Announcements

Presentations in class next Tuesday and Thursday. Maximum of 10 minutes for your presentation. **Send me your slides no later than 9am the day of your presentation.**

Dec 5 Presentations

- Isaac, Adam, Shuzhe: Ventilation in Intensive Care Units
- Huiwen, Siyu: Treasure hunting
- Suyanpeng, Liyang: Pac Man
- Seok-Joo, Aditi, Ryan: Blackjack
- Andrew, Valerie, Anna: Fantasy Football Draft
- Derek, Chandra, Sajan: Robot Navigation

Last homework due next Tuesday

Model-free Approaches

Last time we covered:

- Sampling of Markov chains
- Monte Carlo policy evaluation

Stanislaw Ulam

Today we will cover methods for **policy improvement** for finite horizon problems

Monte Carlo Policy Iteration

The basic idea of MC Policy Iteration parallels standard policy iteration for MDPs:

1) Select an initial policy to evaluate
2) Evaluate the current policy using MC Policy Evaluation
3) Improve the current policy
4) Stop if convergence criteria is satisfied; Otherwise go to Step 2.

MC Policy Iteration Assumptions

In order to achieve convergence in the Monte Carlo context there are two important assumptions that must be met:

Assumption 1 (Exploration): *When following a fixed policy there are some state action pairs, (s, a), that may not be visited. Therefore a randomized policy is necessary to guarantee all state action pairs are visited*.

Assumption 2 (Convergence): *An infinite number of sample paths is necessary to guarantee convergence to the optimal policy.*

Assumption 1 can be satisfied using a ϵ-randomized policy to guarantee all state action pairs are visited. **Assumption 2** is impractical and we must use a finite number of iterations and settle for an approximation of the optimal policy.

ϵ – randomized policy

For any deterministic policy, π, we wish to evaluate we can generate an ϵ- randomized policy, $\hat{\pi}$, as follows:

$$\hat{\pi}(s) = \begin{cases} \pi(s) & with\ probability\ 1-\epsilon \\ \hat{a} & with\ probability\ \epsilon \end{cases}$$

Where action $\hat{a}$ is a randomly selected action from a **uniform distribution** over actions a $\in A$:

$$\hat{a} = a \in A\ \ with\ probability\ \frac{1}{|A|}$$

Monte Carlo Policy Iteration

The following algorithm will converge to an ϵ –optimal policy for a finite horizon problem with N stages.

Algorithm (MC Policy Iteration):

1. For all states $s \in S$ initialize $\pi(s)$; initialize ϵ to create randomized policy $\hat{\pi}(s)$.

2. ***Policy Evaluation***:

   Randomly select an **infinite** number of starting pairs, (s, a), and a corresponding sample path

   For all (s, a) in the sample path compute:

$$\tilde{Q}^{\hat{\pi}}(s,a) = r_0(s,a) + \sum_{t=1}^{N-1} \lambda^t r_t(s_t, \hat{\pi}(s_t)) + \lambda^N r_N(s_N)$$

3. ***Policy Improvement***:

   For all s: $\pi(s) \in \text{argmax}_{a \in A}\{Q(s,a)\}$

   Return to Step 2;

Assumption 2: Infinite Sample Paths

For practical implementation of MC Policy Iteration a **finite number of samples** must be specified for Step 2 (policy evaluation) and a **stopping criteria** must be specified.

Finite Samples: Use experiments with test policies to estimate the number of samples necessary to achieve reasonable confidence in estimates of $\tilde{Q}^{\hat{\pi}}(s,a)$.

Stopping Criteria: Test the norm of difference in $\tilde{Q}^{\hat{\pi}}(s,a)$ from one iteration to the next. Stop if:

