1 Introduction

Today I'll be talking about some of my work in imitation learning that I've been working on for the last four years. This work started when I was at Disney Research, and has taken on new directions.

Let me explain how I got into this area: Behavior modeling. What I mean by that is that rapidly growing availability of tracking data. This includes, for instance, sports data (for instance, basketball). We can track the behavior and positions of all ten players, the referees, the balls, and you can easily annotate the events like shots and balls passed at a rate of 25 Hz. This isn't too hard of a computer vision model since the background is known and there is no clutter. This is commericially economically viable to do.

We have applications where we track people's expressions while they watch movies, the flow of people through Disneyland, biologists who work with lab animals (for instance, 20,000 hours at 100 Hz of fruit flies biting each other), perfectly tracked virtual environments.

This is what I mean by tracking and behavior data, in all flavors, it's very abundant. So what can we do with this data?

(a) Understand the behavior: causal relationships, explain the dynamics behavior of a sequence of decisions

(b) Predictive modeling: Can we predict the next action? This is what imitation learning is.

In this talk, we'll talk about a lot of extensions of the imitation learning paradigm.

2 The Setting

We will be focusing on supervised learning for this talk. Imitation learning, which is learning from demonstrative behavior, is the sequential decision making generalization of conventional supervised learning. We want to learn a mapping to a sequence of actions. The input is a sequence of contexts or states, it gets fed into our policy (what we call the predictor in both imitation and reinforcement learning). This then affects the next state. Given a training set of sequences of demonstrated actions, we want to learn a policy with low error.

The most general formulation of imitation learning can be summarized in the following equation:

$$\text{argmin}_h \mathbb{E}_{s\sim P_h}\left[c_s(h(s))\right]$$

The key difference is that the states depend on the hypothesis. The difference from reinforcement learning is the cost function is not fully observed – so it's like a bandit version of imitation learning. Note that imitation learning is not an i.i.d. setting, since the policy induces the state distribution. The cost function is typically pre-defined.

In practice, people typically use alternating optimization: Alternate over h fixed and distribution fixed. To date, there have been many approaches that fit this paradigm (SEARN

for instance). From the perspective of this talk, we view this pre-existing methods as being very general. The challenge is once you try to apply these methods to particular domains, they may not work. What we will do in this talk is abstract away some structure from a certain group of applications, so that imitation learning methods will converge much faster, sometimes provably so.

3 Concrete Problems

3.1 Camera Control

The first application is camera control. We have a real-time player tracking system, which tries to track players in a basketball game in real time. A human camera operator also exists, and is following the action of the game for public consumption. Given a representation of the context, can we train an autonomous camera policy given the human expert?

The input is a stream of x, e.g. noisy player detections. The state $s = (x, a)$: recent detections and actions. We want to imitate the actions of the human camera. The naive approach is to now supervisedly train at each moment, ignoring the fact that everything is sequential. The main difficulty is that this footage is extremely jittery: You'd get a massive headache. The problem is that it takes an infinite amount of training data to get a smooth principle from just input-output examples. If there's some general principle, you have to bake the assumptions into the model or learn it directly from the data.

Some things people do are post-hoc smoothing: i.e. some nonparametric smoothing method. But, you're ignorant of the external input, and are only smoothing based off the way the camera's pointing (Kalman filters, autoregressive, etc), and so you undersmooth and oversmooth depending on where you are. People engineer this to be better, but maybe there's a better way to go about this.

You can't rely 100% on learning. Black box assumptions are assumption free, underspecified, need lots of examples, and the smoothign approaches are difficult in their own way. How to combine?

Our policy goes to a black box predictor (neural net) or a smooth model (parametric autoregressor or Kalman filter). How to interpolate between the two? We have $h(s = (x, a)) = \text{argmin}_a(f(s) = a')^2 + \lambda(g(a) - a')^2$. Basically, we want the smooth model to control most of the time, but the neural net to course correct from time to time. It's a form of regularization on the *dynamics* of the output on the neural net. So it only makes sense in the sequential decision making domain.

We also developed a new learning approach in a provably efficient way, assuming the neural net can be trained well (ERM reduction). We do a quantitative comparison, and also a qualitative one, where you can superimpose over the real footage and compare. First time a machine learning paper appeared on Sports Illustrated!

If you want to use multiple cameras, the action space blows up with the number of cameras, so you may need to do something clever.

The main challenge with HMM style approaches is they measure complexity by the Markovian property: How many time steps into the past do you condition on? No assumptions about the model in that time. So in the worst case, the model blows up exponentially with the Markovian assumption. We are controlling the complexity a bit differently if we use a neural net, we look 10-20 steps into the past. One of the benefits of our approach is that the definition of smoothness is explicit here. LSTMs don't explicitly model smoothness, though it seems LSTMs may be more biased towards smoothness based on their architecture. Though we compare it to our model, LSTMs are not as smooth. You can plug in play with our algorithm actually, it's a modular black box. Our code is online. We can also do an analysis (if you can learn black box, like assuming you can learn neural net) of convergence guarantees. By explicitly acknowledging there is structure in the output space we care about, you can learn faster. The way we can prove convergence given an ERM reduction to deep learning is if we first learn the smooth part, and then do deep learning on the residuals. If you do it the other way around, intuitively it'll probably work most of the time, but theoretically I can't say anything. This is related to posterior inference regularization. Posterior inference at runtime, and then regularizing afterward to some posterior inference. One difference here is we are doing it sequentially.

