6.792 Reinforcement Learning Midterm

Gatlen Culp · Sunday, 2025 Nov 02

PART 0: Mathematical Foundations

Mathematical Foundations

Background material used throughout course

Linear Algebra

Least Squares: $\boldsymbol{x}^\star := (A^\mathsf{T}A)^{-1}A^\mathsf{T}\boldsymbol{b} = A^\dagger\boldsymbol{b}$ where $A^\dagger$ is pseudoinverse.
Matrix Inversion: $(I+UV)^{-1} = I - U(I+VU)^{-1}V$

Gradients: $\nabla_{\boldsymbol{x}}(\boldsymbol{x}^\mathsf{T}A\boldsymbol{x}) = (A+A^\mathsf{T})\boldsymbol{x}; \quad \nabla_{\boldsymbol{x}}(\boldsymbol{u}^\mathsf{T}A\boldsymbol{x}) = A^\mathsf{T}\boldsymbol{u}$

Probability

Variance ($\boldsymbol{X},\boldsymbol{Y}$ indep): $\mathrm{Var}[aX+bY] = a^2\,\mathrm{Var}[X] + b^2\,\mathrm{Var}[Y]$.
Telescoping: $V(s_0) = \sum_{t=0}^{T}\gamma^t(V(s_t) - \gamma V(s_{t+1}))$. **Jensen:** For convex f: $f(\mathbb{E}[X]) \le \mathbb{E}[f(X)]$

Fixed Point Theory

Banach Theorem: If (A) (X,d) is a complete metric space, (B) $\mathcal{T}$ is a contraction $d(\mathcal{T}x,\mathcal{T}y) \le \gamma d(x,y)$ for $\gamma \in (0,1)$, then unique $x^\star$ exists with $\mathcal{T}x^\star = x^\star$ and $d(x_t,x^\star) \le \frac{\gamma^t}{1-\gamma}d(x_1,x_0)$

Sequential Decision Making Foundations (L01)

L01: Introduction and MDP formulation

Markov Process Tests

Projections: Fail unless eliminated dim doesn't affect future. **Sums:** Fail when different states $\to$ same sum. **Subsampling:** Preserves. **Augmentation:** Preserves if all deps captured. **State Augmentation:** Include all lagged states/actions: $\overline{S}_{t+1} = B\overline{S}_t + Ca_t + Dw_t$

Expectation vs Optimization

$\min_{a\in\mathcal{A}}\mathbb{E}[g(W,a)] \ge \mathbb{E}[\min_{a\in\mathcal{A}} g(W,a)]$. Equality when $a = \mu(W)$: $\min_{\mu\in\Pi}\mathbb{E}[g(W,\mu(W))] = \mathbb{E}[\min_a g(W,a)]$

PART 1: Dynamic Programming

Finite Horizon Dynamic Programming (L02)

L02: Finite horizon DP algorithm

Finite Horizon DP

$J_t^\star(s_t) = \min_{a_t\in\mathcal{A}(s_t)}\mathbb{E}[c_t(s_t,a_t,w_t) + J_{t+1}^\star(s_{t+1})] \quad J_T^\star(s_T) = c_T(s_T)$

Special Structures (L03-05)

L03: DP arguments, inventory problem, optimal stopping

Base-Stock Policy

$a_t^\star := \min\left((\overline{s}_t - s_t)^+, B_t\right)$ (order up to $\overline{s}_t$ if below). Arises from concave value function.

L04-05: Linear quadratic regulator

LQR (Linear Quadratic Regulator)

Dynamics: $\boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t$; **Cost:** $J = \frac{1}{2}\boldsymbol{x}_T^\mathsf{T}Q_T\boldsymbol{x}_T + \frac{1}{2}\sum_{t=0}^{T-1}(\boldsymbol{x}_t^\mathsf{T}Q\boldsymbol{x}_t + \boldsymbol{u}_t^\mathsf{T}R\boldsymbol{u}_t)$

Finite Horizon

Policy: $\boldsymbol{u}_t^\star = -K_t\boldsymbol{x}_t$ where gain $K_t = (R + B^\mathsf{T}P_{t+1}B)^{-1}B^\mathsf{T}P_{t+1}A$.
Value: $V_t^\star(x) = \frac{1}{2}\boldsymbol{x}^\mathsf{T}P_t\boldsymbol{x}$

Riccati: $P_t := Q + A^\mathsf{T}P_{t+1}A - A^\mathsf{T}P_{t+1}B(R+B^\mathsf{T}P_{t+1}B)^{-1}B^\mathsf{T}P_{t+1}A$ with $P_T = Q_T$

Infinite Horizon

Policy: $\boldsymbol{u}^\star = -K\boldsymbol{x}$ where gain $K = (R+B^\mathsf{T}PB)^{-1}B^\mathsf{T}PA$. **ARE:** $P = Q + A^\mathsf{T}PA - A^\mathsf{T}PB(R+B^\mathsf{T}PB)^{-1}B^\mathsf{T}PA$

Stability: If (A,B) controllable and $\left(A, Q^{\frac{1}{2}}\right)$ observable, then closed-loop stable. **Lyapunov:** $P = (A-BK)^\mathsf{T}P(A-BK) + Q + K^\mathsf{T}RK$ shows $\boldsymbol{x}_t^\mathsf{T}P\boldsymbol{x}_t$ strictly decreases

