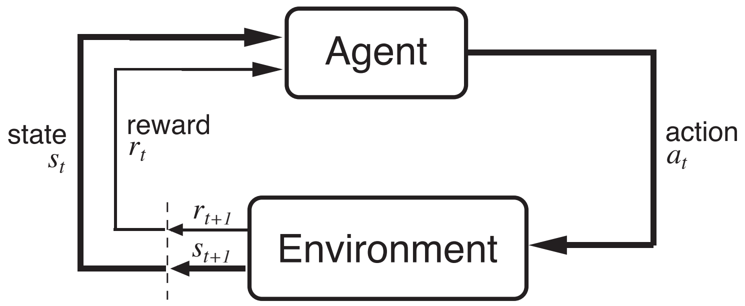
Lecture 17: Reinforcement learning (II)

- Markov Decision Processes
- Bellman equations for policy evaluation
- Model-based learning
- Temporal-Difference (TD) learning
- Eligibility traces

Recall: Reinforcement Learning Problem



- At each discrete time t, the agent (learning system) observes state $s_t \in S$ and chooses action $a_t \in A$
- Then it receives an immediate reward r_{t+1} and the state changes to s_{t+1}

Recall: Policies and value functions

- The goal is to learn a *policy* $\pi : S \times A \rightarrow [0, 1]$ which has a high value at all states.
- One way to attack the problem is to search through policy space, so we need to be able to evaluate a given policy
- We will focus on *discounted return*, so the value of a state s is defined as:

$$V^{\pi}(s) = E_{\pi}\left[r_1 + \gamma r_2 + \ldots | s_0 = s\right]$$

where $\gamma \in (0, 1)$ is the probability of terminating at every step.

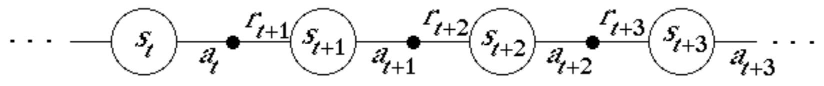
Recall: Monte Carlo estimation

- Monte Carlo methods are supervised learning methods for evaluating a policy
- The state is the input, the return obtained on trajectories obtained using π is the desired output
- Function approximation is used to solve the regression problem.
- Choices include linear function approximators, discretization, nearest neighbor, neural nets, ...
- Problem: variance of returns could be very high
- Can we use environment structure to lower the variance?

Markov Decision Processes

- A general framework for non-linear optimal control, extensively studied since the 1950s
- In optimal control
  - Specializes to Riccati equations for linear systems
  - Hamilton-Jacobi-Bellman equations for continuous-time
- In operations research
  - Planning, scheduling, logistics, inventory control
  - Sequential design of experiments
  - Finance, marketing, queuing and telecommunications
- In Artificial intelligence (last 15 years)
  - Probabilistic planning

Markov Decision Processes (MDPs)



- Set of **states** S
- Set of **actions** $A(s)$ available in each state s
- *Markov assumption:* s_{t+1} and r_{t+1} depend only on s_t, a_t and not on anything that happened before t
- **Rewards**:

$$r_s^a = E\left\{r_{t+1} | s_t = s, a_t = a\right\}$$

- **Transition probabilities**

$$P_{ss'}^a = P\left(s_{t+1} = s' | s_t = s, a_t = a\right)$$

- Rewards and transition probabilities form the **model** of the MDP

Value function for a policy in an MDP

- A **trajectory** ξ is a sequence of states, actions and rewards: $s_0, a_0, r_1, s_1, a_1, s_2, \ldots$
- Let $R(\xi) = r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots$ be the return obtained on trajectory ξ
- The value of a state s is defined as:

$$\begin{aligned} V^\pi(s) &= E_\pi\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots | s_0 = s\right] \\ &= \sum_{\xi : s_0 = s} P^\pi(\xi) R(\xi) \\ &= \sum_{a_0, s_1, \ldots} [\pi(s, a_0) P^{a_0}_{ss_1} P^\pi(\xi_1)] [r_1 + \gamma R_1(\xi_1)] \end{aligned}$$

where $\xi_1 = s_1, a_1, r_2, s_2, \ldots$

Value function for a policy in an MDP (II)

