioning into $K^{(p)}$ experts denoted $\mathbf{z}^{(p)}$. Shown in Figure 3 and Algorithm 1, inference proceeds by first temporarily assigning a new data point to each and every expert over all particles, including a possible empty expert (the $+1$ in the Algorithms). The resulting putative particles are then weighed by their likelihoods under Equation 2.

We use the the optimal threshold technique of [15] to select which putative particles to carry forward. This technique ensures that there is no duplication in the particle set by computing an optimal threshold; all particles with weights over this threshold are kept, and the rest are sampled randomly. We delay incorporating new data into each expert until after the final P particles are chosen. Further, each expert has a maximum capacity β, so a new data point may be incorporated without requiring additional storage.

B. Prediction

Prediction in ROGER for a query input is straightforward. We first chose a particle, select an expert from that particle, and generate an output from the chosen SOGP regressor. Our approach is shown in Algorithm 2, where we have chosen to always use the MAP particle, select the expert stochastically based on the probability that the query point was generated by their gating parameters, and return the predicted mean as our output, as well as the estimated variance, which can be used as a measure of confidence. When using ROGER to control a robot, the stochastic selection of the expert will be replaced by the selection dictated by the transition matrix.

Algorithm 1 ROGER-Inference

Require: Training pair $(\mathbf{x}, y)$
  P particles ($\mathtt{P}$) and weights ($\mathbf{w}$)
  Hyperparameters ($\Omega = \{\alpha, \mu_0, \kappa_0, \Lambda_0, \nu_0, \theta\}$)
Ensure: Updated particles and weights
  for $p = 1 : P$ **do**
    for $k = 1 : K^{(p)} + 1$ **do**
      create ρ_{pk}: assign $(\mathbf{x}, y)$ to expert k in particle p
      Weigh putative particle ρ_{pk} by 2
  Sample P putative particles and weights using [15].
  Assign data to selected experts with [13]

Algorithm 2 ROGER-Prediction

Require: Query point $(\mathbf{x}')$
  P particles ($\mathtt{P}$) and weights ($\mathbf{w}$)
  Hyperparameters ($\Omega = \{\alpha, \mu_0, \kappa_0, \Lambda_0, \nu_0, \theta\}$)
Ensure: predicted output $\hat{y}$, variance σ^2
  $p^* = \text{argmax}_p \, w_p$
  for $k = 1 : K^{(p)} + 1$ **do**
    $e_k = P(z' = i|z^{(p)}) P(\mathbf{x}'|\mu_0, \Sigma_0, \Lambda_0, \kappa_0)$
  Sample e^* according to $\mathbf{e}$
  Return mean ($\hat{y}$) and variance (σ^2) from expert e^*

TABLE I: ROGER parameters

PARAMETER	MEANING
α	CRP concentration
μ_0	Center of Input-Gate Centers
κ_0	Variance of Input-Gate Centers (with Λ_0)
Λ_0	Scale of Input-Gate variances
ν_0	Degrees of freedom of Input-Gate variances
θ_w	Kernel width of the GP
θ_n	Observation noise of the GP
P	The number of particles
β	Maximum storage in each expert

