# Multi-Thresholded Approach to Demonstration Selection for Interactive Robot Learning

Sonia Chernova and Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
{soniac, veloso}@cs.cmu.edu

## ABSTRACT

Effective learning from demonstration techniques enable complex robot behaviors to be taught from a small number of demonstrations. A number of recent works have explored interactive approaches to demonstration, in which both the robot and the teacher are able to select training examples. In this paper, we focus on a demonstration selection algorithm used by the robot to identify informative states for demonstration. Existing automated approaches for demonstration selection typically rely on a single threshold value, which is applied to a measure of action confidence. We highlight the limitations of using a single fixed threshold for a specific subset of algorithms, and contribute a method for automatically setting multiple confidence thresholds designed to target domain states with the greatest uncertainty. We present a comparison of our multi-threshold selection method to confidence-based selection using a single fixed threshold, and to manual data selection by a human teacher. Our results indicate that the automated multi-threshold approach significantly reduces the number of demonstrations required to learn the task.

## Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

## General Terms

Algorithms, Performance, Human Factors

## Keywords

human-robot interaction, learning from demonstration

## 1. INTRODUCTION

Teaching by demonstration is a collaborative learning approach based on human-robot interaction that provides an intuitive interface for robot programming. In this paradigm, a robot learns to imitate the behavior of a teacher based on observations of the teacher's actions.


*HRI'08,* March 12–15, 2008, Amsterdam, The Netherlands.


A key component of all demonstration-based learning algorithms is the method for selecting the states in which the demonstration is to be performed. One commonly used approach is to allow the teacher to guide the learning and select all demonstration examples [2, 4]. Using this technique, the teacher alternates between demonstrating the task and observing the learned behavior, iteratively correcting mistakes made by the robot. This approach is based on the assumption that the teacher's expertise in the task is sufficient to recognize errors upon observation and to know what demonstrations would be useful for correcting the behavior.

Alternatively, several recent works have explored *interactive* approaches to demonstration, in which both the robot and the teacher are able to select training examples [5, 8, 9, 10, 11]. Using a *demonstration selection algorithm*, the robot identifies key states and queries the teacher for the correct action to perform. During the training process, the teacher alternates between observing task execution, selecting demonstrations, and answering robot queries. Demonstration selection is related to the areas of adjustable or adaptive autonomy [3, 7] and mixed initiative control [6]. However, while these approaches provide a mechanism for controlling the autonomy of the robot, they do not aim to improve the robot's long-term performance by acquiring additional training data.

In this paper, we examine demonstration selection techniques used by the robot. Since a teacher's time and patience are limited, an important objective of these algorithms is to learn the task from as few demonstrations as possible. However, all but the most simple tasks require repeated demonstrations to allow the learned model to generalize over variable conditions, such as sensor noise. Frequently, simpler elements of the task are learned before more difficult or noisier ones. In these cases, a good demonstration selection strategy will focus on the remaining problem areas instead of redemonstrating the entire task.

Existing selection algorithms typically select between autonomy and demonstration based on a measure of action confidence [5, 8, 10]. Given the current state of the robot, these algorithms calculate the certainty in selecting an action for that state and select between demonstration and autonomy based on this value. The underlying idea behind this approach is for the robot to seek help from the teacher when it is uncertain about which action to take. Providing an additional demonstration in a low confidence situation improves the robot's policy, leading to increased confidence, and therefore autonomy, in future similar states.

Given the action confidence, selection between autonomous

execution and demonstration is frequently dependent upon a fixed threshold value. For example, using a Bayesian likelihood method, Lockerd and Breazeal [10] apply a fixed threshold of 0.5 to identify low confidence behaviors, during which the robot uses emotive cues to solicit feedback from the teacher. Grollman and Jenkins [8] use a threshold of 0.25 in the Dogged Learning algorithm to identify low confidence states and indicate uncertainty to the human teacher. While a fixed threshold may capture the uncertainty of some algorithms, its use has negative implications in others, as we show in this paper.

In this work, we study thresholded demonstration selection in the context of *measurement level* classifiers [14]. In addition to returning a label, measurement level classifiers also provide a vector of confidence scores (measurement levels) representing the belief that the input belongs to each possible class. Many algorithms use measurement level in their calculations and are able to provide these values as output. Examples of such algorithms include Gaussian mixture models (GMMs), k-Nearest Neighbors (k-NNs), and multi-class Support Vector Machines (SVMs). In this paper, we highlight the limitations of using a single fixed threshold value with these approaches, and contribute an algorithm for automatically setting individual threshold values for each decision boundary of the classifier. Our approach is algorithm-independent and allows multiple thresholds to be set automatically.

In our evaluation, we compare the presented multi-threshold selection method to fixed-threshold and teacher-guided techniques. Demonstration selection is performed in the context of our *Confident Execution* learning algorithm, first presented in [5] and summarized in Section 2. Our results indicate that all three demonstration selection techniques result in policies comparable in performance. However, our multi-threshold approach requires *significantly fewer demonstrations* than the other methods.

