Overcoming Blind Spots in the Real World: Leveraging Complementary Abilities for Joint Execution

Ramya Ramakrishnan[1], Ece Kamar[2], Besmira Nushi[2], Debadeepta Dey[2], Julie Shah[1], Eric Horvitz[2]
[1] Massachusetts Institute of Technology
[2] Microsoft Research

Abstract

Simulators are being increasingly used to train agents before deploying them in real-world environments. While training in simulation provides a cost-effective way to learn, poorly modeled aspects of the simulator can lead to costly mistakes, or blind spots. While humans can help guide an agent towards identifying these error regions, humans themselves have blind spots and noise in execution. We study how learning about blind spots of both can be used to manage hand-off decisions when humans and agents jointly act in the real-world in which neither of them are trained or evaluated fully. The formulation assumes that agent blind spots result from representational limitations in the simulation world, which leads the agent to ignore important features that are relevant for acting in the open world. Our approach for blind spot discovery combines experiences collected in simulation with limited human demonstrations. The first step applies imitation learning to demonstration data to identify important features that the human is using but that the agent is missing. The second step uses noisy labels extracted from action mismatches between the agent and the human across simulation and demonstration data to train blind spot models. We show through experiments on two domains that our approach is able to learn a succinct representation that accurately captures blind spot regions and avoids dangerous errors in the real world through transfer of control between the agent and the human.

Introduction

For handling complex, real-world tasks, many agents are first trained in simulation environments (Tobin et al. 2017; Mahler et al. 2017; OpenAI et al. 2018). While simulators are providing increasingly realistic training environments (Shah et al. 2018; Dosovitskiy et al. 2017; Chang et al. 2017) that can help accelerate initial learning, there is almost always a gap between simulation and reality (Ramakrishnan et al. 2018). Directly transferring learned policies to the open world in these cases can cause costly systematic errors, or *blind spots*, that occur due to differences between the two environments. Identifying blind spots before deployment in the real world is crucial for safe execution, yet it is challenging since the simulator lacks signals of such failures.

We focus on blind spots that occur due to missing observational features when going from a simulator to the real world. We assume that agents can perceive numerous features but learn to ignore many of them due to the incompleteness of the simulator to reflect the complexity of the real world. Consider an automated driving car that has been trained in a simulation environment. The simulator models general driving conditions, with different vehicles, such as cars and trucks, and the agent learns to drive well in this setting. To learn, the agent focuses on the most important features for decision making, such as distance to nearby vehicles and which lane it is in. Because the agent trains extensively in this simulator and has some sensor capabilities that are more accurate than human perception, it acts optimally in this world, much better than a human can act. However, the agent never encounters some aspects of the real-world (e.g., emergency vehicles) so the agent learns to ignore these aspects that did not matter in simulation. For example, the agent knows how to avoid cars and be cautious around trucks, but does not recognize ambulances as special types of vehicles that require different behavior. Because correct behavior around emergency vehicles is different than behavior around other vehicles, executing the learned policy in the real-world may cause costly mistakes – blind spots.

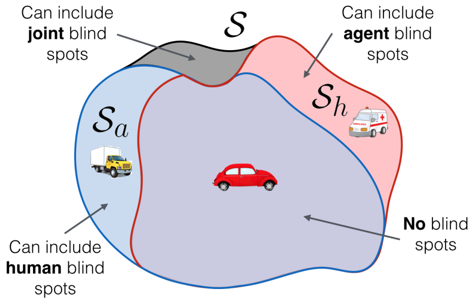

Can include **joint** blind spots
$\mathcal{S}$
Can include **agent** blind spots
$\mathcal{S}_a$
$\mathcal{S}_h$
No blind spots
Can include **human** blind spots

Figure 1: The agent is trained in a simulator world, which sees a subset S_a of the true state space S. Human demonstrations operate in subset S_h, and joint execution happens in the real world space S.

Directly executing the agent policy in the real world poses safety concerns, so we propose complementing the agent with a *black-box* helper agent. In contrast to prior work (Ramakrishnan et al. 2018), we do not assume that the helper agent is perfect in the real world. In fact, we assume that the helper agent has its own blind spots and noise in execution. While this secondary agent could be any autonomous agent, our assumptions about the helper agent being a black-box agent with suboptimal, yet complementary knowledge, are inspired by having a human in this role. Therefore, throughout this paper, we will refer to the helper agent as the human.

The only input we assume from the human is limited demonstration data collected from "demo world", a constrained version of the real world in which the human is comfortable acting. We assume that the human behaves in settings in which she is trained well in, which is complementary to the areas where the agent acts well. In the driving example, the agent is trained in an environment with many types of vehicles but no ambulances. A human would recognize ambulances as important for generating safe plans, but may not be comfortable driving on a highway with large trucks and vans so the human drives mainly on roads with no large vehicles. The demonstration data can thus be used to learn agent blind spots, while comparison of data from the simulator and demonstrations can provide signals of human blind spots. The real world can contain both of these scenarios. Figure 1 depicts the setup of the different worlds.

We formulate the problem of addressing agent and human blind spots as a two-step problem. First, the agent learns important features missed during training in simulation by using human demonstrations. Second, it learns separate predictors for agent and human blind spots to allow for *safe joint execution* by handing off control to the most capable agent. The first step requires learning a minimal feature representation that captures the important aspects from the simulator as well as from the real world. This representation learning is crucial because failing to learn important features in the real world can result in the agent being unable to identify important blind spot regions, leading to costly errors.