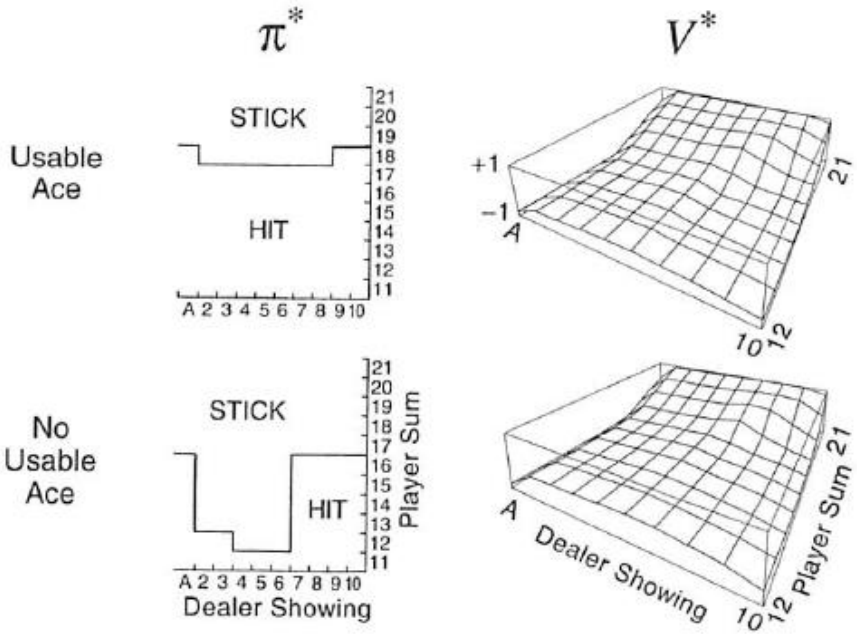
$$||\tilde{Q}^{\hat{\pi},n+1} - \tilde{Q}^{\hat{\pi},n+1}|| < \boldsymbol{\delta}$$

Where δ is a suitably chosen tolerance.

Example: Blackjack

Black jack is a sequential card game with a "player" and a "dealer". The player receives two cards face up; the dealer one card face down and one face up. The player must decide one at a time whether to take a card (hit) or not (stick). The goal is to get as close to 21 without going over.

Results from MC Policy Iteration:



Taken from "Reinforcement Learning: An Introduction", By Sutton and Barto, MIT Press

Other Reinforcement Learning Methods

There are **many other RL methods** that have been shown to be more efficient than MC Policy Iteration.

A common theme of these methods is to learn more quickly by updating the policy frequently (e.g. every time a path is sampled).

Q-Learning is one such method that estimates the **Q-value** of a given state action pair (s, a), defined as:

$$Q_m(s,a) = (1-\mu_m)Q_{m-1}(s,a) + \mu_m(r(s,a) + \lambda \max_{a' \in A} Q_{m-1}(s,a'))$$

Parameter $\mu_m \in (0,1)$ generates a **weighted average** that weights the most recent estimate of the Q-value against the newly updated estimate.

Incremental updating

Q-Learning uses the **Robins-Monro (RM) Algorithm** which is a **recency-weighted updating** approach to estimate the expectation of random variables incrementally.

Algorithm (Robins-Monro):

1. Let $\mathrm{X_i}$ denote the i^{th} sample of random variable X, and let $\mathrm{Y^m}$ denote the estimate of $\mathrm{E[X]}$ in the $\mathrm{m^{th}}$ iteration; Set $\mathrm{Y^0} = 0$.

2. Update Y as follows: $\mathrm{Y^m} = (1 - \mu_m)Y^{m-1} + \mu_m X_m$

3. If $|\mathrm{Y^m} - \mathrm{Y^{m-1}}| < \epsilon$ then stop; Otherwise $\mathrm{m} = \mathrm{m} + 1$ and return to Step 2

The parameter $\mu_m \in [0,1]$ is the **step size** and it defines the relative weight of the current estimate and most recent sample.

Q-Learning for Infinite Horizon Problems

MICHIGAN ENGINEERING
UNIVERSITY of MICHIGAN ■ COLLEGE of ENGINEERING

Algorithm (Q-Learning):

1. Set $\mathrm{Q(s,a)} = 0$ for all $\mathrm{s} \in S$ and $\mathrm{a} \in A$. Set $\mathrm{k} = 0$ and max iterations K.

2. Let the current state be i. With probability ϵ select action, a, randomly; with probability $1 - \epsilon$ choose $a = \mathrm{argmax}\{\mathrm{Q}_m(\mathrm{i,a})\}$

3. Sample a new state, j, given action a, and update $\mathrm{Q_m(i,a)}$ as

$$Q_m(i,a) = (1-\mu_m)Q_{m-1}(i,a) + \mu_m\left(r(j,a) + \lambda \max_{a' \in A} Q_{m-1}(j,a')\right)$$

4. If $\mathrm{k} < \mathrm{K}$ set $\mathrm{i} = j$ and return to Step 2. Otherwise go to Step 5.

5. For each state, s, choose the action, a, that maximizes $\mathrm{Q^K(s,a)}$

Learn More

There are many approaches being developed for reinforcement learning and more broadly in the are of artificial intelligence. Following are good resources to start learning more:

Abhijit Gosavi, 2009, Reinforcement Learning: A Tutorial Survey and Recent Advances, INFORMS Journal on Computing, 212, 178-192. (**on Canvas**)

“Reinforcement Learning: An Introduction”, By Sutton and Barto, MIT Press