$$\begin{aligned} V^\pi(s) &= \sum_{a_0, s_1, \ldots} \left[\pi(s, a_0) P^{a_0}_{ss_1} P^\pi(\xi_1)\right] [r_1 + \gamma R_1(\xi_1)] \\ &= \sum_{a_0, \xi_1} \left[\pi(s, a_0) P^{a_0}_{ss_1} P^\pi(\xi_1)\right] r_1 \\ &+ \sum_{a_0, s_1} \sum_{\xi_1 : s_1} \left[\pi(s, a_0) P^{a_0}_{ss_1} P^\pi(\xi_1)\right] \gamma R_1(\xi_1) \\ &= \sum_a \pi(s, a) r^a_s \sum_{\xi_1} P^{a_0}_{ss_1} P^\pi(\xi_1) \\ &+ \gamma \sum_a \sum_{s'} \sum_{\xi_1 : s_1 = s'} \pi(s, a) P^a_{ss'} P^\pi(\xi_1) R_1(\xi_1) \\ &= \sum_a \pi(s, a) r^a_s \cdot 1 + \gamma \sum_a \sum_{s'} \pi(s, a) P^a_{ss'} \sum_{\xi_1 : s_1 = s'} P^\pi(\xi_1) R_1(\xi_1) \\ &= \sum_a \pi(s, a) r^a_s + \gamma \sum_a \sum_{s'} \pi(s, a) P^a_{ss'} V^\pi(s') \text{ (def. in reverse)} \end{aligned}$$

Bellman equations for policy evaluation

- The Bellman equations relate the value of a state to the value of its successor states:

$$V^\pi(s) = \sum_a \pi(s,a) r_s^a + \gamma \sum_{s'} P_{ss'}^a V^\pi(s'), \forall s$$

- This is a system of $|S|$ linear equations with $|S|$ unknowns (the values of each state)
- If the state set S and action set A are finite, we can represent the system above in matrix-vector form
  - The value function $\mathbf{V}^\pi$ is a vector with $|S|$ elements
  - The reward model $\mathbf{r}$ is a vector of size $|S||A|$
  - The transition model $\mathbf{P}$ is an $|S||A| \times |S|$ matrix
  - The policy can be written as an $|S|$ by $|S||A|$ matrix $\mathbf{\Pi}$ such that $\mathbf{\Pi}_{s,(s,a)} = \pi(s,a), \forall s, a$ and all other elements are 0.

Bellman equations for policy evaluation (II)

- Let $\mathbf{r}^\pi = \mathbf{\Pi}\mathbf{r}$ and $\mathbf{P}^\pi = \mathbf{P}\mathbf{\Pi}$
- We can re-write the Bellman equations in vector form as:

$$\mathbf{V}^\pi = \mathbf{r}^\pi + \gamma \mathbf{P}^\pi \mathbf{V}^\pi$$

- Now we can solve the system of equations:

$$(\mathbf{I} - \gamma \mathbf{P}^\pi)\mathbf{V}^\pi = \mathbf{r}^\pi$$

- If $\gamma < 1$, because $\mathbf{P}^\pi$ is a stochastic matrix (each row sums to 1, and all elements are positive), we can show that the determinant of $\mathbf{I} - \gamma \mathbf{P}^\pi$ is non-zero
- Hence, the system has a unique solution:

$$\mathbf{V}^\pi = (\mathbf{I} - \gamma \mathbf{P}^\pi)^{-1} \mathbf{r}^\pi = (\mathbf{I} - \gamma \mathbf{P}\mathbf{\Pi})^{-1} \mathbf{\Pi}\mathbf{r}$$

Model-based reinforcement learning

- Usually, the model of the environment ($\mathbf{r}$, $\mathbf{P}$) is unknown
- Instead, the learner observes transitions and rewards in the environment
- Model-based learning algorithms use this data to build an *approximate model* $\hat{\mathbf{r}}$, $\hat{\mathbf{P}}$
- Note that this is just a supervised machine learning problem!
- In the simplest case, $\hat{\mathbf{r}}$ can be estimated as the mean reward for every state-action pair, and $\hat{\mathbf{P}}$ can be estimated using counts.
- Then we pretend the approximate model is correct and use it to compute the value function through the same system as above
- Very useful approach if the models have intrinsic value, can be applied to new tasks (e.g. in robotics)

Problems with the model-based approach

- If the state set S and action set A are very large or infinite, it will be very hard to estimate the model from data, especially for the transition probabilities, as we need many data points for every matrix entry
- If $\hat{\mathbf{P}}$ has errors, these are amplified by the inversion operation
- Even if we can estimate the model, solving the system will be very expensive
- So we need to *approximate*

Approximate state representation

- Suppose that we represented every state s with a *feature vector* ϕ_s, of size $k \leq |S|$
- We can represent all the feature vectors, for all the state, in a *feature matrix* $\mathbf{\Phi}$, of size $k \times |S|$, where the sth column is ϕ_s
- Important special cases:
  - If each column has exactly one element equal to 1 and all the others are 0, the matrix represents a *state partition*, where the state space has been partitioned in k disjoint subsets
  - If each column has exactly $l \leq k$ elements equal to 1 and all the others are 0, we have a CMAC with l overlapping tilings
- In general, the features (also called *basis functions*) can be anything (Gaussian, sine-cosine, etc)

Properties of the feature matrix

- We want $\mathbf{\Phi}$ to have rank k (features that are linear combinations of each other are useless)
- Because of the meaning of $\mathbf{\Phi}$, as mapping states to features, we have: $\mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{I}$
- This is because $\mathbf{\Phi}^T\mathbf{\Phi}$ is an $|S| \times |S|$ matrix, and we want it to map states to themselves