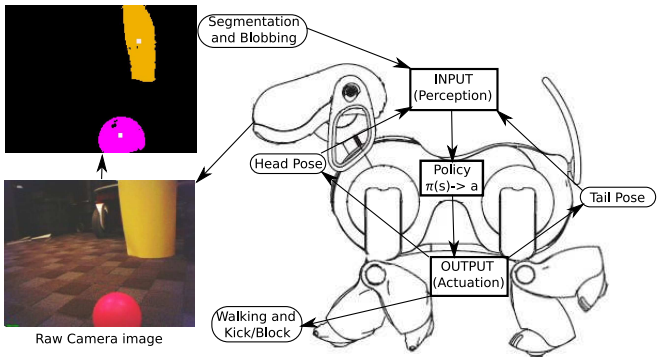
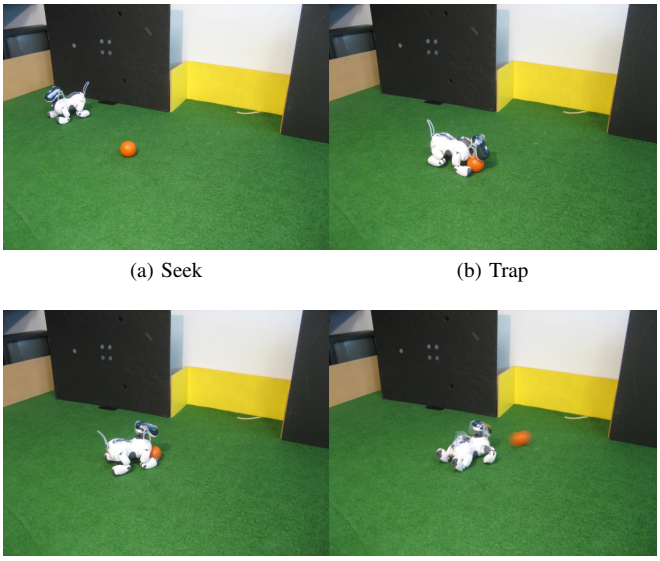


Fig. 4: Our platform, a Sony AIBO equipped with rudimentary vision and locomotion. We seek to learn the policy mapping perceived INPUTs to action OUTPUTs.



(a) Seek (b) Trap

(c) Aim (d) Kick

Fig. 5: Our multimap FSM task. In (c) the robot's perception does not include the location of the ball. Thus it can either seek the ball (a), if it is absent, or kick (d) if it is present.

C. Parameters

ROGER has 9 parameters, summarized in Table I, 7 which define the model ($\alpha, \mu_0, \kappa_0, \Lambda_0, \nu_0, \theta_w, \theta_n$), and an additional 2 which control the sparsity of the technique (P, β). The sparsity parameters are the 'knobs' that enable us to deliver realtime performance ($\sim$ 30 Hz). The lower they are (fewer particles, sparser experts), the faster the algorithm will be. Speed comes at the cost of accuracy, and we set P and β as high as possible while maintaining the desired runtime.

We set our other parameters heuristically, based on analysis of the data and prior knowledge. If we expect few subtasks, we set α low. μ_0 we set to the expected mean of the data (known from the limits of the perceptual space), and ν_0 to the size of the input space plus one. We set the other parameters to correspond to vague priors, so that they are rapidly outweighed by observed data. In our experiments we found our results robust in that optimizing the parameters to maximize the likelihood of the training data did not significantly change the results. The GP parameters we found to be the most sensitive and future work will investigate adapting these as data arrives, perhaps on a per-expert basis.

IV. EXPERIMENTAL VALIDATION

To validate ROGER as an incremental approach to model selection and subtask learning, we generate data from an FSM robot controller for a robot soccer scorer. The robot platform is in Figure 4 and consists of a Sony AIBO with color-based vision, a walk-gait engine, and two time-extended motions. The full input space is 23D (6 colors $\times$ 3 attributes (2D location and size), 3 DOF head pose, 2 DOF tail pose), while the output space is 9D (3 walk velocities (front/back, left/right, angular), head/tail poses, and discrete action indicator (kick, block, null)).