In the next section, we present an overview of the Confident Execution algorithm. Section 3 discusses the drawbacks of the fixed threshold approach and Section 4 presents our algorithm for calculating multiple, adjustable thresholds. Section 5 presents a performance comparison of the three demonstration selection techniques in a complex simulated driving domain.

# 2. CONFIDENT EXECUTION

In this section, we describe our demonstration approach, the Confident Execution learning algorithm, and the effects of the autonomy threshold.

## 2.1 Demonstration Approach

Each demonstration results in a training point consisting of the robot's state and the action that must be performed. The robot's state is represented using an $n$-dimensional feature vector that can be composed of continuous or discrete values. The robot's actions are bound to a finite set $\mathcal{A}$ of action primitives, which are the basic actions that can be combined together to perform the overall task. Each labeled training point consists of the pair $(s, a)$, with state $s$ and teacher-selected action $a \in \mathcal{A}$. The goal is for each robot to learn to imitate the demonstrated behavior by learning a policy mapping states $s_i$ to actions in $\mathcal{A}$.

## 2.2 Learning Algorithm

Confident Execution is an interactive learning algorithm in which the robot must select demonstration examples, in real time, as it interacts with the environment. At each timestep, the algorithm determines whether a demonstration of the correct action in the robot's current state will improve the robot's policy. If demonstration is required, the robot requests help from the teacher and updates its policy based on the resulting action label. Otherwise, the robot continues to perform its task autonomously based on its policy. This incremental learning approach, in which each datapoint is either trained on or discarded, is similar to the active learning method used in the Query by Committee algorithm [13].

The robot's policy is represented and learned using supervised learning based on training data acquired from the demonstrations. Confident Execution can be combined with any supervised learning algorithm that provides a measure of confidence in its classification. The policy is represented by classifier $\mathcal{C} : s \rightarrow (a, c)$, which is trained using states $s_i$ as inputs, and training actions $a_i$ as labels. For each classification query, the model returns the model-selected action $a \in \mathcal{A}$ and action confidence $c$. Given a new state, the algorithm selects between demonstration and autonomy by comparing the action confidence $c$ to the autonomy threshold $\tau$. Confidence above the threshold allows the robot to autonomously execute the model-selected action, while confidence below the threshold leads the robot to request a demonstration.

Algorithm 1 presents the details of the Confident Execution algorithm with a fixed threshold value. We assume no preexisting knowledge about the task and initialize the algorithm with an empty set of data points $D$. The autonomy threshold $\tau$ is initialized to a constant value selected by the teacher.

The main learning algorithm consists of a loop (lines 3-12), each iteration of which represents a single timestep. At the beginning of each timestep, the robot records the state of its environment (line 4). The state vector $s$ is then used to query the learned policy model $\mathcal{C}$ and obtain the model selected action $a$ and confidence $c$ (line 5). The returned confidence value, which represents the robot's certainty in selecting an action in its current state, determines the robot's behavior. If the confidence is above the autonomy threshold $\tau$, the robot finishes the timestep by executing the model selected action $a$ (line 7). Confidence below the threshold initializes a request for teacher demonstration (lines 9-12).

**Algorithm 1** Confident Execution Algorithm with Single Fixed Threshold

```
 1: D ← {}
 2: τ ← constant
 3: while true do
 4:     s ← GetSensorData()
 5:     (a, c) ← C(s)
 6:     if c > τ then
 7:         ExecuteAction(a)
 8:     else
 9:         teacherAction ← GetTeacherAction()
10:         D ← D ∪ {(s, teacherAction)}
11:         C ← UpdateClassifier(D)
12:         ExecuteAction(teacherAction)
```

The robot requests a demonstration by pausing the execution of the task and indicating to the teacher (through sound, LEDs or other available interface) that a demonstration is required. Once the teacher indicates the correct action to perform, a new training datapoint consisting of the current state and the corresponding demonstrated action is added to the training set (line 10). The demonstration timestep is then completed by retraining the classifier (line 11) and executing the teacher-selected action (line 12).

Using this approach, the robot incrementally acquires datapoints representing the desired behavior. As more datapoints are acquired, the performance and classification confidence of classifier $\mathcal{C}$ improves, increasing the autonomy of the robot. Task learning is complete once the robot performs the desired behavior without requesting further demonstrations.

## 2.3 Autonomy Threshold

The autonomy threshold defines the domain region within which $\mathcal{C}$ classifies queries with high confidence. Equivalently, from the robot's perspective, the purpose of the autonomy threshold is to prevent unwanted behavior by limiting actions in regions of uncertainty, instead triggering a demonstration request to acquire more data.