Once the agent has a minimal and accurate feature representation, the agent then learns supervised blind-spot predictor models for itself and the human based on simulator and demonstration data respectively. We assess the utility and success of our learned blind spot models by analyzing the effectiveness of a safety-oriented hand-off policy, which leverages the models to decide who should act in the real world, preventing therefore dangerous errors. Our meta-learning approach for modeling blind spots is motivated by the fact that neither the agent nor human has an understanding of each other's blind spots. Thus, the human may fail to recognize the incompleteness of the simulator. We address this gap in human and agent reasoning using a meta-model that can jointly learn about blind spots of both.

In our experiments, we evaluate the end-to-end approach over two different domains with distinct agent and human blind spots. We show that our framework learns important features and blind spot regions that help avoid costly mistakes in the real world by appropriately transferring control to the safest agent.

Problem Statement

We first discuss the overall formulation of our problem and then outline two subproblems that we aim to solve.

Overall Formulation

We define a real-world environment $M_{real} = \{\mathcal{S}, \mathcal{A}, T, R\}$ as an MDP with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function T, and reward function R. Each state $s \in \mathcal{S}$ can be mapped to a set of features $\Phi : \mathcal{S} \rightarrow F$, where $F = \{f_1, f_2, ..., f_n\}$ is a set of random feature variables. The function Φ maps a state to a set of feature values and is defined according to the domain. The agent is trained in a simulation environment $M_{sim} = \{\mathcal{S}_a, \mathcal{A}, T, R\}$, which is an imperfect model of the real world in that the state space is a subset of the true state space $\mathcal{S}_a \subset \mathcal{S}$. Because of this limited world, the agent only focuses on a subset of the full feature set $F_a \subset F$, where $F_a = \Phi(S_a)$. For example, the color of trees or buildings may not have mattered for acting in the simulation world, so the agent ignores these distracting features. Most importantly, because the simulation did not model some parts of the world, the agent ignores other important features that are needed for safe acting in the real world, such as uncommon traffic lights or ambulances.

The agent trains in the simulation environment using Q-learning and learns a policy $\pi_a : \Phi(S) \rightarrow \mathcal{A}$, which maps state features to actions. In this training process, the agent collects data from all of the state-action pairs it visited in simulation $D_{sim} = \{(s_i, a_i) | i = 1, .., n\} \sim M_{sim}$. We assume that π_a is optimal with respect to the states seen in simulation $s_a \in \mathcal{S}_a$ and that agent errors in the real world are solely due to features missing from F_a.

We define blind spots as regions in $\mathcal{S}$ for which the agent or the human make systematic errors in the real world that cause high negative reward. The agent blind spot region $BS_a \subset S$ is a subset of the real-world state space where there are clusters of errors caused by missing features from imperfect simulator modeling. For each state in the agent blind spot region $s \in BS_a$, the following equation holds:

$$Q^{\pi^*}(s, \pi^*(\phi(s))) - Q^{\pi^*}(s, \pi_a(\phi(s))) > \epsilon \qquad (1)$$

where Q^{π^*} is the optimal Q-value function in the real-world and π^* is the optimal policy obtained from this value function. We do not know either of these, which makes estimating blind spots in the real world challenging.

We assume access to human demonstrations in the form of state-action pairs $D_{demo} = \{(s_i, a_i) | i = 1, .., n\}$. These demonstrations are collected from a demo environment $D_{demo} \sim M_{demo}$, where $M_{demo} = \{\mathcal{S}_h, \mathcal{A}, T, R\}$ is an MDP with a state space that is a subset of the real world $S_h \subset S$. The human may make errors and act suboptimally when acting in this world. We do not know what features the human is using to act, but the agent observes the demonstrations through its feature set F. We assume that the human actions can be approximately explained by a subset of the agent feature set $F_h \subseteq F$, where $F_h = \Phi(S_h)$. The agent's estimate of the human representation is $\hat{F}_h$.

Similar to agent blind spot regions, we define a region in the real-world state space $BS_h \subset S$ that denotes human blind spots. For each state in this region $s \in BS_h$, the following equation is satisfied.

$$Q^{\pi^*}(s, \pi^*(\phi(s))) - Q^{\pi^*}(s, \pi_h(\phi(s))) > \epsilon \qquad (2)$$

Since the human is a black-box helper agent, we do not have access to π_h, so we cannot estimate the true set of human blind spots.

In terms of features, the set learned from simulation F_a is limited due to a limited S_a. The human feature set F_h is also limited because of a limited state space S_h in the demonstration world. The true feature space that includes all important features needed to take safe actions in the real world is defined as $F^* \subset F$, where $F^* \subset (F_a \cup F_h)$. We do not have access to the true optimal feature space so we estimate it in our approach $\hat{F}^* \subset (F_a \cup \hat{F}_h)$. We assume F will include some features that correlate with blind-spot regions. The feature set does not need to be complete with respect to the "true" representation of the real world, but the stronger the correlations of features in F are to agent and human blind spots, the better the blind spot models will be.

Breakdown into Subproblems

Because agent errors occur due to ignored features in simulation, we first need to identify important features needed for safe execution in the world. These features can then inform our predictions of where blind spots are likely to occur. Thus, we separate the problem of learning blind spots into two subproblems:

1. Learn an estimate $\hat{F}^*$ of the true feature representation that captures important features from both F_a and $\hat{F}_h$ needed for taking safe actions in M_{real}.
2. Learn estimated blind spot models $\Theta_a : \hat{F}^* \to p_a$ and $\Theta_h : \hat{F}^* \to p_h$, where $p_a, p_h \in [0, 1]$, to predict agent and human blind spots, respectively, for safe transfer of control in the real world.

The reason we do not use the full set of features F is because there are many distracting features that differ between the simulation and real-world environments that are not informative for acting in the real-world (e.g., the color of trees). Including such features into blind-spot modeling would lead to incorrectly labeling regions as blind spots.