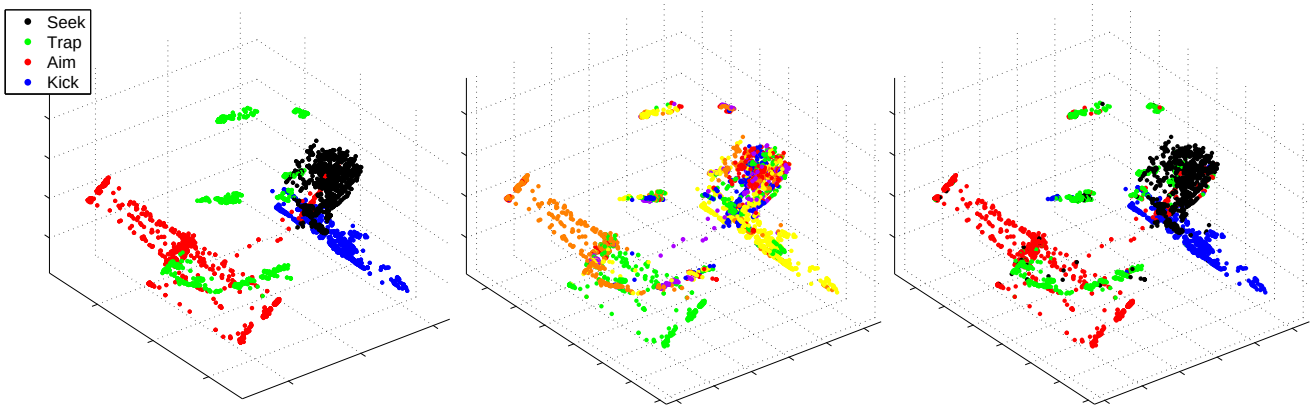
We use the FSM policy shown in Figure 5. The policy is to locate the ball, walk towards it, and when it is in range, "trap" the ball under the chin of the robot. Then, the robot turns towards the goal and kicks. However, while aiming, a multimap scenario occurs, as the robot is unable to see the ball and goal at the same time, and the robot's pose is insufficient to determine the appropriate action.

To provide demonstration data from this policy, we use a hand-coded controller (HCC), structured as the FSM in Figure 6a. As our focus is on learning the subtasks, we use an HCC instead of human joystick control in order to remove variability in the demonstration, and generate a ground-truth subtask assignment. Testing our HCC on the spread of locations in Figure 6b, we establish a baseline 69% success rate (goal scored within 2.5 minutes).

We collect data from one shot (failures included) from each location. Anticipating a bias towards subtasks that have produced more data, we first downsample (without changing the order) to 1000 random data points from each subtask and train ROGER on this data. While data *collection* was done in batch, data *processing* was incremental to show the applicability of ROGER to incremental learning scenarios.

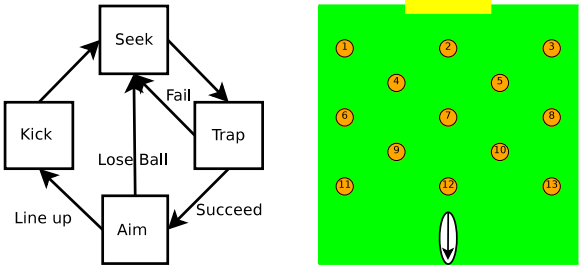
A. Results

Using ROGER ($\alpha = 0.5, \mu_0 = 0, \kappa = 0.1, \Lambda = 1, \nu_0 = 24, \theta_w = 0.1, \theta_n = 0.1, \beta = 300, P = 10$), we automatically determine a number of experts, the assignment of data to experts, and their individual policies. ROGER discovers 85 experts in our data, which is more than the 4 that were coded in the HCC. However, this difference is not an indication



(a) Human (b) ROGER (c) Grouped

Fig. 7: (a) The multimap goal scoring demonstration data, mapped into 3D with PCA and colored by hand-coded subtask. (b) Using ROGER, 85 experts are discovered. (c) Grouping the experts by subtask for transitioning purposes, we see that the discovered segmentation accords with the ground truth.



(a) FSM Controller (b) Field Setup

Fig. 6: (a) Our controller as a finite state machine. Multiple subtasks that are applicable in in the same perceived state lead to multimap control policies. (b) The 13 locations from which we test our goal scoring behaviors.