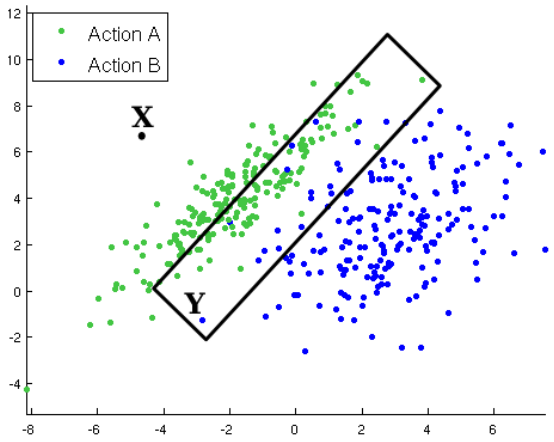
Consider, for example, the dataset presented in Figure 1(a). Each datapoint in the figure represents a demonstration of one of two actions, A or B. Our goal is to use the autonomy threshold to divide the state space into regions of high confidence (autonomous execution) and low confidence (demonstration). Low confidence areas represent uncertainty, and should include points that are unlike anything previously encountered by the robot, such as the point X in the figure.

Additionally, low confidence regions should include areas in which multiple classes overlap, such as region Y. From the robot's perspective, points in region Y represent demonstrations of two distinct actions from states that appear similar, and are difficult to distinguish based on the sensory data. This problem frequently arises in demonstration learning for a number of reasons, such as the teacher's inability to demonstrate the task consistently, noise in the sensor readings, or an inconsistency between the robot's and teacher's sensing abilities. We would like to set the autonomy threshold to a value that prevents either model from classifying the overlapping region with high confidence.

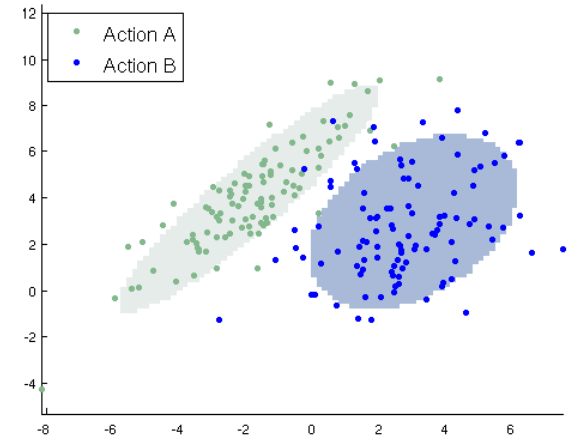
Previous implementations of the Confident Execution approach utilized a fixed threshold value that was manually selected through an extensive trial-and-error process [5]. For example, Figure 1(b) presents the classification of the sample dataset after a threshold of 0.01 has been applied to two Gaussian mixture models trained on the dataset. Shaded areas in the figure represent high confidence regions with confidence above this threshold. The resulting classification successfully avoids classifying points in the central overlapping region, while correctly classifying over 80% of the datapoints with high confidence. The remaining 20% of the points are considered outliers, and future observations similar to these points would trigger a demonstration request.

# 3. FIXED THRESHOLD APPROACH

A single, fixed autonomy threshold value provides a simple mechanism to approximate the high confidence regions of the state space. However, choosing an appropriate value can be difficult for a constantly changing dataset and model.




(a)




(b)

**Figure 1: (a) Example dataset, each point represents a single demonstration of one of two actions. Point X presents an example of a low confidence outlier. Rectangle Y highlights a region in which points from multiple action classes overlap. (b) Highlighted areas mark the high confidence regions for each action class, as defined by a fixed autonomy threshold value of 0.01 applied to a Gaussian mixture model.**

Figure 2 presents examples of three frequently encountered problems. Each diagram in the figure shows points from two action classes. A Gaussian is used to model the distribution of each data set, and the same fixed threshold value is applied to both Gaussian distributions in each example. Shaded areas represent the high confidence regions with confidence above this threshold.

Figure 2(a) presents a case in which two action classes are distinct and fully separable. A model trained on this dataset is able to classify the points with complete accuracy, without misclassifications. However, the current threshold value classifies only 72% of the points with high confidence, mark-