As shown in Figure 2, the inputs to our problem are: the simulator policy π_a, the feature set F, data from simulation training D_{sim}, and human demonstration data D_{demo}. We treat the agent policy π_a as a black box to minimize assumptions about how the agent is trained. The first output is an estimate of the important set of features needed to act safely in the world $\hat{F}^* = F_a \cup (\hat{F}_m \subseteq \hat{F}_h)$, which includes features from the simulator world F_a as well as a subset $\hat{F}_m$ of features the agent is missing as estimated through human feature representation $\hat{F}_h$. Learning these missing features is crucial as they indicate blind spots of the agent. The second output is an estimated blind spot model for the agent $\Theta_a : \hat{F}^* \to p_a$ and one for the human $\Theta_h : \hat{F}^* \to p_h$ under the compressed feature representation $\hat{F}^*$.

In summary, these are the following characteristics of our problem that inform our approach for learning blind spots:

- Agent blind spots are due to representational incompleteness. In a real-world state in which the agent has a complete representation, the agent would act optimally.
- The human is a black box. Insights about the human model can be extracted from demonstration data collected from the demo world. Humans have complementary abilities to agents but can make errors and act suboptimally.
- Agent blind spots can be learned by observing data from demonstrations. Identifying features that the human is using for decision making in the demo world tells us about features the agent is missing. The missing features along with action mismatches signal agent blind spots.
- Human blind spots can be learned from simulation data. Since the agent has extensive experience in this world, it can compute differences in utility values of the agent's optimal action in simulation and the human's predicted action in that world as indicators of human errors.

Approach

We now present a two-step approach for discovering agent and human blind spots as shown in Figure 2. The first step uses behavior cloning and feature selection techniques to identify the set of features that the agent learned to ignore in simulation. The second step involves learning blind-spot models for the agent and human to safely transfer control. To do this, we extract noisy labels from simulator and demonstration data in the form of action mismatches and then train supervised learning models to predict blind spots.

Step 1: Identifying Missing Features

The goal of this step is to learn an augmented representation $\hat{F}^* = F_a \cup (\hat{F}_m \subseteq \hat{F}_h)$, which includes important features that the agent learned to ignore in the simulation environment but are important for safe acting in the real world.

We have multiple challenges to this end:

- *Insufficient features*: The agent's learned feature representation from the simulator F_a is not sufficient to represent the true world, so we need to identify missing features that are important to act in M_{demo} using demonstration data.
- *Distractor features*: Some features in the demo world are different from those in the simulation world, but they do not always indicate blind spots. Some of them might distract the agent because they indicate that these regions are different from the training environment while not affecting the policy. Thus, outlier detection techniques that identify which states differ from simulation would have low precision to identify blind spots.
- *Redundant features*: Features included into $\hat{F}_m$ are used as signals for identifying agent blind spots. Since F may include multiple features that are highly correlated, a feature should be included into $\hat{F}_m$ only if it is adding information for predicting human behavior in addition to the features in $\hat{F}_a$. Therefore, the construction of $\hat{F}_m$ should be conditional on $\hat{F}_a$.

To learn $\hat{F}^*$, we first use behavior cloning (BC) (Sammut et al. 1992; Urbancic and Bratko 1994) to learn an estimate of the human's representation and policy. This involves learning a state-action mapping from the demonstration data D_{demo}. We used a logistic regression model and selected

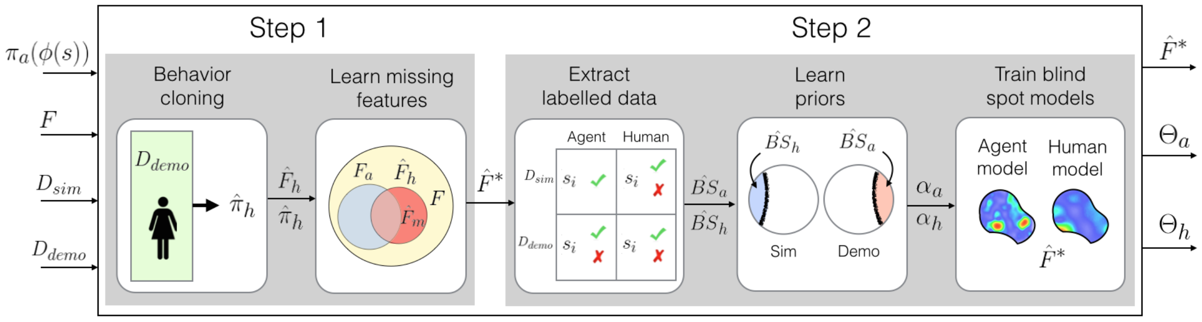


Figure 2: Pipeline of approach. The first step involves learning missing features that the agent learned to ignore in simulation. The second step is learning agent and human blind spot models in the real world, which involves extracting labeled data, learning priors of blind spots, and finally training the supervised learning models.

features with non-zero weights to obtain an estimate of the features the human is using $\hat{F}_h$. This step provides us two outputs: an estimate of the human policy $\hat{\pi}_h$ and an estimate of the human's representation $\hat{F}_h$. Both are noisy estimates because demonstration data is limited.

BC addresses two challenges: First, it learns features that are informative for the human policy, $\hat{F}_h$, by identifying which features correlate with actions that the human takes. Second, it eliminates distractor features that differ between simulation and real-world but are not important for acting by only selecting features that contribute to human decisions.

Since BC estimates $\hat{F}_h$ independently of F_a, $\hat{F}_h$ may include parallel redundant features. We only want to add missing features that are not already explained by features in S_a because we will then use these features to provide blind spot labels. For example, imagine that the agent already has features for detecting the distance of a car in front of it using a proximity sensor. We estimate that the human is using the feature of a car detector from camera data to obtain similar information. The agent ignored the camera data as it was less informative. These are two different features but correlated so the agent should not add this extra feature and consequently should not associate this part of the state space with agent blind spots.

