Psych 128/290Q: Probabilistic models of cognition
April 28: Reinforcement learning

Our discussion of learning so far has paid relatively little attention to the costs of errors and rewards associated with success. Both supervised and unsupervised learning have an implicit expectation that getting the correct answer or identifying appropriate latent structure are desirable, and this assumption can be made explicit by formally defining a set actions an agent might take and the utility associated with those actions. However, thinking about these rewards also leads us to another way of casting learning problems, in which the goal of the agent is simply to maximize reward. In these *reinforcement learning* problems, learning is necessary in order to establish which patterns of behavior are likely to lead to reward.

Formalizing reinforcement learning

The basic reinforcement learning model assumes that there exists a discrete set of states of the environment $\mathcal{S}$, a discrete state of actions an agent can take in that environment $\mathcal{A}$, and a set of reinforcement signals (including rewards and penalties, represented by positive and negative real numbers) probabilistically associated with those states and actions. This model can be augmented with assumptions about whether the agent perceives the state of the environment directly, or receives noisy signals about that state (known as *partial observability*). The goal of the agent is to find a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ (ie. a function mapping states to actions) that maximizes long-term reward. While there are different ways of defining what long-term reward means, one standard approach is to use the *expected discounted infinite horizon* reward, given by

$$E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \tag{1}$$

where r_t is the reward at time t and the expectation is with respect to the values of r_t when pursuing the policy π.

The long-term reward given by pursuing a policy π clearly depends on the joint distribution of states and the influence of actions on those states. In principle, both the distribution and the influence of actions can be quite complicated, with states depending on other states and actions arbitrarily far in the past. Making headway on solving the reinforcement learning problem thus usually requires making some simplifying assumptions. One standard assumption is that the environment in which the agent is operating has a Markov property, with the probability distribution over the next state depending only on the current state and the current action. The joint distribution on states given actions can thus be found by multiplying together a sequence of distributions of the form $p(s_{t+1}|s_t, a_t)$, where s_t and a_t are the state and action at time t respectively. If we add the assumption that r_t depends only on s_t and a_t, the result is a *Markov decision process* (MDP). The MDP is fully specified by the states $\mathcal{S}$, actions $\mathcal{A}$, reward function $R(s, a)$, and state transition function $T(s, a, s') = p(s_{t+1} = s|s_t = s', a_t = a)$.

Solving a MDP means finding the policy π that maximizes the long-run reward, as given in Equation 1. Characterizing the solution is made easier by defining the value of a state s to be

$$V^*(s) = \max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t t_t\right] \tag{2}$$

where we assume that the learner starts in state s and thus chooses a policy π beginning with that state. This can also be expressed recursively, with

$$V^*(s) = \max_{a}\left(R(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s')V^*(s')\right) \tag{3}$$

making it clear that the value of the state s is related to the value of the states s' reached by taking the best

action a. This is known as a Bellman equation. The optimal policy is then

$$\pi^*(s) = \arg\max_a \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') V^*(s') \right) \tag{4}$$

which can be computed directly from the reward function $R(s,a)$, the state transition function $T(s,a,s')$ and the value function $V^*(s)$. We can thus solve any MDP if we can estimate these three functions.

Finding an optimal policy in a known model

If we know $R(s,a)$ and $T(s,a,s')$, we can estimate the value function by using a procedure known as *value iteration*. Defining $Q(s,a)$ to be the total long-run reward associated with taking action a in state s, we have

$$Q(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') V(s'). \tag{5}$$

Value iteration iterates between estimating $Q(s,a)$ using the current estimate of $V(s)$ (by applying this equation) and estimating $V(s)$ using the current estimate of $Q(s,a)$ (by taking $V(s) = \max_a Q(s,a)$). This procedure will converge to the true function $V(s)$, and can consequently be used to identify the optimal policy.

An alternative approach is trying to estimate the optimal policy directly. The *policy iteration* algorithm starts with an arbitrary policy, then computes the value of each state under that policy by solving the equations

$$V_\pi(s) = R(s,\pi(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s,\pi(s),s') V_\pi(s') \tag{6}$$

for $V_\pi(s)$. The resulting value function can be used to compute $Q(s,a)$ for each state-action pair (substituting V_π for V in Equation 5), and then the policy can be updated such that $\pi(s) = \arg\max_a Q(s,a)$. This procedure can be iterated until it reaches a fixed point. Since the policy improves at every iteration, the resulting policy is optimal.

Learning an optimal policy in an unknown model

Value iteration and policy iteration provide ways to identify an optimal policy when the model of the environment is known. However, a large part of the challenge of reinforcement learning is dealing with situations where $R(s,a)$ and $T(s,a,s')$ are unknown, and learning an optimal policy in these cases is an ongoing topic of research. Current approaches can be divided into *model free* approaches, which aim to make it possible to estimate the value function without estimating $R(s,a)$ and $T(s,a,s')$, and *model-based* approaches, which seek to build a model of the environment from which these functions can be identified. Model-based approaches largely reduce to the kind of unsupervised learning problems we have already discussed in detail, so the focus here will be on model-free approaches.

Model-free approaches typically define simple learning rules that can be used by a learner to estimate the quantities required to identify an optimal policy without needing to build a full model of their environment. One such algorithm is *temporal difference (TD) learning*, in which the estimated value of a state is updated each time the learner visits that state and receives a reward. The update rule is

$$V(s) = V(s) + \alpha(r + \gamma V(s') - V(s)) \tag{7}$$

where s' is the state reached by the agent after taking an action a from state s, and r is the reward associated with that action. Intuitively, this can be justified by viewing s' as a sample from the distribution defined by $T(s,a,s')$. More formally, we can observe that if we assume we are taking the optimal action given our current beliefs, then

$$V(s) = r + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') V(s') \tag{8}$$

where r is the reward provided by that action. This equation should hold provided our estimate of $V(s)$ is accurate. If this is not the case, then subtracting $V(s)$ from the right hand side gives us a kind of error signal, indicating whether $V(s)$ is too high or too low based on the values of the other states. Denoting this error signal δ, we have

$$\delta = r + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s')V(s') - V(s) \tag{9}$$

which justifies the simple learning rule

$$V(s) = V(s) + \alpha\delta \tag{10}$$

where α is a learning rate. However, this still requires us to know $T(s, a, s')$. We obtain the model-free TD learning rule by treating the state s' as a sample from the distribution associated with $T(s, a, s')$ – a single sample Monte Carlo approximation to the expectation in Equation 9. Provided enough iterations of learning are performed, and the learning rate is decreased appropriately over time, the algorithm will converge to a correct estimation of $V(s)$.

Another model-free approach is Q learning, in which we seek to estimate the function $Q(s, a)$ instead of $V(s)$ directly. Substituting the definition of $V(s')$ into Equation 5, we have

$$Q(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s') \max_{a'} Q(s', a') \tag{11}$$

which gives a recursive definition of $Q(s, a)$. Using a similar argument to that given for $V(s)$ above, we can define a simple learning rule for estimating $Q(s, a)$ based on how well it satisfies these equations. Specifically, we have

$$Q(s, a) = Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a)) \tag{12}$$

where α is once again a learning rate. Convergence to the correct $Q(s, a)$ will occur as the number of iterations increase, provided α is decreased appropriately over time.