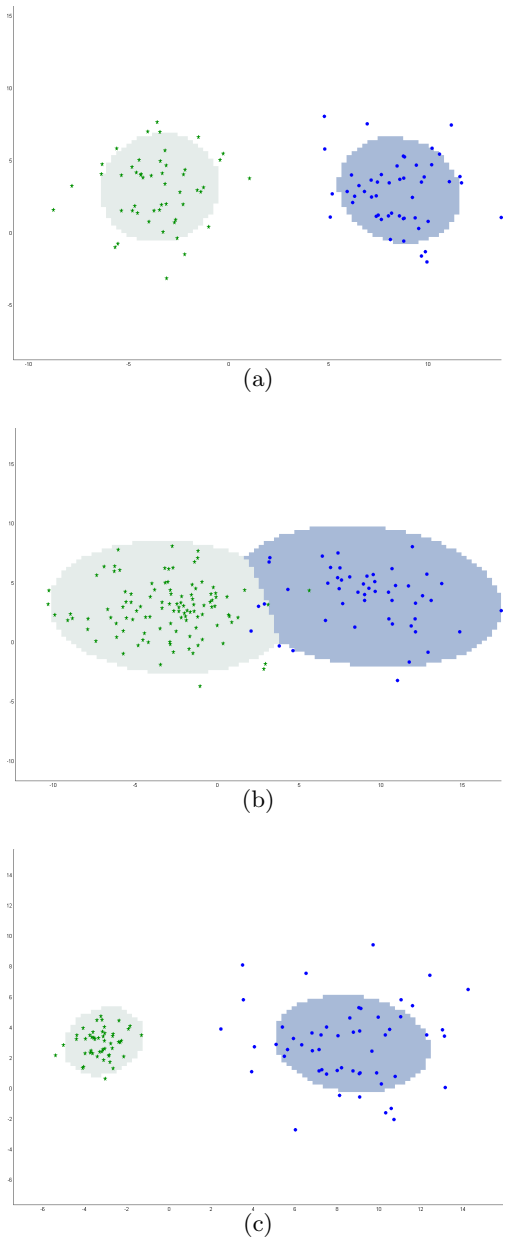


**Figure 2: Examples of fixed threshold failure cases: (a) Fully separable data classes with an overly conservative threshold value (b) Overlapping data classes with an overly general threshold value (c) Data classes with different distributions and common threshold value.**

ing the remaining 28% of the points as uncertain. In this case, a lower threshold value would be preferred that would allow the model to generalize more freely. The resulting larger high confidence region would reduce the number of redundant demonstrations without increasing the classification error rate of either data class.

Figure 2(b) presents an example of the opposite case, in which a stricter threshold value would be preferred. In this example the data classes overlap, resulting in a middle region in which points can not be classified with high accuracy. A higher threshold value would prevent the classification of points in this region into either data class, initiating instead a request for demonstration that would allow the teacher to disambiguate the situation.

Figure 2(c) presents a case in which the datapoints of the two data classes have very different distributions. While the fixed threshold value is appropriate for the left class, 42% of the points in the right class fall outside the threshold boundary.

Finally, the classification of complex multi-class data requires multiple decision boundaries (e.g., multiple Gaussian components for GMMs). Using the same value for all decision boundaries can exacerbate the problems highlighted above, as a single value often can not be found that constrains model classification in some areas while allowing generalization in others. The resulting effect is that the robot requests too many demonstrations about states it already knows, and too few demonstrations about unlearned behavior.

## 4. MULTIPLE ADJUSTABLE THRESHOLDS

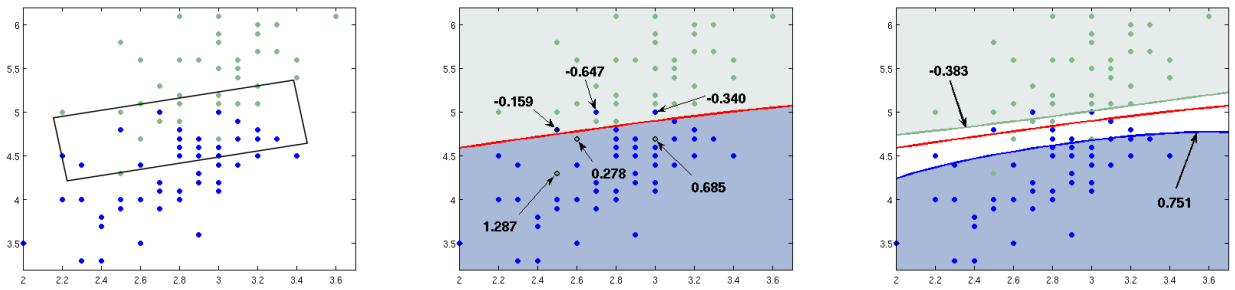
In this section, we contribute an algorithm for calculating multiple thresholds automatically, allowing a unique value to be set for each classifier decision boundary. This approach enables us to customize each threshold to its respective distribution. In our analysis, we assume that we are able to query the classifier and obtain three values: the most likely action class for the queried state, a confidence score representing the likelihood that the query belongs to this action class, and the decision boundary with the highest confidence for the query (e.g., Gaussian component for GMMs).

Algorithm 2 presents the details of the Confident Execution algorithm with multiple adjustable thresholds. The algorithm is initialized with an empty set of data points $D$. The array of autonomy thresholds $\mathcal{T}$, with one value for each decision boundary, is also initially empty.