We eliminate redundant features through a greedy feature elimination step. We start from the set of all possible relevant features $\{F_a \cup \hat{F}_h\}$ and iteratively eliminate features that do not decrease the accuracy of predicting the human demonstration data. The process is as follows: we first compute the initial accuracy of predicting the human actions in D_{demo} using $\{F_a \cup \hat{F}_h\}$. We then remove features one-at-a-time from $\{\hat{F}_h \setminus F_a\}$ and retrain the model to predict the demonstration data. If the best accuracy is not worse than the initial accuracy with all joint features, we remove the feature from the representation as it is not contributing to positive performance of the model. If many features, when removed, result in identical scores, we randomly choose one to remove. We continue with another iteration through all features and remove features individually, unless all features in the remaining set result in an accuracy drop. This gives us a final set of missing features $\hat{F}_m \subseteq \hat{F}_h$ that are not expressed in F_a. The final output is a succinct representation $\hat{F}^* = F_a \cup (\hat{F}_m \subseteq \hat{F}_h)$, which only includes features that add information over F_a for predicting D_{demo}.

The greedy feature selection process is one way to remove correlated features. While there can be interactions among features that could cause greedy elimination to incorrectly remove individual features when groups of features matter, we observed through experiments that our approach was able to keep enough features to obtain high performance.

Step 2: Modeling Agent and Human Blind Spots

Our setup does not allow for computing ground-truth labels for blind spots since reward signals or Q-values are unaccessible for demonstrations and the real-world environment. We instead use action mismatches between predicted human behavior and agent policy as noisy labels of blind spots. These extracted labels are noisy because mismatches are not true indicators of where blind spots may exist. Our agent and human blind spot models learn to identify potential error regions using this information by identifying patterns (state subspaces) that correlate with action mismatches.

Extracting Human Blind Spot Labels from Simulation The main insight for label extraction from simulations is that the agent policy is optimal for any state in simulation and any significant divergence of human behavior on these states indicates a human blind spot. For the human blind spot model, the labels are estimated as follows: For each $s \in D_{sim}$, we estimate the action the human will take $\hat{a}_h = \hat{\pi}_h(\phi(s))$ based on the estimated human policy learned from the first step.

To estimate how well $\hat{a}_h$ would perform in this state, the agent can use its learned Q-value function from the simulator. If the value of the human's action is much worse than the agent's action, as defined by some threshold, this state is assigned a blind spot label for the human. Otherwise, the human gets a safe label for this state. The top right quadrant in Figure 3 illustrates the human blind spot estimation in simulation states.

$$\Delta Q(s) = Q^{\pi_a}(s, \pi_a(\phi(s))) - Q^{\pi_a}(s, \hat{\pi}_h(\phi(s))) \qquad (3)$$

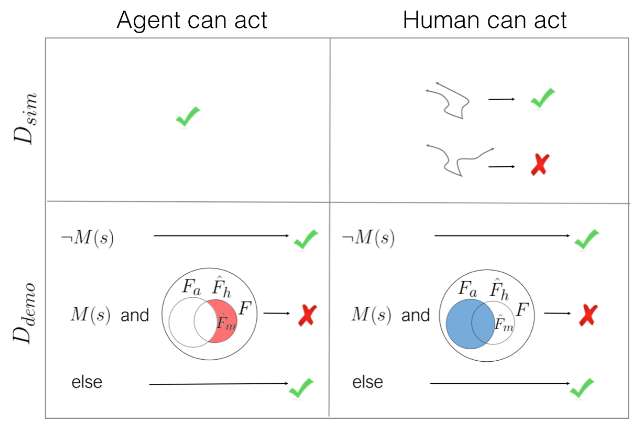


Figure 3: Data labeling to learn agent and human blind spots. Labels are extracted from simulation and demonstration data.

$$\hat{BS}_h = \{s \in D_{sim} | \Delta Q(s) > \epsilon\} \tag{4}$$

Each $s \in D_{sim}$ is assigned a safe label for the agent.

Extracting Agent Blind Spot Labels from Demonstrations Extracting labels for demonstration data is more challenging. The agent is no longer assumed to be perfect since the agent can have blind spots outside of the simulation environment. Further, humans can be suboptimal and make errors in demonstration data as well. Thus, deviations in human and agent behavior do not necessarily indicate that the agent has a blind spot. When actions match, both agents have similar behavior and either one can act. When actions mismatch, this is a noisy indicator of where one agent may be better suited to act than the other.

To estimate agent blind spots for states with action mismatches, we leverage the learned missing features $\hat{F}_m$ from our first step. We say that a feature $f_i \in \hat{F}_m$ is "activated" in a given state s, if $f_i = 1$ in s. Only a subset of features will be activated at any given state. If agent and human actions deviate and there is an activated feature from $\hat{F}_m$, this is an indication that the human is using a feature that the agent ignored, which is important for acting in the world. In this case, we assign a blind spot label to the agent. If actions do not match but the state does not have an activated feature from $\hat{F}_m$, there is no reason to think that the agent is acting suboptimally. Thus, this is an indication that the mismatch may be due to human random errors. In this case, we assign a safe label to the agent.

