# Apprenticeship learning and mimicry

Paul Christiano Follow
Oct 3, 2015 · 7 min read

This post compares my recent proposal with Abbeel and Ng 2004, Apprenticeship Learning via Inverse Reinforcement Learning. My hope is to point out how similar the two schemes are, and also to illustrate some practical issues that I've been discussing in the abstract.

## The correspondence

- Hugh : the expert demonstrator
- Arthur : the RL algorithm used in step 4 of the max-margin algorithm
- Eva : the SVM solving the inverse RL problem in step 2.

In each step, Eva produces a classifier that can distinguish Hugh from every strategy identified by Arthur so far. Arthur then searches for a new policy that is classified as "Hugh" by this classifier.

[Abbeel and Ng] call Eva's classifier a "reward function," but they stress that it need not correspond to the actual reward function that we care about. I think that both names are reasonable, since this classifier is intended to converge towards a reward function for "successful human-like performance" or at least some sufficient criterion for such performance. I'll probably stick with the less-loaded word "classifier" anyway.

The correspondence is essentially perfect. The primary differences are simplifying assumptions in [Abbeel and Ng]:

- They assume that the unknown reward function is a linear function of a given set of features, so that they get the desired result even if Eva only considers linear classifiers.
- They assume their RL agent can optimize any linear reward function as well as the human, guaranteeing that Arthur can find a policy that Eva can't distinguish from the expert demonstration.

I'm happy to stick with linear classifiers, but I am especially interested in the case where the reinforcement learning algorithm is not able to exactly reproduce human behavior.

In these domains, I recommend that once Eva has learned a candidate human-recognizing classifier, and once Arthur has failed to fool Eva, Hugh attempts to adjust his behavior so that he continues to achieve his goal without being flagged by the human-recognizer.

## Example

Consider a video game, where we want to to teach an RL agent to play as well as possible without doing anything a human wouldn't. Assume that "score" is one of the features available to Eva.

The scheme in [Abbeel and Ng] will sometimes fail to perform reasonably. For example, suppose that the human sometimes collects a star, which the RL agent cannot learn to do. In this case , Eva might learn the rule "if it collects a star, it's a human." An RL agent trained to get stars may end up with very few points, even if the RL agent is capable of getting scores just as high as the human.

Our proposal is for the human expert to inspect the model learned by Eva—perhaps by just trying to play the game and observing what causes Eva to guess "human," or by using tools for understanding what the learned model is really doing, or so on. Then the human expert attempts to maximize their score while looking like an AI to Eva. In the case of linear models it is plausible that the human can just look directly at the coefficients.

For example, if Eva learns the model "if it collects a star, it's human" then the human player simply needs to stop trying to collect stars. They then train Eva again to learn a different classifier and repeat the process.

In general the problem will be much harder. For example, it may be that the human and AI take different kinds of trajectories through the game world, and it may be difficult for a human to take an AI-like trajectory.

Some of these difficulties can be addressed by using AI assistant to help impersonate the AI player. For example, a human might control an AI avatar, who can carry out a sequence of instructions of the form "move to point X." The resulting trajectories might be able to reproduce much of human performance without looking distinctively

human. Or we might use gradient descent to find a trajectory near the human's which nevertheless fools Eva.

## Playing games

In my post I describe Eva as learning to distinguish Arthur from Hugh. In [Abbeel and Ng], Eva learns to distinguish Hugh from *every* strategy learned by Eva. Why the apparent difference?

In fact this isn't a difference at all. My proposal calls for Eva to learn a classifier which Arthur can't learn to fool. This cannot be done by simply training on the current strategy that Arthur is using, or else the two could continue to go around in a loop indefinitely, with Eva losing at every turn. The strategy given by [Abbeel and Ng] is the most natural way for Eva to solve this learning problem in their setting.

In general, we should think of Eva and Arthur as learning to play a zero sum game against each other—Eva needs to use a learning strategy that is appropriate for playing games. Responding to your opponent's last move is not a good idea in general. However there are almost-as-simple iterative techniques that do work well—for example you can obtain strong guarantees using multiplicative weights, which amounts to (1) train against all of the strategies that have appeared so far, giving them equal weight, (2) use a regularization that rewards you for playing more unpredictably.

Moreover, Eva and Arthur don't inherently have to use iterative strategies, if they are smart enough agents that they can perform sensibly in their first time playing the game. This is a key observation for the scalability of the proposal to very powerful AI's, for whom supervision is more costly and training is expected to generalize further, but who are also more likely to be able to learn to play the game without iteration. (It is also important if we want to train Arthur without ever actually carrying out this dynamic—Arthur still picks a good strategy to play in the game, but the game is only actually played very rarely to provide training data.)

This game framing continues to apply just as well when we introduce Hugh—the game is now a cooperative game, with Arthur and Hugh competing against Eva.

Another nice fact about two player (or two team) games is that we can choose which player moves first without changing the essential character of the game. (This is clear in [Abbeel and Ng]—when Eva

succeeds she has a particular linear classifier that Arthur can't fool, and when Arthur succeeds we can take a convex combination of his strategies that Eva can't distinguish from Arthur's.) Thus Eva can pick her model before Hugh picks his demonstration, if Eva is a clever enough learner to anticipate how Hugh will approach the task in question. This means that Hugh only needs to reason about the particular model that Eva has learned, which he is free to inspect while devising his strategy, rather than thinking about what classifiers Eva is capable of learning.

## Meeting halfway implicitly or explicitly

Even if we don't use an explicit process of "meeting halfway," training a system by imitation learning is likely to involve a similar implicit dynamic—researchers focus on performing the kinds of tasks that available systems can actually accomplish, by means that that available systems can actually imitate.

There may be practical benefits to making this procedure explicit and having an interactive process between the human and the distinguisher; or there may not be. But in either case, making this process explicit is useful from the AI control perspective: it allows us to think in advance about how this process will scale as AI becomes more sophisticated and human involvement becomes more costly, and whether there are any fundamental obstacles that don't appear in existing models of the problem.

One important observation about the formalized process is that it does not inherently require much human interaction. The use of multiple rounds of interaction is necessary only for learners who are too weak to "see where this is going" in advance—for learners strong enough to find human-like ways to accomplish novel tasks, no additional effort is needed to actually provide appropriate demonstrations. This makes the technique a plausible ingredient in scalable AI control.

## Conclusion

Although [Abbeel and Ng] uses reinforcement learning as a technical tool, it directly imitates the expert's behavior. For example, its easy to see that their framework will never achieve a higher performance than the expert, and will reproduce every linearly measurable quirk of the expert's demonstration. Their work nicely illustrates how the *goal* of imitation is compatible with algorithmic techniques and representations based on reward functions.

Many researchers concerned with AI safety in particular have been historically uninterested in this kind of proposal because mimicry seems inherently unscalable beyond human level. While this is intuitively plausible, I think it isn't quite right; because expert demonstrators can themselves make use of AI systems, the resulting performance can radically exceed human capabilities. I suspect that this approach can make full use of whatever AI abilities are available, though for now that is a big open question.

Instead, I think that the key problem with mimicry is that directly learning to imitate is often more challenging than directly accomplishing a goal by other means. Fortunately, this is a challenge that is already being addressed today. Moreover, unlike some other difficulties in AI control, I don't think this problem will change fundamentally or become radically more difficult as AI systems become more capable.