

# Monte Carlo Policy Evaluation

# Why Estimate the state-value?

- The state-value indicates how good it is to be in a particular state.
- An optimal policy is one that will go between the states with the highest state-value.
- Hence, if the state-values are found, the optimal policy can be found.

# Learning the state-value

- State value is the expected return when starting from that state.
- Good way of estimating this from experience is by averaging the returns after visits to that state.
- As more returns are observed, the average converges to the expected value.

# Finding state-value $V^\pi(s)$

- Episodes are generated by following policy  $\pi$
- *Every-Visit* method estimates  $V^\pi(s)$  as an average of the returns following all the visits to  $s$  in an episode.
- *First-Visit* method averages the returns of just the first visits to  $s$  in each episode.

# Monte Carlo Estimation of Action Values

# Why action values?

- If there is no model, then there is no way of knowing how to get into one state from another.
- It is then needed to know what *action* to take to get to the next desired state. (The state with the highest state value)

# The policy evaluation problem

- Need to estimate  $Q^\pi(s, a)$
- This is the expected return when starting in state  $s$  taking action  $a$ , then thereafter following policy  $\pi$

# Finding Action Values

- Action values can be found in the same way as State Values.
- *Every-Visit* estimates the value of the state-action pair as the average of the returns following visits to a state in which the action was selected
- *First-Visit* estimates the value from the first time a state is visited and an action selected.

# Why doesn't this work?

- If  $\pi$  is a deterministic policy, then only one of the actions from each state will ever be observed.
- Hence, the Monte-Carlo estimates of the other (unused) actions will not improve with experience.
- To compare alternative actions *all* actions from a state need to be estimated.

# Maintaining exploration

- This is the same problem as discussed in chapter 2.
- Problem solved by using *exploring starts*.
  - Each time an episode is generated, the starting state and action is chosen randomly.
- An alternative is to use a stochastic policy.