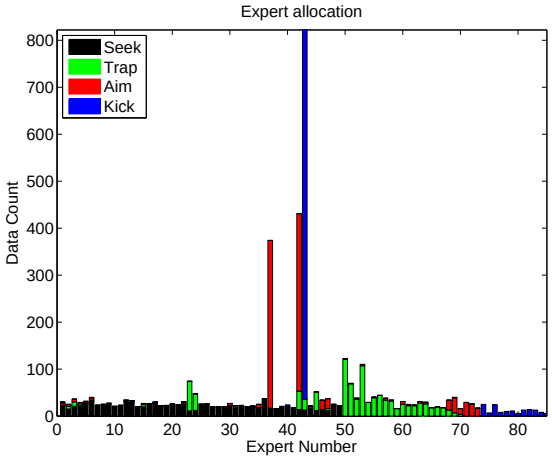


Fig. 8: The ROGER discovered experts and the data assigned to them, colored by subtask.

that ROGER has failed. Rather, the discovered experts may just represent an alternate partitioning of the overall task into subtasks than that which we used when coding. In fact, under our model, our discovered partitioning is *more* likely than the ground truth, even when the parameters were optimized to maximize the ground truth, including setting α to a very small value. These differences may result from simplifying assumptions in our model that do not accurately reflect the properties of the data. Changes to the model may make partitions with fewer experts more likely.

1) Segmentation analysis: We project the data into 3D using PCA on the joint space and color it according to ground-truth subtask in Figure 7a, and ROGER-discovered experts in Figure 7b. Further, to examine the data assignments, we plot them per expert as a stacked histogram in Figure 8.

We first note that almost all of the data from the kick subtask is assigned to one data point, number 43, corresponding to our a priori segmentation of the task. The data associated with this subtask is thus sufficiently similar to itself (in terms of inputs and outputs), and different from the data from other tasks that it is isolated and assigned to an expert. However, not all data from this subtask is in this expert, and this expert contains data from other subtasks as well. The data assigned to other experts can be thought of as outliers from this subtask, data which is sufficiently different that the gating GMM and output SOGP do not well describe it. Likewise, data from other subtasks that are included in this expert can be thought of as overlap data, from the regions where multiple subtasks are applicable.

Concerning the aiming subtask, the data is assigned predominantly to two experts, numbers 37 and 42. Examining in Figure 7b we see that one expert (orange) corresponds to one side of the data, and the other expert (green) to the other side. These sides correspond physically to the situations when the robot has the ball and must turn left to the goal, and the one where it must turn right. Upon consideration, this is a reasonable distinction to make. While the HCC treats these two cases as the same, it could also have contained

different subtasks for each. ROGER is thus able to discover distinctions not originally thought of by the demonstrator. This property is key, as it will allow ROGER to determine subtasks unknown even to the demonstrator.

Regarding the other subtasks (seek and trap) we note that data for these subtasks are not distributed randomly over experts, but rather split over a series of them. Each individual expert contains data from predominantly only one subtask, rather than a blend. Extrapolating from our previous point, we posit that ROGER is finding distinctions in the data that are not as obvious to us as the left/right split of the aiming task. Thus, a problem here may be that ROGER is finding too many distinctions, rather than not learning to distinguish.

2) Task Performance: Our goal is to discover experts that are sufficient for reperforming the overall task, which we test by combining the discovered experts with our known transitioner. If we are able to successfully score a goal in the intended fashion (via seek-trap-aim-shoot instead of by accident), then we can deem the subtasks sufficient. However, to use our coded transitioner we must map between subtasks and experts. We choose to group experts according to the the subtask that generated the majority of their data (major color in Figure 8), and the resulting groups are shown in Figure 7c, where they visually accord well with the ground truth.

We tested the resulting controller (HCC transitioner with ROGER-derived subtasks), with 3 attempts from each of the same 13 locations and achieved a 31% success rate. Thus we say that the learned experts, when combined appropriately, are sufficient (at least in some situations) to perform the overall task. For comparison, consider that standard DPA approaches score 0% of the time when trained on the same data, as they are unable to deal with the underlying multimap.

V. Discussion and Future Work

While our results show that ROGER is capable of discovering sufficient subtasks, with an additional 38% of goals missed over the performance of the HCC, there is much room for improvement in the learned goal scorer. The majority of the misses resulted from improper transitioning where, for example, the robot would find the ball, but fail to execute the trap maneauver. Future work that learns the transition matrix alongside the experts should alleviate these errors.