We define a function M (Equation 5) that indicates whether agent and human actions mismatch given a state. We then estimate agent blind spots using the mismatches and missing features $\hat{F}_m$ learned from step 1 (Equation 6).

$$M(s) = \begin{cases} \text{True} & \text{if } \pi_a(\phi(s)) \neq \hat{\pi}_h(\phi(s)) \\ \text{False} & \text{otherwise} \end{cases} \tag{5}$$

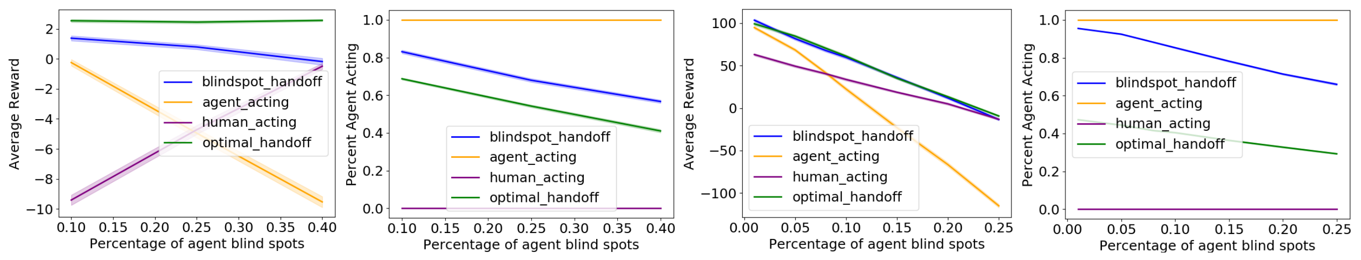
$$\hat{BS}_a = \{s \in D_{demo} | M(s) \text{ and } \phi(s) \cap \hat{F}_m \neq \emptyset\} \tag{6}$$

Extracting Human Blind Spot Labels from Demonstrations We assume that humans can make errors and act suboptimally in demonstration data. Disagreements with the optimal agent could indicate either human blind spots (systematic mistakes) or random errors. Assigning a disagreement label informs our models in two ways: If models can associate such disagreements with available features, the model identifies a blind spot. If not, disagreements inform the model about random human errors and guide blind spot models to prefer agent acting over human acting when no blind spot is detected for either. When blind spot models are oblivious to human suboptimality, hand-off decisions may be misguided. Imagine that the human is 20% suboptimal (randomly deviates from the optimal policy in 20% of demonstrations) in regions without blind spots, while the agent is optimal in this region. Blind spot models that do not know about these human errors would give equal chance of control to the human and the agent, leading to degraded performance. Thus, if agent and human actions deviate at a state, and the state does not have an activated feature from $\hat{F}_m$, we consider this mismatch as a random human error and assign a human blind spot label for this state.

Prior Estimation We have two data sources from which we are extracting blind spot labels. Demonstration data is limited in size, while simulation data is easily available in large amounts. These data sets differ in their representation of agent and human blind spot labels. However, these data proportions do not necessarily reflect the real-world proportions of blind spots. Simply training a supervised learner with both datasets will result in an imbalance in blind spot predictions (i.e., significantly more human blind spots as there is more simulation data available). Getting informative blind spot predictions from these models depends on estimating informative priors, guiding models about the likelihood of agent and human blind spots.

In our problem, the simulation world is more likely to contain human blind spots, and the demo world is more likely to contain agent blind spots. We assume that the relative proportions of human and agent blind spots respectively in D_{sim} and D_{demo} are representative of the proportions of these blind spots in the real world. This means that although there might be imbalance between D_{sim} and D_{demo}, the frequency of blind spot regions vs. safe regions within each environment is realistic in isolation. We use this insight to estimate the prior of agent blind spots α_a, human blind spots α_h, and no blind spots (i.e., both agent and human can act) α_n in the real world by solving the following equations, which provides a unique solution. The term α_n represents the unknown proportion of the real world in which both the agent and human can act. Estimating priors allows us to prioritize demonstration data appropriately to better learn blind spots.

Equation 7 computes the proportion of blind spots in simulation and demonstration data respectively based on the extracted data labels. Equation 8 relies on the insight that the simulation world contains mainly safe regions (both can act) and human blind spots, and similarly, the demo world contains safe regions and agent blind spots. Equation 9 states that the full real-world state space consists of regions with



(a) Catcher: Average reward (b) Catcher: Percent agent acting (c) Driving: Average reward (d) Driving: Percent agent acting

Figure 4: Performance of approach with varying agent blind spots in both domains.

no blind spots, agent blind spots, and human blind spots. We do not handle the case where there are joint blind spots. Values α_h^{sim} and α_a^{demo} are directly computed from data as in Equation 7, while the actual priors α_h and α_a are recovered based on Equations 8 and 9.

$$\alpha_h^{sim} = \frac{|\hat{BS}_h|}{|s \in D_{sim}|}, \alpha_a^{demo} = \frac{|\hat{BS}_a|}{|s \in D_{demo}|} \tag{7}$$

$$\alpha_h^{sim} = \frac{\alpha_h}{\alpha_h + \alpha_n}, \alpha_a^{demo} = \frac{\alpha_a}{\alpha_a + \alpha_n} \tag{8}$$

$$\alpha_a + \alpha_h + \alpha_n = 1 \tag{9}$$

Training Blind Spot Models Given the labels extracted from simulation and demonstration data as well as the learned priors, we train supervised learning models to predict agent and human blind spots in the real world. We oversample the extracted labeled data in a way that the sampled data matches the learned priors. We then train two random forest classifiers, one for predicting agent blind spots and one for predicting human blind spots. We perform cross-validation over various hyperparameters and choose the one with the best average F1-score. We train the final model on the entire training data and output a blind spot model for the agent $\Theta_a : \hat{F}^* \rightarrow p_a$ and one for the human $\Theta_h : \hat{F}^* \rightarrow p_h$, where $p_a, p_h \in [0, 1]$ are probabilities of a state being a blind spot. While we learn with batch simulator and demonstration data, similar concepts can be applied to already established systems that can improve with real-world data. This allows for iterative and incremental deployments. Here, we focus on developing a framework for safe execution in the real world *before* acting in the true environment.

