CS 280 Computer Vision
Spring 2018 Agrawal Lecture 27

1 Lecture

Consider a self-driving car. It takes in sensory input (e.g. LIDAR data) and has to decide how to move. And we want to give it some semantic understanding of its sensory input, even in unprecedented scenarios.

General problem: given sensory input, what action should our system take?

- *Approach #1 to problem*: behavioral cloning.
- *Approach #2 to problem*: inference from physics (trajectory optimization).
- *Approach #3 to problem*: reinforcement learning.

An issue in classical model-based control: we might want to extract information from an image (using *computer vision*) to pass to our physics model. However, the kind of data we'd want to use as physics input (mass, friction, etc.) is very hard to estimate using computer vision.

In an RL setting:

- Humans invoke prior knowledge during exploration.
- We can try to get RL agents to learn these priors from experience.

Learning a Feature Space

A *forward model in pixel space*: one that takes a state (as an image) and an action, and predicts the next state (i.e. all of the pixels in the next image). We can use such a model to infer which action to take.

- However, it's often not necessary to predict in exact detail all of the pixels.
- We can instead embed the image in a feature space as z (some abstract representation of the image).
- Concretely, we have a network $\phi(x; \theta_z)$ which takes an image x and outputs a latent representation z.
- We also have an action prediction model $f(z_t, z_{t+1}; \theta_f)$ which takes $z_t = \phi(x_t)$ and $z_{t+1} = \phi(x_{t+1})$ and outputs an estimate $\hat{a}_t$ of the actual action a_t we take.
- We would like to minimize the loss $\mathcal{L}(\hat{a}_t, a_t)$ as

$$\min_{\theta_z, \theta_g} \mathcal{L}(\hat{a}_t, a_t)$$

  This is an inverse model because it goes from images to actions.

- As demonstrated by prior paper(s), the learned feature space encodes things which are useful for image recognition and pose matching. So we can learn a useful feature space through this kind of proxy task.

In computer vision, this is known as **self-supervision**: learning features from an auxiliary task. We don't require an external agent to provide a task; we can come up with our own task that doesn't require humans to label data yet still allows us to learn useful feature spaces.

Evidently, we can use a joint forward/inverse model for our tasks, and avoid having to make predictions in pixel space. Here, the inverse model helps us learn some kind of features, and the forward model makes predictions in that latent feature space.

By building a joint forward/inverse model, agents may be able to learn a useful model of interactions.

Curiosity-Driven Exploration

Random exploration is limiting. If we take entirely random actions in a state space, we'll probably only end up exploring a small part of it (assuming a fixed time interval). In exploring, the agent's goal should be to seek novelty, so that it can learn about more and more things. The agent should take actions that help it move to new parts of the environment.

Say that we adhere to the forward model approach. As an agent takes actions, it is trying to build a model that will predict what happens next. Suppose an agent takes an action which goes beyond anything it's ever seen before. Presumably, this will lead to a higher prediction error. So if we incentivize our agent to take actions which lead to high prediction error, it will move to unseen parts of the environment. And it will theoretically manage to explore more and more.

The agent should be curious about taking actions which lead to higher prediction error, so that it ends up exploring lots of new things. In particular, the agent should be curious about things that can affect it.

So we can generate a curiosity reward r_t^i via our forward model (note that the reward doesn't come from the environment anymore) and run RL to create an exploration policy which maximizes this reward.

The policy should be learned using reward in feature space, not pixel space,
so that it can ignore irrelevant stuff and transfer better.

The result: an agent which explores well despite not having any reward from the environment.

When it goes to new environments, it can also explore more quickly and generally do better.
It learns from experience, and transfers knowledge to do well in slightly different environments!