One possible adaptation would be to change the distribution over partitions, as in [16]. By making it more likely that a point is assigned to the same expert as the data point immediately temporally preceding it, we can encourage the formation of chains of data points that represent temporally-extended subtask execution. The trace of subtask activity over time would then accord with our intuition that in FSM controllers, rapid subtask oscillation does not occur.

Other errors result from improper subtask learning, due either to over- or under-generalization in each expert, or contamination between experts. The second issue should be addressed by the same techniques that encourage temporal consistency. The first issue we plan to address by allowing the GP parameters in each expert to vary, to better fit the data assigned to it. Further, this modification should also help with the large number of experts that are identified. Currently, our GP parameters are set small, so that multiple experts may be needed where one, larger, expert would suffice. However, uncoupling the GP parameters greatly enlarges the parameter space for each particle, and negatively impacts runtime.

VI. Conclusion

In the paradigm of interactive RLfD, we investigate incremental algorithms for policy learning. When dealing with perceptual aliasing arising from FSMs, current state of the art techniques require that the user provide indications of subtask switching, or take the subtasks as known in advance. We have gone beyond this by applying multimap regression based on an infinite mixture of sparse Gaussian process experts to automatically infer the number of subtasks and their individual policies from unsegmented, unannotated demonstration data. Using these learned subtasks, we have successfully recreated the overall composite task using a hand-coded transitioner. By incorporating a transition learner, future developments may be able to learn complete FSMs.

Acknowledgements

This work was supported in part by NSF Awards IIS-0534858 and IIS-0844486, ONR PECASE Award N000140810910, and a Brown Salomon Award.

References

[1] T. Inamura, M. Inaba, and H. Inoue, "Acquisition of probabilistic behavior decision model based on the interactive teaching method," in *Intl. Conf. on Adv. Robotics*, Tokyo, Oct. 1999, pp. 523–528.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469 – 483, May 2009.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[4] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Phil. Trans. of the Royal Soc. of London*, vol. 358, no. 1431, pp. 537–547, Mar. 2003.

[5] D. H. Grollman and O. C. Jenkins, "Sparse incremental learning for interactive robot control policy estimation," in *Intl. Conf. on Robotics and Automation*, Pasadena, May 2008, pp. 3315–3320.

[6] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Intl. Conf. on Machine Learning*, July 2004.

[7] D. H. Grollman and O. C. Jenkins, *From Motor to Interaction Learning in Robots*. Springer, 2009, ch. Can We Learn Finite State Machine Robot Controllers from Interactive Demonstration?

[8] S. D. Whitehead and D. H. Ballard, "Learning to perceive and act by trial and error," *Machine Learning*, vol. 7, no. 1, pp. 45–83, Jan. 1991.

[9] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Intl. Conf. on Mach. Learning*, June 2000, pp. 663–670.

[10] M. N. Nicolescu, O. C. Jenkins, A. Olenderski, and E. Fritzinger, "Learning behavior fusion from demonstration," *Interaction Studies*, vol. 9, no. 2, pp. 319–352, June 2008.

[11] A. K. Mccallum, "Reinforcement learning with selective perception and hidden state," Ph.D. dissertation, Univ. of Rochester, May 1996.

[12] E. Meeds and S. Osindero, "An alternative infinite mixture of Gaussian process experts," in *Neural Information Processing Systems*, Vancouver, CAN, Dec. 2005, pp. 883–890.

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

[14] J. Pitman, "Combinatorial stochastic processes," Notes for Saint Flour Summer School, 2002.

[15] P. Fearnhead, "Particle filters for mixture models with an unknown number of components," *Statistics and Computing*, vol. 14, no. 1, pp. 11–21, Jan. 2004.

[16] E. B. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky, "An HDP-HMM for systems with state persistence," in *Intl. Conf. on Machine Learning*, Helsinki, Finland, July 2008, pp. 312–319.