Transfer of Control in the Real World through Blind Spot Models We use the learned agent and human blind spot models to do safe transfer of control in the real-world environment M_{real}. At each state $s \in S$, we query our learned agent blind spot model $p_a = \Theta_a(\phi(s))$ and human blind spot model $p_h = \Theta_h(\phi(s))$. If $p_a \leq p_h$, the agent acts. Otherwise, the human acts. Note that there are many ways to do transfer of control given these models (e.g., give preference to the agent if it can act, minimize the number of hand-offs). We use a simple hand-off scheme to evaluate how well our models capture failure regions and can avoid them through joint execution.

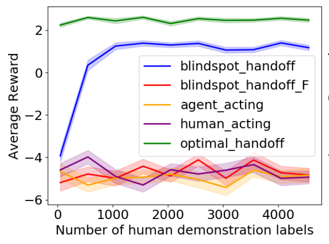
Experiments

Catcher The first domain is a variation of the game Catcher, in which the agent needs to catch falling fruits (Ramakrishnan et al. 2018). There are two features that describe the fruits: size and color. The feature space has 19 features for each value of the attributes, corresponding to the location of the agent with respect to the fruit (-9 to +9). The encoding is one-hot for each attribute-value (e.g., if there is a red fruit 4 units away, the red-color-+4 feature would be turned on while red-color-i $\forall i \in [-9, +9] \backslash (+4) = 0$). There are 20 additional distractor texture features that differ between simulation and real world but do not affect the policy. The actions the agent can take are moving left, moving right, or staying. The agent sees red and blue fruits with the same size in simulation that causes it to ignore size. The human only sees blue fruits that are either large or small. The agent needs to learn the missing and important size features while disregarding unnecessary texture features. In the true real-world representation, human blind spots are red, small fruits, and agent blind spots are blue, large fruits. The agent and human policies are obtained using linear Q-learning on the simulation and demo worlds, respectively.

Driving The second domain is based on simulated highway driving (Abbeel and Ng 2004; Syed, Bowling, and Schapire 2008; Levine, Popovic, and Koltun 2010). The agent must navigate a three-lane highway while avoiding cars and trucks. The simulator, however, does not model ambulances, which exist in the real world. The feature space includes the following one-hot encoded features: 5 features for which lane the agent is in, 27 features for the closest car in each lane (i.e., 9 features for the closest car in the agent's lane, 9 for the closest car to the left, and 9 for the closest car to the right), a similar set of 27 features for the closest ambulance in each lane, and another 27 features for closest trucks. Additionally, there are 20 distractor tree features. The action space includes moving left, moving right, and staying in the same lane. Human blind spots are states with trucks nearby, and agent blind spots are states with ambulances nearby.

Performance

We first show that there is benefit from learning agent and human blind spot models to do safe transfer of control in the real world. The baselines are: An agent that executes in the real world by simply using π_a learned from training and a human that acts in the world using the true human policy π_h.


(a) Catcher: Average reward

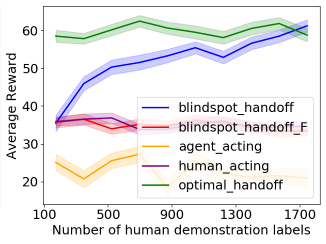
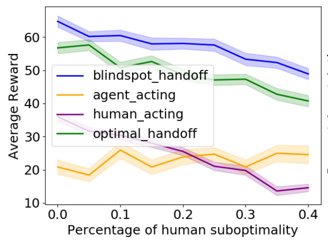

(b) Driving: Average reward

Figure 5: Effect of distractor features on predicting blind spots in the real world.


(a) Driving: Average reward

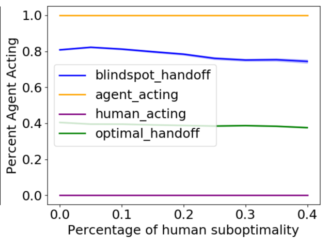

(b) Driving: Percent agent acting

Figure 6: Effect of human suboptimal behavior on our model's performance in the Driving domain.

We compare these baselines to our approach which uses the learned blind spot models to appropriately hand over control. We also compare against an "optimal hand-off" baseline, which we obtain by training a meta Q-learner agent in the real-world M_{real} that has two action choices at each state s: $\pi_a(\phi(s))$ or $\pi_h(\phi(s))$. This meta-agent learns a Q-function specifying the value of taking each action at each state. The plotted reward is the average reward received in M_{real} when the appropriate hand-off policy is run.

In Figure 4a, we increase the percentage of agent blind spots in the real world in the Catcher domain, while keeping the total percentage of human and agent blind spots the same. The blindspot_handoff condition, which uses our learned blind spot models to act in the world, is able to do almost as well as the optimal_handoff policy. Executing the agent policy π_a or the human policy π_h in the real world receives much less reward due to their inability to operate in their respective blind-spot regions. Figure 4b shows the percentage of times the agent acted using each hand-off policy. Our model gives more control to the agent while still obtaining high reward. The preference to the agent acting is due to our approach reasoning about features and transferring control to the human only when the state contains important real-world features the agent ignored in simulation. Further, our prior estimation step helps to better estimate the proportion of times the agent and human should act in the world, according to how likely their blind spots will occur. Giving greater control to the agent in areas where it can act well allows us to be more robust to human noise (see Figure 6).

Figures 4c and 4d show similar results on the Driving domain: The agent and human policies perform worse as the percentage of blind spots is varied, while our model is able to achieve similar performance to the optimal handoff. In this domain, performance is reduced as the percentage of agent blind spots increases because the scenarios become harder, and the best reward that can be achieved decreases. In other words, although the blind spot models may accurately predict a blind spot, the human or agent policies might not have a safe action available to take due to the high difficulty of the state (e.g., too many ambulances in the surrounding region).

