CS294-40 Learning for Robotics and Control Lecture 11 - 10/02/2008

Bandits

Lecturer: Pieter Abbeel *Scribe: David Nachum*

Lecture outline

1. Multi-armed bandits
2. Optimal stopping
3. Put it together (use 2 to solve 1)

1 Multi-armed bandits

Consider slot machines H_1,H_2,...,H_n.

Slot machine i has pay-off $= \begin{cases} 0, & \text{with probability } 1-\theta_i \\ 1, & \text{with probability } \theta_i \end{cases}$

where θ_i is unknown.

Now the objective to maximize is:

$$E[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)] \text{ (where } s_t \text{ is unchanged).}$$

The most natural way to turn this into an MDP is by using "information" states:

$$\begin{pmatrix} \#\text{ of successes on } H_1 \\ \#\text{ of failures on } H_1 \\ \#\text{ of successes on } H_2 \\ \#\text{ of failures on } H_2 \\ \vdots \\ \#\text{ of successes on } H_n \\ \#\text{ of failures on } H_n \end{pmatrix}$$

Note that the state space grows with the number of slot machines and with the number of moves that have been made. However, we will show that the problem can be decomposed in a way that avoids the requirement to compute with the joint state space over all bandits; rather it allows us to solve optimal stopping problems for each bandit independently; from this, it will have sufficient information to act optimally in the original problem (for which the state space grows exponentially in the number of bandits).

2 Optimal stopping

We define "Semi-MDP" as MDP's in which we have some sequence of times at which we change state, and these are the only times we pay attention to.

The transition model is now represented as follows:

$$P(s_{t_{k+1}} = s', t_{k+1} = t_k + \Delta | s_k = s, a_k = a)$$

and we seek to maximize

$$E[\sum_{k=0}^{\infty} \gamma^{t_k} R(s_{t_{k+1}}, \Delta_k, s_{t_k})].$$

Here the Bellman update is:

$$V(s) = \max_a \sum_{s',\Delta} P(s', \Delta | s, a)[R(s, \Delta, a, s') + \gamma^{\Delta} V(s')]$$

A specialized version of the Semi-MDP is is the "Optimal stopping problem". At each of the times, we have two choices:

1. continue
2. stop and accumulate reward g for current time and for all future times.

The optimal stopping problem has the following Bellman update:

$$V(s) = \max\{\sum_{s',\Delta} P(s', \Delta | s)[R(s, \Delta, s') + \gamma^{\Delta} V(s')], \frac{g}{1-\gamma})$$

For a fixed g, we can iterate the Belmman updates to find the value function (and optimal policy), and this allows us to determine whether a state's optimal policy is stopping.

Our challenge is to find for each state the minimal g required to make the optimal policy stop in that state. I.e., we are interested in finding:

$$g^*(s) = \min\{g | \frac{g}{1-\gamma} \geq \max_{\tau} \mathbb{E}_{\tau}[\sum_{k=0}^{\tau-1} \gamma^{t_k} R(s_{t_k}, \Delta_k, s_{tk+1}) + \sum_{t=\tau}^{\infty} \gamma^t g]$$

where τ is the stopping time. Note τ is a random variable that which encodes the stopping policy: concretely, any stopping policy can be represented as a set of states in which we decide to stop (and we continue in the other states). The random variable τ takes on the value corresponding to the first time when we visit a state in the stopping set.

We define a "reward rate" $r(s_{t_k}, \Delta_{t_k}, s_{t_{k+1}})$ such that

$$\sum_{t=0}^{\Delta_{t_k}-1} \gamma^t r(s_{t_k}, \Delta_{t_k}, s_{t_{k+1}}) = R(s_{t_k}, \Delta_k, s_{t_{k+1}})$$

This allows us to compute the expected reward rate for states:

$$\bar{r}(s) = \mathbb{E}_{\Delta,s'}[r(s, \Delta, s')] = \sum_{\Delta,s'} P(s', \Delta | s) r(s, \Delta, s')$$

The expected reward rate allows us to compare reward obtained in different states.