The main loop of the algorithm (lines 4-15) again begins with the robot sensing its environment. The robot's current state is then used to query the classifier and obtain the recommended action $a$, classification confidence $c$, and decision boundary $db$. The threshold value of $db$, the most likely decision boundary to represent the current state, is used to decide between demonstration and autonomy (line 6). Con-

**Algorithm 2** Confident Execution Algorithm with Multiple Adjustable Thresholds

```
1: D ← {}
2: T ← [ ]
3: while true do
4:     s ← GetSensorData()
5:     (a, c, db) ← C(s)
6:     if c > T[db] then
7:         ExecuteAction(a)
8:     else
9:         teacherAction ← GetTeacherAction()
10:        D ← D ∪ {(s, teacherAction)}
11:        C ← UpdateClassifier(D)
12:        for each decision boundary db do
13:            M_db ← CalcMisclassifications(D, db)
14:            T[db] ← CalcAverageConf(M_db)
15:        ExecuteAction(teacherAction)
```




(a) (b) (c)

**Figure 3: Autonomy threshold calculation: (a) Example dataset, with highlighted overlapping region (b) Learned decision boundary, misclassified points marked with confidence values (c) Learned threshold values for each data class, a low confidence region containing most of the overlapping points remains in the center.**

fidence above threshold $\mathcal{T}[db]$ leads the robot to complete the timestep by executing the classifier selected action $a$ (line 7). Otherwise, the robot requests a demonstration and updates the classifier with the additional training data (lines 9-11).

Each time the classifier is relearned, autonomy threshold values are updated to reflect the new decision boundaries. Given the confidence scores of a set of points mistakenly classified by a decision boundary, *we assume that future classifications with confidences at or below these values are likely to be misclassifications as well.* The value of an autonomy threshold is therefore calculated as a function of the misclassification confidence scores.

Specifically, we define a classified point as the four-tuple $(s, a, a_m, c)$, where $s$ is the original state, $a$ is the demonstrated action, $a_m$ is the model-selected action, and $c$ is the classification confidence. Let $M_i = \{(o, a_i, a_m, c) | a_m \neq a_i\}$ be the set of all points mistakenly classified by decision boundary $i$. Given this data, the autonomy threshold value for each decision boundary is calculated as the average classification confidence of the misclassified points (lines 13-14):

$$\tau_{db} = \frac{\sum^{M_{db}} c}{|M_{db}|} \quad (1)$$

We take the average to avoid overfitting to noisy data. Other values, based on the maximum or standard deviation, can be used if a more conservative estimate is required. A threshold value of 0 indicates that no misclassifications occurred and the decision boundary can be applied freely for any state where it is most likely. The timestep of the learning algorithm is completed by executing the action demonstrated by the teacher (line 15).

Figure 3 presents an example of the threshold calculation process. Figure 3(a) shows a sample dataset, the rectangular box in the figure highlights a region of the state space in which points from both classes overlap. Figure 3(b) shows the learned decision boundary (in this case a SVM) separating the two data classes. Six misclassified points are marked with the (mis-)classification confidences returned by the model. Misclassified points on each side of the decision boundary are used to calculate the respective autonomy thresholds. Figure 3(c) shows the autonomy threshold lines and values based on the above calculations. The resulting low confidence region in the middle of the image captures most of the noisy datapoints.

| Algorithm | Correct-Misclas.-Unclass. | Thresholds |
|---|---|---|
| GMM | 98.6% – 0.4% – 1.0% | (0, 0, 0.012) |
| RF | 99.1% – 0.1% – 0.8% | (0.14, -0.355) |
| SVM quad. | 98.5% – 0.1% – 1.4% | (335.33, -68.77) |
| SVM RBF | 98.9% – 0.1% – 1.0% | (0.825, -0.268) |

**Table 1: Classifier comparison.**

The presented approach for calculating multiple autonomy thresholds is algorithm independent, and Figure 4 presents classification results for four different classification methods: Gaussian mixture models, random forests (RF), Support Vector Machine with a quadratic kernel, and SVM with a radial basis function (RBF) kernel. Table 1 summarizes the classification performance of each algorithm and lists the threshold values for each of the models.

