- **Model Free Control**
  - Last lecture, we saw Model-Free Prediction. This time we want to optimize VF
  - Most interesting problems have MDPs that are either:
    1. Unknown to us
    2. Too big to use except with sampling
       - ↳ Model-free controls addresses these limitations

- **On and Off-Policy Learning**
  - On-policy Learning: "Learning on the job"
    - ↳ Learning about policy π from experience sampled from π
  - Off-policy Learning: "Looking over Someone's Shoulder"
    - ↳ Learn about policy π from experience sampled from μ

- **Generalized Policy Iteration (Refresher)**
  - Alternate between policy evaluation and greedy policy improvement
    - ↳ Converges on optimal VF
  - Can we swap in MC Evaluation to evaluate VF instead of using DP?
    - ↳ There are two problems:

We'll come back to this ⟶

2. If you act greedily, there is a probability that you do not explore the entire State Space and therefore miss optimal policy.

1. We want to be model free, but you need a model of the MDP to know state transition dynamics.
   - ↳ Alternative is to use action value function

$$\pi^*(s) = \underset{a \in A}{\operatorname{argmax}}\ Q(s, a)$$

↳ Policy Iteration with Action-Value Function

Can you discuss how this is done? ⟶ Policy Evaluation: Monte Carlo policy Evaluation $Q = q_\pi$

↳ Take mean of all Sth action pairs

Policy Improvement: Greedy policy improvement?

↳ No! We may miss the optimal choice

↳ Back to problem ②

↳ You need to keep exploring all options to properly estimate value function at each State

↳ ε-Greedy Exploration:

↳ Take random action with probability ε

↳ Take greedy action with probability 1-ε

↳ ε-Greedy Guarantees a policy improvement:

$$q_\pi(s, \pi'(s)) = \sum_{a \in A} \pi'(a|s)\, q_\pi(s, a)$$

$$= \underbrace{\frac{\varepsilon}{m} \sum_{a \in A} q_\pi(s, a)}_{\text{Random action}} + \underbrace{(1-\varepsilon) \max_{a \in A} q_\pi(s, a)}_{\text{Greedy Action}}$$

Max is greater than any weighted sum over all actions. ⟶

$$\geq \frac{\varepsilon}{m} \sum_{a \in A} q_\pi(s, a) + (1-\varepsilon) \sum_{a \in A} \frac{\pi(a|s) - \frac{\varepsilon}{m}}{1-\varepsilon}\, q_\pi(s, a)$$

$$= \sum_{a \in A} \pi(a|s)\, q_\pi(s, a) = v_\pi(s)$$

- Monte Carlo Policy Iteration

↳ Policy Evaluation: Monte Carlo policy Evaluation, $Q = q_\pi$

↳ Mean return from every State-action pair

↪ Policy Improvement: ϵ-greedy policy improvement (Stochastically)

* This will converge to optimal policy, given enough time

↪ Now let's try to make this more efficient.

- Monte Carlo Control: Improve policy after each episode. Always act greedily with respect to most recent updated value function.

↪ Iterate between running single episodes and immediately improving policy

- How can we guarantee that we find the best possible policy?

↪ We must balance exploration of new policies and exploitation of existing policies.

↪ One idea: Greedy in the limit of infinite exploration (GLIE)

↪ Come up with a schedule such that two conditions are met:

① All state-action pairs are explored infinitely many times

$$\lim_{k \to \infty} N_k(s,a) = \infty$$

↪ E.g., ϵ-greedy policy improvement satisfies this

② The policy converges on a greedy policy:

$$\lim_{k \to \infty} \pi_k(a|s) = \mathbb{I}\left(a = \underset{a' \in A}{\operatorname{argmax}}\, Q_k(s,a')\right)$$

↪ E.g. Let ϵ decay slowly such as $\epsilon_k = 1/k$

- GLIE MC Control:

- Sample k^{th} episode using π: $\{S_1, A_1, R_2, \ldots S_T\} \sim \pi$

- For each state S_t and action A_t in the episode:

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$

Incremental Update to the mean $\longrightarrow$ $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)} \left(G_t - Q(S_t, A_t)\right)$$

Taking returns from better and better policies into account. Past gets dominated by more informed, recent q-values.

- Improve Policy based on new action-value function

$$\epsilon \leftarrow 1/k$$

$$\pi \leftarrow \epsilon\text{-Greedy}(Q)$$

* This algorithm finds optimal action-value function
  - ↳ You can update policy after each algorithm
  - ↳ Does it matter how you initialize q?
    - ↳ For this particular algorithm, it makes no difference. In other algorithms, a good initialized q will converge sooner, but convergence guaranteed eventually regardless of q init.

- **MC vs. TD Control:**

TD has advantages over MC:

1. Lower Variance
2. Online
3. Incomplete Sequences

Natural Idea: use TD instead of MC in our policy iteration strategy

- ↳ Use TD instead of MC to estimate $Q(s, a)$
- ↳ Use ϵ-greedy improvement
- ↳ Run policy Improvement after every time step

- **SARSA:** $S, A \xrightarrow{R} S' \longrightarrow A'$

$$Q(S, A) \leftarrow Q(S, A) + \alpha \left(R + \gamma Q(S', A') - Q(S, A)\right)$$

↳ Each step, we update q-value using SARSA

↳ You've already changed your value function, so at every timestep, update policy using ϵ-greedy policy improvement

Algorithm

Initialize $Q(s,a)$

Repeat: For every episode

Initialize S (pick random starting state)
Choose action from S using ϵ-greedy policy

Repeat: For each step of the episode

Take action, observe R, S'
Choose A' from S' using ϵ-greedy policy
Update $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S', A') - Q(S,A)]$

Until S is terminal