So then, I thought about other areas which exploit model based approaches. Can we add a layer of flexibility using say deep learning (like in optimal control for instance). Can we combine black box learning with model based optimal control.

3.2 Speech Animation

We were talking to people at Pixar and Walt Disney animation. They spent over 50% of the time animating the face. The vast majority is on the eyes or the mouth. If you get that wrong, you get into the uncanny valley, and things look creepy. For this application, we looked at hte mouth. How do we do data-driven lipsynching very accurately? Co-articulation challenge. How is your mouth shaped to enunciate a phone? It depends on the surrounding phones. This is hard. You can't just do Bezier interpolation between images of a mouth for each phone. There are 44 unique phones in English, 200 in global.

We have developed an approach to solving this. We get an input sequence of phones (phonetics, not audio), then parametrizations of a mouth model using computer graphics (active appearance model, 30 degrees of freedom). Given a sequence of phones, we want to predict a sequence of configurations of the mouth. Frame-by-frame decomposition of saying the word prediction can be visualized. If you frame the problem correctly, it's simple to learn it well. A few observations:

(a) The temporal curvature of the output can vary smoothly or sharply (This is the co-articulation problem).

(b) In this particular problem, there are basically no long term dependencies. We need a predictor that can nail seemingly arbitrary types of predictions in a local window, and can ignore anything global.

Our solution is basically an overlapping sliding window predictor. We feed each of the sequence into some black blox predictor (window of size 5), so it's a $30*5$ dimensional regression problem. Since we are mapping phonetics to animation, we are language-independent.

3.3 Hierarchical Behaviors

We focused on different policy classes for modeling behavior data. When humans do decision making or taking sequences of actions, there are short term goals and long term goals. You could think of a long term goal in basketball as getting to the basket, and the short term goal as a sequence of short steps to get to that goal. The methods are instantiated with deep neural nets, but random forests work as well. We have been looking at heirarchical generative models as well. In a relatively contrived user study, we compared against deep generative models that don't do this hierarchy and also against the ground truth. So we beat all baselines convincingly, and about competitive with the ground truth. This was recently published at NIPS.

We also do this with lab animals, we work with fruit flies. We try to learn the behaviors of the flies. Didn't pick up the behavior of males following females to mate. Our model is not explicitly modeling multi-modal behavior, which may be a problem.

There is a difference between learning a function and learning probabilistic model over things. Sample the output rather than compute the argmax. Trying to model the shape of the whole distribution seems harder.

By hierarchical, we are modeling the hierarchy jointly across individual actors.

Why use machine learning for the fruit fly problem? We give representations of the behaviors to the scientists, and they interpret the model in order to help guide their further scientific inquiry: For instance, learning about other behaviors of the flies.

3.4 Multi-agent systems

A popular technique in sports analytics is called "ghosting": The coaches point out where the defenders should have been, to help coach the defense better. This is manually intensive, so automating it is of interest. Can we automatically ghost replays. The technical challenge, from imitation learning perspective, is learning a coordination mechanism. The final predictor in most cases is a deep neural net or a random forest. But to implement a deep net in Tensorflow, you need a semantically consistent input representation. Unfortunately, the data we get is an unstructured set of trajectories. And morevoer, people swap roles dynamically! You might not have people have the same roles over time. So can we infer the roles automatically? Latent variable permutation problem. What happens if you ignore this? It is complete garbage. To address this challenge, my student proposed a more meta-learning framework: Standard imitation learnign with latent variable inference, and then train jointly using stochastic variational inference so it's differentiable end-to-end. We are doing inference on the role assignments. We are only doing assignments: More than one player can be assigned the same role. Permutation models (graphical models) are very

complicated and don't simplify the problem.

3.5 Learning to Optimize

This is imitation learning applied to computational systems. Here I will focus on combinatorial optimization: integer programming, satisfiability, submodular optimization, etc. These are typically solved via local search heuristics. We have oracle based methods (some expensive oracle to solve subroutine of the problem), and learning how to query them intelligently can be thought of as learning problem. Branch-and-bound, searching over state space, can lead to long search time and low quality solutions. An example of the first case: submodular optimization, where evaluating the submodular function is expensive.

We take the perspective that these solvers are sequential (greedy algs for submodular opt). We can think of solver as an agent, and states are potential solutions. We want to find good reward (good solution). For example, trajectory prediction: We want to quickly identify successful trajectories. For instance, robotic arm grasping. Ideally you want high probability of succeeding, as well as diverse policies for some definition of diverse. Submodular functions are natural way to define diversity: What might be a redundant test depends on obstacles in the scene: things are context-dependent. We measure diversity using neural net.

The goal is to imitate the greedy algorithm, based only on features so we don't have to call oracle over and over again. This paper is "Learning polices for contextual submodular prediction".

3.6 Ongoing Research

We have an ongoing project with JPL: hard risk constraints for Mars rover. The constraints become NP hard, becomes a mixed integer program which is hard to solve efficiently. The goal is to learn a statistical model to make better branch and bound decisions. The thing I want to highlight is that in terms of number of nodes expanded in the search space, we are significantly more efficient than the commercial solver. So we can find new solutions, and learn heuristics that do well on specific hard problems.

4 Takeaways

Hopefully I painted a convincing picture that imitation learning is a fruitful new research direction, due to the amount of data. There are new frontiers focusing on structure: smoothness, permutation structure, etc. We can design new frameworks for solving these specific problem. For instance, combinations of smoothness and black box predictors, or latent graphical models + prediction.