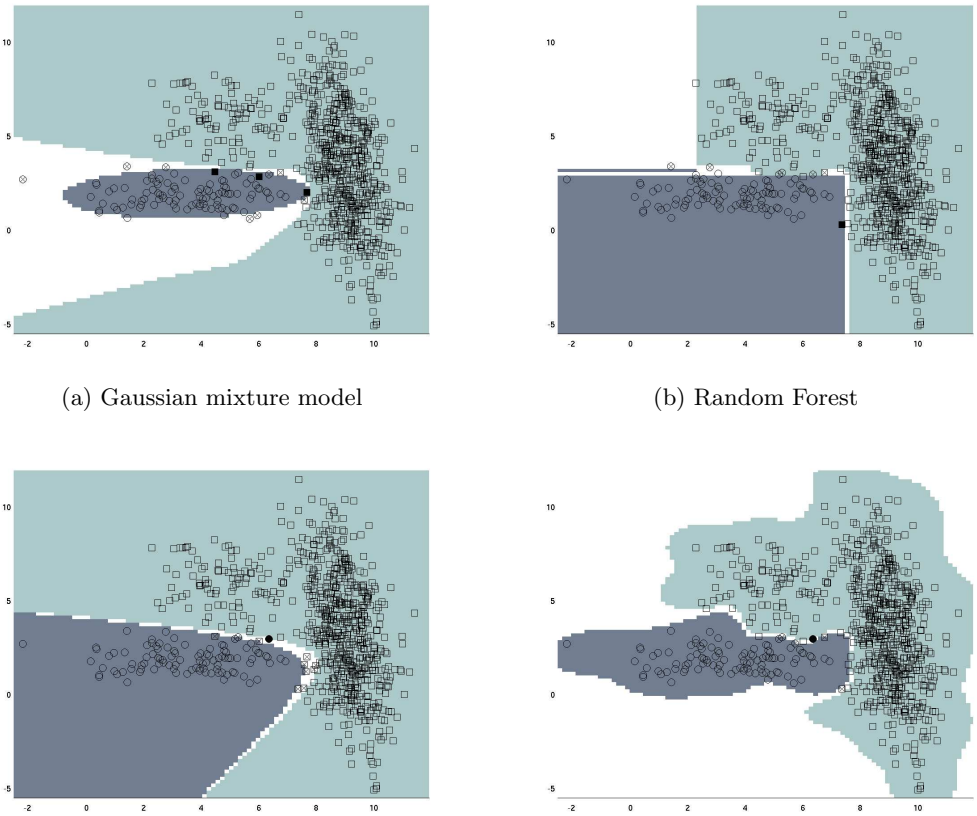
## 5. EXPERIMENTAL RESULTS

In this section, we analyze the performance of the confidence-based learning approach combined with the following demonstration selection techniques:

- *Single fixed threshold* – manually selected fixed threshold value
- *Multiple adjustable thresholds* – multiple thresholds, recalculated using above algorithm each time the classifier is relearned
- *Teacher-guided* – all demonstrations manually selected by the teacher, without confidence feedback from the algorithm

Although the Confident Execution algorithm assumes that the teacher is always able to initiate demonstrations, in order to allow a direct comparison between selection methods, only the robot was allowed to initiate demonstrations for both threshold-based approaches. The underlying policy of the robot was learned using multiple Gaussian mixture models, one for each action class, as described in [5].

The experiments were performed in a challenging simulated car driving domain (Figure 5), first introduced by Abbeel and Ng [1]. In this domain, the robot takes the shape



(a) Gaussian mixture model

(b) Random Forest

(c) SVM (quadratic)

(d) SVM (RBF)

**Figure 4: Classification of dataset into high and low confidence regions using four different classification algorithms.**

of a car that must be driven by the teacher on a busy road. The learner's car travels at a fixed speed of 60 mph, while all other vehicles move in their lanes at predetermined speeds between 20-40 mph. The learner can not change its speed, and must navigate between other cars to avoid collision. To do this, the robot is limited to three actions: remaining in the current lane, or shifting one lane to the left or right of the current position. The road has three normal lanes and a shoulder lane on both sides; the car is allowed to drive on the shoulder but can not go off-road. The teacher demonstrates the task through a keyboard interface, and the simulator is paused during demonstration requests.

The environment is represented using four features: the distance to the nearest car in each of the three lanes and the current lane of the learner. The learner's lane is represented using a discrete value symbolizing the lane number. The distance features are continuously valued in the [-25,25] range; note that the nearest car in a lane can be behind the learner. Distance measurements are corrupted by noise to create a more realistic testing environment. Learning is completed once the car is able to drive autonomously for 1000 timesteps.

The driving domain presents a varied and challenging environment; if car distances were to be discretized by rounding to the nearest integer value, the domain would contain over 600,000 possible states. Due to the complexity of the domain, the learner requires a large number of demonstrations to initialize the learned model, resulting in nearly constant demonstration requests early in the training process. To simplify the task of the teacher, we add a short (300 datapoint, or approximately 30 second) non-interactive driving demonstration session to initialize the learning process. While this learning stage is not required, it simplifies the task of the teacher for whom continuous demonstration is preferred over frequent pauses for demonstration requests.

Since the algorithm aims to imitate the behavior of the expert, no 'true' reward function exists to evaluate the performance of a given policy. However, we present two domain-specific evaluation metrics that capture the key characteristics of the driving task.

Since the demonstrated behavior attempts to navigate the domain without collisions, our first evaluation metric is the number of collisions caused by the learner. Collisions are measured as the percentage of the total timesteps that the learner spends in contact with another car. Always driving straight and colliding with every car in the middle lane results in a 30% collision rate.

