State Construction

Markov and Agent State

Sept 5 and 6, 2019

A *Markov state* is an incrementally updatable summary of the past that is informationally sufficient to predict the next observation, and thus to predict anything, as well as could be predicted from the complete past. We now detail and formalize each part of this verbal definition.

The "past" at time t is a *history*, a trajectory of the preceding actions and observations:

$$H_t \doteq A_0, O_1, A_1, O_2, \ldots, A_{t-1}, O_t. \tag{1}$$

Let $\mathscr{H} = \{\mathscr{A} \times \mathscr{O}\}^*$ be the set of all histories of all lengths. A general environment is (of course) a properly specified probability distribution over histories.

For a state to be a "summary of the past" means that it is a function of the history. Let that function be denoted $f : \mathscr{H} \to \mathscr{S}$, where the set $\mathscr{S}$ is an arbitrary set of states.

To be "incrementally updatable" means that there exists a state-update function $u : \mathscr{S} \times \mathscr{A} \times \mathscr{O} \to \mathscr{S}$ such that

$$f(hao) = u(f(h), a, o), \qquad \forall h \in \mathscr{H}, a \in \mathscr{A}, o \in \mathscr{O}. \tag{2}$$

Note that every u completely specifies a corresponding f, but of course not every f has a corresponding u. We are ultimately interested only in updatable fs, in other words, in us.

To be "informationally sufficient to predict the next observation" means that any two histories, h and h', that are mapped by f to the same state, also have the same probability distributions over their next observations:

$$f(h) = f(h') \implies \Pr\{O_{t+1} = o \mid H_t = h, A_t = a\} = \Pr\{O_{t+1} = o \mid H_t = h', A_t = a\}, \tag{3}$$

for all $o \in \mathscr{O}$ and $a \in \mathscr{A}$. It follows that a Markov state, defined in this way, is informationally sufficient to predict not only the next observation, but *anything* about the future. This follows because knowing how likely each next observation is means also knowing how likely each next state is, by applying u. Repeating this one knows the probability of each possible sequence of observations conditional on action, which is to know everything about the environment. It also follows that a Markov state is informationally sufficient for selecting actions to achieve any goal expressed in terms of the observations.

Is it possible for a function f to make all the one step predictions correctly but not be incrementally updatable? In other words, are there fs that satisfy (3) but not (2)? When I was writing the last chapter of the reinforcement learning book I thought not, but Shibhansh points out that there are. Basically all you need is an MDP where you need to remember something, but you don't use it right away. Here is a simple example with three observations denoted 1, 2, and 3 (and no actions). If your last observation was a 1 or a 2, then the next observation is 1, 2, or 3 with equal probability, but if your last observation was a 3, then your next observation is the same as the observation that preceded the 3. So, a history might be:

…1121312322131112131311 2

For this MDP, to predict the next observation after a 1 or a 2, you don't need to remember which it was—the next-step predictions are the same—but after you see a 3 you need to have remembered the last 1 or 2 in order to get the next-step predictions right after the 3. Specifically, for this example the "state" set could be {*1next, 2next, anynext*}. A function f could then by defined by

$$f(*1) = f(*2) = \textit{anynext}$$

$$f(*13) = \textit{1next}$$

$$f(*23) = \textit{2next}$$

This f is sufficient for the next-step predictions (it satisfies (3)) but is not updatable (cannot satisfy (2)). If you are in state *anynext* and observe 3, then you can't know if the next state should be *1next* or *2next* from *anynext* alone; the needed information has been lost, and thus no state-update function is possible.

Thus, (3) is not sufficient for an f to determine a Markov state. We also need (2).

—

The *agent* state is the agent's approximation to a Markov state. It is an incrementally updatable summary of the past that the agent uses to predict and control its future but, as an approximation, it is not necessarily informationally sufficient for predicting or controlling anything perfectly. The requirement of informational sufficiency for complete accuracy and optimality is softened to informational and computational *utility*, with accuracy and optimality balanced against resources, for predicting and controlling whatever the agent seeks to predict and control.

This is a good definition of (agent) "state". State is what the agent needs to maintain as it rolls forward in time. Presumably it will be a real-valued feature vector. Its components will be state features or, more generally, state variables. The state-update function u, which determines f, is a recurrent process that produces the state-feature vector. Prediction and control decisions will be made as a function of the state vector…as described in the State-Constructing Recurrent Network.

A feature being a *state* feature need not be a hard or absolute thing, and there may be several subcategories or classes of stateness within the broad definition given here.

In planning in particular, it may be useful to explicitly distinguish state variables that are used directly to make predictions from those that are used only to maintain state (i.e., by the state-update function). I often say that "environmental models should be *state-to-state*", meaning that they take states as inputs and produce states (or state distributions) as outputs (in addition to the option inputs and the reward-along-the-way outputs). But, if there are some state variables used only to update state, then those variables are not needed as inputs or outputs of the model. They are not needed in casts (the operation of looking ahead to consequences using a model). This is true even in the most general case in which models are *composed* (i.e., the state output from one model is used as input to another model). Models and casts only use states after they have been created by state update. They are not at all concerned with how the states are created or maintained. Thus they can ignore state variables that are used solely for state update. This is absolute, almost trivial. If no model or approximate value function uses a state variable as input, then no model need predict that state variable as output.

Going further, recall that I have proposed a particular, extreme style of planning consisting entirely of one-cast jumps followed by an approximate value function, in other words, single expected backups. In such planning operations, as in value iteration, models are never composed—models are combined only with the approximated value function, never with other models. In this case, models need only produce the subset of the state variables that participate in the approximate value function (or that sometimes participate in the approximate value function, as we have come to assume that the approximate value function changes often).

Another class are the candidates state features. These are the features that do not as yet succeed in participating in predicting or controlling anything, but nevertheless exist so that they can be tested to see if they could help in predicting or controlling something.