Bellman equations with features

- Consider the Bellman equations:

$$\mathbf{V}^\pi = \mathbf{r}^\pi + \gamma \mathbf{P}^\pi \mathbf{V}^\pi$$

- We multiply at the left by $\mathbf{\Phi}$:

$$\mathbf{\Phi}\mathbf{V}^\pi = \mathbf{\Phi}\mathbf{r}^\pi + \gamma \mathbf{\Phi}\mathbf{P}^\pi \mathbf{V}^\pi$$

- We make $\mathbf{\Phi}\mathbf{V}^\pi$ appear on the right hand side as well:

$$\mathbf{P}^\pi\mathbf{V}^\pi = \mathbf{P}^\pi\mathbf{I}\mathbf{V}^\pi = \mathbf{P}^\pi\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{V}^\pi$$

- Now we can re-write the Bellman equations:

$$\mathbf{\Phi}\mathbf{V}^\pi = \mathbf{\Phi}\mathbf{r}^\pi + \gamma\mathbf{\Phi}\mathbf{P}^\pi\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{V}^\pi \Rightarrow (\mathbf{I} - \gamma\mathbf{\Phi}\mathbf{P}^\pi\mathbf{\Phi}^T)\mathbf{\Phi}\mathbf{V}^\pi = \mathbf{\Phi}\mathbf{r}^\pi$$

Approximate models

- We re-write the above equation as:

$$\mathbf{\Phi}\mathbf{V}^\pi = (\mathbf{I} - \gamma\mathbf{\Phi}\mathbf{P}^\pi\mathbf{\Phi}^T)^{-1}\mathbf{\Phi}\mathbf{r}^\pi$$

- Let $\hat{\mathbf{r}}^\pi = \mathbf{\Phi}\mathbf{r}^\pi$; this is a vector of size k, representing the reward for every feature
- E.g., in the special case of state partitioning, the reward associated with a partition will be the sum of the rewards for the states in that partition (why?)
- Let $\hat{\mathbf{P}}^\pi = \mathbf{\Phi}\mathbf{P}^\pi\mathbf{\Phi}^T$; this is a $k \times k$ matrix showing transitions between features
- Since this is typically much smaller than the original matrix, it can be estimated more accurately with less data

Approximate value function

$$\mathbf{\Phi V}^{\pi} = (\mathbf{I} - \gamma \mathbf{\Phi P}^{\pi} \mathbf{\Phi}^{T})^{-1} \mathbf{\Phi r}^{\pi}$$

- Let $\hat{\mathbf{V}}^{\pi} = \mathbf{\Phi V}^{\pi}$; this is the approximation of the value function using the features
- E.g., in the case of a state partition, each partition will have a value associated with it, and all states in the partition share the same value
- Obviously, not all value functions can be represented correctly anymore.
- The Bellman equations for approximate values become:

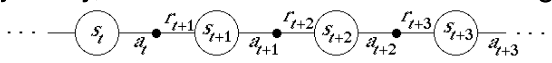
$$\hat{\mathbf{V}}^{\pi} = (\mathbf{I} - \gamma \hat{\mathbf{P}}^{\pi}) \hat{\mathbf{r}}^{\pi}$$

Trade-off

- The above systems as k equations with k unknowns
- Model-based approximate methods will estimate $\hat{\mathbf{r}}^{\pi}$ and $\hat{\mathbf{P}}^{\pi}$ from data
- The smaller k is, the less data we need to do this estimation, and the easier it is to solve the system
- But the smaller k is, the less accurate will the value function be

Using experience instead of dynamics

Consider a trajectory, with actions selected according to policy π:



The Bellman equation is:

$$V^\pi(s_t) = E_\pi\left[r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t\right]$$

which suggests the dynamic programming update:

$$V(s_t) \leftarrow E_\pi\left[r_{t+1} + \gamma V(s_{t+1}) | s_t\right]$$

In general, we do not know this expected value, but we do have an *unbiased sample* of it, $r_{t+1} + \gamma V(s_{t+1})$
In RL, we make an update *towards* the sample value, e.g. half-way

$$V(s_t) \leftarrow \frac{1}{2}V(s_t) + \frac{1}{2}\left(r_{t+1} + \gamma V(s_{t+1})\right)$$

Temporal-Difference (TD) Learning (Sutton, 1988)

We want to update the prediction for the value function based on its *change*, i.e. temporal difference from one moment to the next

- Tabular TD(0):

$$V(s_t) \leftarrow V(s_t) + \alpha\left(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right) \forall t = 0, 1, 2, \dots$$

- Gradient-descent TD(0):
  If V is represented using a parametric function approximator, e..g a neural network, with parameter θ:

$$\theta \leftarrow \theta + \alpha\left(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right) \nabla_\theta V(s_t), \forall t = 0, 1, 2, \dots$$