Our second evaluation metric is the proportion of the time the learner spends in each lane over the course of a trial. This metric captures the driving preferences of the expert and provides an estimate of the similarity in driving styles. Each evaluation trial was performed for 1000 timesteps over

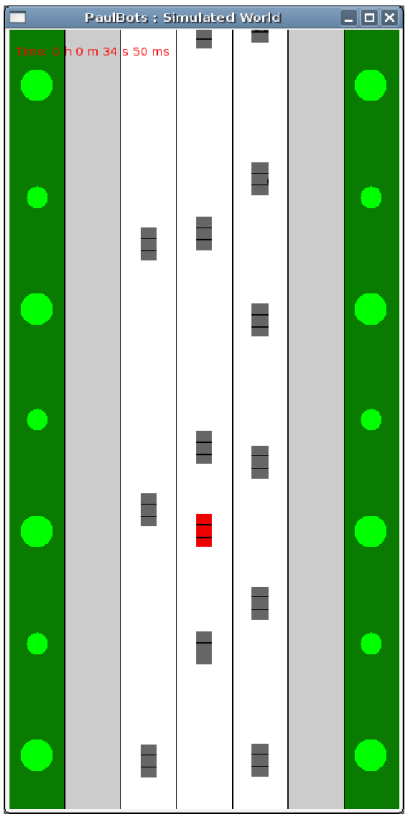



**Figure 5: Screenshot of the driving simulator.**

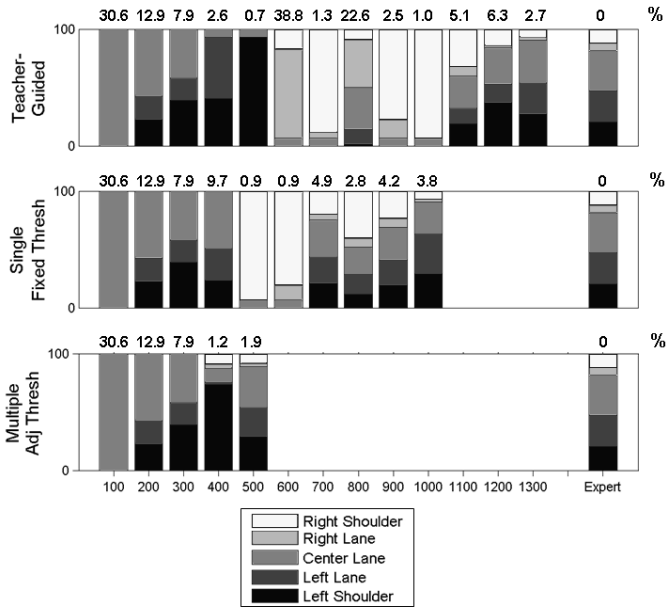



**Figure 6: Comparison of the agent's driving style throughout the learning process using three demonstration selection techniques. Each bar in the figure indicates the percentage of the time the learner spent in each road lane, with corresponding collision percentages above each bar. The teacher's lane preference over the same road segment is presented on the right.**

an identical road segment. Performance was evaluated at 100-datapoint intervals.

Figure 6 shows the agent's performance with respect to these metrics for each learning method. Each bar in the figure represents a composite graph showing the percentage of time spent by the agent in each lane. Collision percentages for each evaluated policy are reported above the bar graphs. The rightmost bar in the figure shows the performance of the expert over the evaluation road segment (not used for training). We see that the expert successfully avoids collisions, and prefers to drive most of the time in the center and left lanes, followed in preference by the left shoulder, right shoulder and right lane.

The top row in the figure summarizes the performance of the teacher-guided demonstration method. This non-interactive approach used the same learning algorithm as the other two methods, but demonstrations were selected manually by the teacher based on observations of the driving behavior. The teacher attempted to select demonstrations that would correct mistakes, without any feedback from the agent about action confidence. During learning, the agent's policy fluctuates greatly with regard to lane preference and collision performance. For example, after 500 demonstrations, the agent's preference is to drive on the empty left shoulder, thereby incurring few collisions. One hundred demonstrations later, the policy has shifted to prefer the center lane. However, the agent has not yet learned to avoid other cars, resulting in a 38.8% collision rate. The policy stabilizes after approximately 1100 demonstrations. Without confidence feedback from the robot, it is difficult for the teacher to select an exact termination point for the learning. Demonstrations persisted until the learner's policy showed little improvement, and learning was terminated by the teacher after 1300 demonstrations, upon obtaining final performance very similar to that of the expert with a low 2.7% collision rate.

The middle row presents results of the confidence-based, interactive learning approach with a fixed threshold value of 0.095. This threshold value was selected through multiple performance trials, and the best fixed threshold results are presented here. Using this approach, the agent's lane preference begins to resemble that of the expert after 700 demonstrations. Learning continued until, after 1000 demonstrations, the agent no longer requested demonstration examples, indicating high certainty in all encountered states. The final policy was similar to that of the expert, although with a slightly higher collision rate of 3.8%.

The bottom row presents learning results using multiple adjustable thresholds. This approach required the fewest datapoints to learn the task, completing learning after only 500 demonstrations. The resulting policy was again similar to that of the expert, with a small collision rate of 1.9%. Throughout the learning process, the number of Gaussian components within the model varied between 9 and 41. This large variation highlights the importance of automating the threshold calculation process, since hand-selecting individual thresholds for each component would be impractical.