Effect of Distractor Features

We next analyze the ability of our approach to prune away distractor features, which are features that differ between simulation and the real world but do not affect policy behavior. As we increase the amount of human demonstration data, performance in the world improves. Specifically, Figures 5a and 5b show that our approach blindspot_handoff is able to reach the performance of optimal_handoff with enough budget in both domains. Again, both agent or human acting obtain much less reward.

To highlight the benefit of our feature selection algorithm (Step 1 of our pipeline), we create a baseline that skips the feature selection step and instead uses all features in F (named blindspot_handoff_F). For example, in Driving, the experimental setup is that there are green trees in simulation and red trees in the demo/real worlds. Because the features are not pruned out, this hand-off policy focuses on red trees as an important difference between sim and demo/real. Thus, the hand-off policy always gives the human full control because red trees exists in all states, which is mistakenly associated with agent blind spots. Figures 5a and 5b show that in both domains, blindspot_handoff_F only does as well as π_h because it always gives control to the human. Our model learns to ignore distracting texture features in Catcher and tree features in Driving to learn an effective hand-off policy.

Effect of Human Suboptimality

In the results presented so far, the human had blind spots, but there were no random errors in the human policy, or existence of suboptimal actions in non-blind-spot states. Next, we analyze how robust our hand-off approach is to human suboptimality. As shown in Figure 6a, we see a degradation in the real-world performance as we increase human suboptimal behavior. However, our model is more robust than the baselines because it learns to prioritize the agent in cases where both can act, while still transferring control to the human when there is a high chance of an agent blind spot. Figure 6b shows that our model gives more control to the agent than the baselines. Blindspot_handoff does better than optimal_handoff because the optimal policy transfers control based on a human that does not make random errors. Even though our model has no prior knowledge on human suboptimality, we can still be robust and handoff control to avoid blind spot regions and regions with random human errors.

Related Work

Imitation Learning Inverse reinforcement learning methods (Zhifei and Meng Joo 2012; Arora and Doshi 2018;

Ng, Russell, and others 2000; Abbeel and Ng 2004; Ziebart et al. 2008) can be used to estimate a policy of an agent given demonstrations, under the assumption that the agent is optimizing a reward function of an unknown MDP. This assumption does not apply to our helper agent as it is modeled by characteristics of humans. Thus, we instead use behavior cloning methods for estimating the policy of the helper agent. The IRL approach proposed by (Levine, Popovic, and Koltun 2010) is relevant to our work in that it jointly learns a feature representation and a reward function. The approach learns the best features to explain the data, by constructing new features that are logical conjunctions of basic features and by pruning out irrelevant features. This work is complementary to ours, as it can be used to augment our feature set with high-level features constructed from the available set.

Novelty/Outlier Detection Methods for outlier and novelty detection (Chandola, Banerjee, and Kumar 2007; Pimentel et al. 2014) identify datapoints that differ significantly to most examples in a dataset. Since these methods focus on identifying rare instances, the discovery is solely guided by the estimated distance of a given instance to the training distribution. Such distance metrics become uninformative when agents have access to many features that include distractor and redundant features. Instead, we focus on identifying blind spot regions that can be characterized by discoverable missing feature patterns and agent-human mismatches.

Ad-Hoc Teamwork Research on ad-hoc teamwork investigates approaches for agents to coordinate with many teammates of known or unknown types to achieve a shared goal (Stone et al. 2010). The problems in this space include learning how to model teammates, learning how to communicate with them, and achieving robustness and adaptation to diverse partners (Barrett et al. 2014; 2012; 2016). Our problem shares similarities with this literature in that the agent is tasked with collaborating with another agent of unknown characteristics, and they cooperate towards a common goal. Our work is complementary, as it leverages joint execution to address the blind spots of cooperating agents.

Safety and Transfer Safe reinforcement learning is an active research area. Many techniques focus on cautious exploration but do not address the scenario where the agent representation is flawed that prevents calibrated uncertainty estimates (Sui et al. 2015; Garcıa and Fernández 2015; Kahn et al. 2017). Wray et.al., proposed an approach for managing hand-off decisions between an agent and a human when models of both are perfectly known (Wray, Pineda, and Zilberstein 2016). Transfer learning (Taylor and Stone 2009; Barrett, Taylor, and Stone 2010; Christiano et al. 2016) is a more general technique that applies knowledge learned in one domain to a new one, which can be simulation to real-world or across different domains. While representations may differ between domains, and agents may need to learn a mapping, it is assumed that knowledge of the agent generalizes to the new domain. We study adapting two agents with blind spots to a new domain through joint execution.

Discussion

Our meta-learning approach for learning agent and human blind spots is an important step towards more active consideration of safety when transferring from simulation to the real-world. In our work, we assume access to additional features that the agent can leverage to learn blind spots. The performance of our model will vary based on how correlated the available features are to error regions. For example, if the agent has a noisy feature that can predict 30% of ambulances, our model can learn to transfer control to the human for those states. If a perfect ambulance detector is added, accuracy would improve on predicting these blind spots.

Given some correlated features, our approach aims to learn blind spots *before* acting in the real world. In our setting, both the agent and the human can make errors, and we do not have access to a reward signal. Thus, we use action mismatches as noisy indicators of where blind spots may exist. We expect disagreements to be good indicators because (1) mismatches can indicate suboptimal behavior and (2) unsafe regions are generally subsets of suboptimal regions. Therefore, if the agent can predict action mismatches well, it is likely to over-approximate blind spots, resulting in a conservative hand-off policy that avoids suboptimal, and thus, unsafe states. It may be interesting in future work to combine these ideas with scenarios in which we have sparse reward signals to make more informed blind spot predictions.

We chose to learn a hand-off policy rather than update the agent's policy because estimating the true policy can require much more data than just learning blind spot regions. Further, the agent may not be capable of representing the correct policy. For example, if the agent does not have the full feature set, it may be able to identify a blind spot region but not able to update its policy for those different situations. Blind spot detection and safe transfer of control allows for better awareness of where the policy needs to be improved.