Now consider

$$s^* = \arg\max_s \bar{r}(s).$$

Of all states, the state s^* would require the highest payoff to be willing to stop. Namely,

$$g^*(s^*) = \bar{r}(s^*)$$

This means that when $g < g^*(s^*)$, the optimal stopping policy will choose to continue at s^*.
Note that for $s \neq s^*$, $g^*(s) < g^*(s^*)$.

To compute $g^*(s)$ for the other states, we consider a new semi-MDP which differs from the existing one only in that we always continue when at state s^*. This is equivalent to letting the new state space $\bar{\mathcal{S}} = \mathcal{S} \backslash \{s^*\}$. We run this algorithm:

$while |\bar{\mathcal{S}}| > 0:$

$\mathcal{S} \leftarrow \mathcal{S} \backslash \{s^*\}$

Adjust the transition model and the reward function accordingly—namely, assuming we always continue when visiting state s^*.

Compute reward rates$r$ by solving: $\sum_{t=0}^{\Delta-1} \gamma^t \bar{r}(s, \Delta, s') = \bar{R}(s, \Delta, s')$

Compute expected reward rates $\bar{r}(s) = \mathbb{E}_{\Delta, s'} r(s, \Delta, s')$.

$s^* \leftarrow \arg\max_s \bar{r}(s)$

$g^*[s^*] \leftarrow \bar{r}(s^*)$

When the algorithm terminates, we have $g^*[s]\ \forall s$ (since each s was the s^* at some point).

3 Solving the multi-armed bandit over M_1,M_2,...,M_n

We can now solve the multi-armed bandit as follows.

1. Find the optimal stopping cost $g^{(i)}(\cdot)$ for each state for each of the bandits M_i.
2. When asked to act at time t, pull an arm i such that $i \in \arg\max_i g^{*(i)}(s_t^{(i)})$.

3.1 Key properties/requirements

1. Reward at time t only depends on state M_i at time t.
2. When pulling M_i, only state of M_i changes.

3.2 Observation

Assume you are given an MDP with unknown $P(s_{t+1}|s_t, a_t)$, and unknown $R(s_t, a_t, s_{t+1})$. Assume $\exists s_0 \in s$: $\forall$ policies π: with probability 1, we visit s_0 after a finite number of steps.

In this case bandits are equivalent to policies. [However, solving the problem by treating each policy as a bandit will typically not render the most efficient solution, as it does not share any data between evaluations of different policies.]

3.3 Example: Simple queuing problem

Consider the model in Figure 1.
Cost at time $t =$

$$\sum_{i=1}^{n} c_i s_i$$

where $c_i =$ cost per customer in queue i, $s_i =$ number of customers in queue i.
Pay-off for serving queue i (ie the cost we avoid paying by serving a customer in queue i) $=$

$$\sum_{t=0}^{\infty} \gamma^t c_i s_i$$

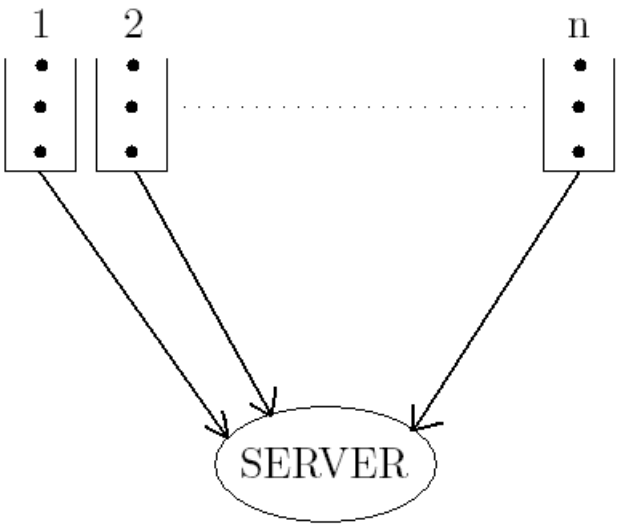

Figure 1: n queues feeding in to one server.

3.4 Example: Cashier's nightmare

Same as queuing problem above, but now the people return to a queue after finishing with the server/cashier. We have a probability distribution $P(j|i)$ which describes the transitions between queues.

We can turn this into a multi-armed bandit problem by considering each customer to be a bandit. The pay-off corresponds to the difference in cost between the current queue and the queue the customer transitions to.