In summary, all three selection techniques resulted in policies similar to that of the teacher and comparable to each other in performance. Most importantly, however, the presented multi-thresholded approach required significantly fewer

demonstrations to achieve this performance compared to the other two methods.

As presented in this paper, the learning algorithm does not flawlessly reproduce the teacher's behavior, as characterized by a small number of collisions in the driving domain. The cause of this problem is that an unwanted behavior, such as a collision, is expressed through *lack* of its demonstration in this training approach. As a result, the negative behavior is not explicitly represented by the algorithm. While this does not hinder our comparison, this challenge will be addressed in future work.

# 6. CONCLUSION

In this paper, we introduced a novel technique for selecting demonstration examples in the context of an interactive, human-robot learning algorithm. Using our approach, the robot selects between autonomous execution and demonstration requests based on multiple confidence thresholds designed to target domain states with the greatest uncertainty. Unique threshold values are automatically calculated for each decision boundary of the learned classifier based on misclassifications of the training data.

Experimentally, we compared three different methods of selecting demonstration training data: manual data selection by the teacher, confidence-based selection using a single fixed threshold, and confidence-based selection using multiple automatically calculated thresholds. The results indicate that while the learning algorithm is able to imitate the demonstrated behavior using all three data gathering techniques, the most informative demonstration examples are obtained using the multiple adjustable threshold approach introduced in this paper. By focusing demonstrations onto regions of uncertainty and reducing redundant demonstrations, this technique significantly reduces the number of demonstrations required to learn the task.

In this paper we have shown that an approach with a simple query-based interface can be used to teach complex tasks with relatively few demonstrations. This work lays a foundation for many additional studies that will enable non-technical users to program robot behaviors through interaction. The presented approach is algorithm-independent, allowing many systems to build upon this work. To address the problem of negative behavior, such as collisions, our future work focuses on developing an extension to the algorithm that will enable to teacher to correct mistakes made by the robot. Developing other interaction capabilities may also improve usability and performance. For example, enabling the robot to ask clarification questions or to catch user mistakes may speed up the learning process. A user study evaluating the usability of the system would also help to evaluate additional modes of interaction, such as enabling the teacher to provide high level guidance, rewards or punishments.

# 7. ACKNOWLEDGMENTS

This research was partially sponsored by BBNT Solutions under subcontract no. 950008572, via prime Air Force contract no. SA-8650-06-C-7606. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

# 8. REFERENCES

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, New York, NY, USA, 2004. ACM Press.

[2] B. Argall, B. Browning, and M. Veloso. Learning by demonstration with critique from a human teacher. In *Second Annual Conference on Human-Robot Interaction*, 2007.

[3] K. S. Barber, A. Goel, and C. Martin. Dynamic adaptive autonomy in multi-agent systems. *Journal of Experimental & Theoretical Artificial Intelligence*, 12:129–147, 2000.

[4] D. C. Bentivegna, A. Ude, C. G. Atkeson, and G. Cheng. Learning to act from observation and practice. *International Journal of Humanoid Robotics*, 1(4), 2004.

[5] S. Chernova and M. Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS'07)*, May 2007.

[6] T. W. Fong, C. Thorpe, and C. Baur. Collaboration, dialogue, and human-robot interaction. In *Proceedings of the 10th International Symposium of Robotics Research, Lorne, Victoria, Australia*, London, November 2001. Springer-Verlag.

[7] M. A. Goodrich, T. W. McLain, J. D. Anderson, J. Sun, and J. W. Crandall. Managing autonomy in robot teams: Observations from four experiments. In *Second Annual Conference on Human-Robot Interaction*, 2007.

[8] D. Grollman and O. Jenkins. Dogged learning for robots. In *IEEE International Conference on Robotics and Automation*, pages 2483–2488, 2007.

[9] D. H. Grollman and O. C. Jenkins. Learning robot soccer from demonstration: Ball grasping. In *Robotics: Science and Systems - Robot Manipulation: Sensing and Adapting to the Real World*, 2007.

[10] A. Lockerd and C. Breazeal. Tutelage and socially guided robot learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[11] M. N. Nicolescu and M. J. Mataric. Learning and interacting in human-robot domains. In *IEEE Transaction on Systems, Man and Cybernetics*, pages 419–430, 2001.

[12] M. N. Nicolescu and M. J. Mataric. Natural methods for robot task learning: instructive demonstrations, generalization and practice. In *Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 241–248, New York, NY, USA, 2003. ACM Press.

[13] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Computational Learning Theory*, pages 287–294, 1992.

[14] L. Xu, A. Krzyzak, and C. Suen. Several methods for combining multiple classifiers and their applications in handwritten character recognition. *IEEE Transactions on System, Man and Cybernetics*, pages 418–435, 1992.