Finally, our approach assumes access to a set of features to learn limitations of the simulator. In future work, it would be interesting to apply these ideas to deep RL approaches, which are often featureless. One idea is to form a feature library that includes high-level features extracted from deep learning models (Ribeiro, Singh, and Guestrin 2016) (e.g., superpixels or detector for car), which can then be used by our approach. Such a library can be used across applications to identify important high-level features missed in simulation that are important for the real world.

Conclusion

We have formalized the problem of agent and human blind spot detection in reinforcement learning to do safe transfer of control in the real world. In the proposed approach, the agent first learns important features from human demonstrations, which were ignored in its simulation environment. The agent then estimates its own blind spots as well as the human's to allow for safe execution in the world. Results show that using our blind spot models, the agent can learn to transfer control to the human intelligently to avoid blind spots in the real world. In future work, the representation and blind spot models can be updated in an online fashion with real-world feedback. The agent can also use active learning approaches to guide the collection of demonstration data for learning better blind spot models.

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, 1. ACM.

Arora, S., and Doshi, P. 2018. A survey of inverse reinforcement learning: Challenges, methods and progress. *arXiv preprint arXiv:1806.06877*.

Barrett, S.; Stone, P.; Kraus, S.; and Rosenfeld, A. 2012. Learning teammate models for ad hoc teamwork. In *AAMAS Adaptive Learning Agents (ALA) Workshop*, 57–63.

Barrett, S.; Agmon, N.; Hazon, N.; Kraus, S.; and Stone, P. 2014. Communicating with unknown teammates. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, 1433–1434. International Foundation for Autonomous Agents and Multiagent Systems.

Barrett, S.; Rosenfeld, A.; Kraus, S.; and Stone, P. 2016. Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence*.

Barrett, S.; Taylor, M. E.; and Stone, P. 2010. Transfer learning for reinforcement learning on a physical robot. In *Ninth International Conference on Autonomous Agents and Multiagent Systems-Adaptive Learning Agents Workshop (AAMAS-ALA)*.

Chandola, V.; Banerjee, A.; and Kumar, V. 2007. Outlier detection: A survey. *ACM Computing Surveys*.

Chang, A.; Dai, A.; Funkhouser, T.; Halber, M.; Nießner, M.; Savva, M.; Song, S.; Zeng, A.; and Zhang, Y. 2017. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*.

Christiano, P.; Shah, Z.; Mordatch, I.; Schneider, J.; Blackwell, T.; Tobin, J.; Abbeel, P.; and Zaremba, W. 2016. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*.

Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; and Koltun, V. 2017. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*.

Garcıa, J., and Fernández, F. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16(1):1437–1480.

Kahn, G.; Villaflor, A.; Pong, V.; Abbeel, P.; and Levine, S. 2017. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*.

Levine, S.; Popovic, Z.; and Koltun, V. 2010. Feature construction for inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, 1342–1350.

Mahler, J.; Liang, J.; Niyaz, S.; Laskey, M.; Doan, R.; Liu, X.; Ojea, J. A.; and Goldberg, K. 2017. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*.

Ng, A. Y.; Russell, S. J.; et al. 2000. Algorithms for inverse reinforcement learning. In *Icml*, 663–670.

OpenAI; :; Andrychowicz, M.; Baker, B.; Chociej, M.; Jozefowicz, R.; McGrew, B.; Pachocki, J.; Petron, A.; Plappert, M.; Powell, G.; Ray, A.; Schneider, J.; Sidor, S.; Tobin, J.; Welinder, P.; Weng, L.; and Zaremba, W. 2018. Learning Dexterous In-Hand Manipulation. *ArXiv e-prints*.

Pimentel, M. A.; Clifton, D. A.; Clifton, L.; and Tarassenko, L. 2014. A review of novelty detection. *Signal Processing* 99:215–249.

Ramakrishnan, R.; Kamar, E.; Dey, D.; Shah, J.; and Horvitz, E. 2018. Discovering blind spots in reinforcement learning. *Proceedings of the 17th international joint conference on Autonomous agents and multiagent systems*.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144. ACM.

Sammut, C.; Hurst, S.; Kedzier, D.; and Michie, D. 1992. Learning to fly. In *Machine Learning Proceedings 1992*. Elsevier. 385–393.

Shah, S.; Dey, D.; Lovett, C.; and Kapoor, A. 2018. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, 621–635. Springer.

Stone, P.; Kaminka, G. A.; Kraus, S.; Rosenschein, J. S.; et al. 2010. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *AAAI*.

Sui, Y.; Gotovos, A.; Burdick, J.; and Krause, A. 2015. Safe exploration for optimization with gaussian processes. In *International Conference on Machine Learning*, 997–1005.

Syed, U.; Bowling, M.; and Schapire, R. E. 2008. Apprenticeship learning using linear programming. In *Proceedings of the 25th international conference on Machine learning*, 1032–1039. ACM.

Taylor, M. E., and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10(Jul):1633–1685.

Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; and Abbeel, P. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint arXiv:1703.06907*.

Urbancic, T., and Bratko, I. 1994. Reconstructing human skill with machine learning. In *Proceedings of the 11th european conference on artificial intelligence*, 498–502. John Wiley & Sons, Inc.

Wray, K. H.; Pineda, L.; and Zilberstein, S. 2016. Hierarchical approach to transfer of control in semi-autonomous systems. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, 1285–1286. International Foundation for Autonomous Agents and Multiagent Systems.

Zhifei, S., and Meng Joo, E. 2012. A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics* 5(3):293–311.

Ziebart, B. D.; Maas, A. L.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, 1433–1438. Chicago, IL, USA.