Stochastic LQR

Dynamics: $\boldsymbol{x}_{t+1} = A_t\boldsymbol{x}_t + B_t\boldsymbol{u}_t + \boldsymbol{w}_t$ with $A_t, B_t, \boldsymbol{w}_t$ mutually independent. **Policy:** $\boldsymbol{u}_t^\star = L_t\boldsymbol{x}_t$ where $L_t = -(R_t + \mathbb{E}[B_t^\mathsf{T}K_{t+1}B_t])^{-1}\mathbb{E}[B_t^\mathsf{T}K_{t+1}A_t]$

Riccati: $K_t = \mathbb{E}[A_t^\mathsf{T}K_{t+1}A_t] - \mathbb{E}[A_t^\mathsf{T}K_{t+1}B_t](R_t + \mathbb{E}[B_t^\mathsf{T}K_{t+1}B_t])^{-1}\mathbb{E}[B_t^\mathsf{T}K_{t+1}A_t] + Q_t$ with $K_T = Q_T$ (expectations around random matrices)

Controllability

(A,B) controllable if $\mathcal{C} = \begin{pmatrix} B & AB & \dots & A^{n-1}B \end{pmatrix}$ is rank-n. Can then place eigenvalues of $A - BK$ arbitrarily.

Infinite Horizon Dynamic Programming (L06-07)

L06-07: Bellman equations, value iteration, policy iteration

Bellman Operator

$\mathcal{T}V(s) := \min_a\left(c(s,a) + \gamma\sum_{s'}p(s' \mid s,a)V(s')\right)$
$\mathcal{T}_\mu V(s) := c(s,\mu(s)) + \gamma\sum_{s'}p(s' \mid s,\mu(s))V(s')$

Properties: Monotonicity ($V \le V' \Rightarrow \mathcal{T}V \le \mathcal{T}V'$), Contraction ($\|\mathcal{T}V - \mathcal{T}V'\| \le \gamma\|V-V'\|_\infty$), Fixed Point ($\mathcal{T}V^\star = V^\star$ unique), Convergence ($\lim_{k\to\infty}\mathcal{T}^kV_0 = V^\star$)

Matrix Form: $V^\pi = R^\pi + \gamma P^\pi V^\pi \Rightarrow V^\pi = (I-\gamma P^\pi)^{-1}R^\pi$

Value Iteration

$V_{k+1} = \mathcal{T}V_k$ converges: $\|V_k - V^\star\| \le \gamma^k\|V_0 - V^\star\|$. **Greedy:** $\mu_k(s) \in \arg\min_a\left(c(s,a) + \gamma\sum_{s'}p(s' \mid s,a)V_k(s')\right)$

Policy Iteration

1. **Eval:** Solve $V^\mu = \mathcal{T}_\mu V^\mu$; 2. **Improve:** $\mu' \in \arg\min_a\left(c(s,a) + \gamma\sum_{s'}p(s' \mid s,a)V^\mu(s')\right)$; 3. Repeat. Each iteration improves: $V^{\mu'} \le V^\mu$

VI vs PI

PI: Fewer iterations, sensible policies, expensive per iteration (solve $|\mathcal{S}|$ linear eqs), better for small state spaces. **VI:** Simple updates, more iterations, intermediate policies not guaranteed to improve, better for large state spaces

Performance Loss Bound

If $\|V - V^\star\| \le \varepsilon$ and $\mathcal{T}V = \mathcal{T}_\mu V$, then $\|V^\mu - V^\star\| \le \frac{2\gamma\varepsilon}{1-\gamma}$ (tight bound)

PART 2: Reinforcement Learning

Model-Free Policy Evaluation (L08, L10)

L08: Monte Carlo, TD, TD(λ) – Estimating $\hat{V}^\pi$ (possibly) without model

All V here are an alias for $\hat{V}^\pi$

Monte Carlo (MC/TD(1))

Update:

$$V_{n+1}(s_0) \leftarrow (1-\eta_{n+1})V_n(s_0) + \eta_{n+1}\hat{R}_{n+1}(s_0)$$

where $\hat{R}_i(s_0) := \sum_{t=0}^{T_i}\gamma^t r_{t,i}$ (full return)

Properties: High variance, no bias, needs full episode

Temporal Difference Methods

TD(0)

Update:

$$V_{t+1}(s) \leftarrow V_t(s) + \eta_t\delta_t \quad \text{where} \quad \delta_t := r_t + \gamma V_t(s') - V_t(s)$$

Properties: Bootstrapped one-step, low variance, biased, online

TD(λ)

Backward View (Eligibility Traces): $\forall s : V(s) \leftarrow V(s) + \eta_t\delta_t e_t(s)$ where $e_t(s) := \gamma\lambda e_{t-1}(s) + \mathbb{1}(s_t = s)$ tracks recently-visited states (more efficient)

Forward View: Mixes n-step returns: $G_t^{(\lambda)} := (1-\lambda)\sum_{n=1}^{T-t}\lambda^{n-1}G_t^{(n)}$ where $G_t^{(n)} := \sum_{k=0}^{n-1}\gamma^k r_{t+k} + \gamma^n V(s_{t+n})$

L10: Convergence of TD Methods – Noisy bootstrapped updates work

Not obvious that bootstrapping doesn't blow up your system (TD(1) converges more simply).

Martingales and Supermartingales

Martingale: Sequence $\{M_t\}$ w.r.t. $\mathcal{F}_t$ where $\mathbb{E}[|M_t|] < \infty$ and $\mathbb{E}[M_{t+1} \mid \mathcal{F}_t] = M_t$ (expected value doesn't change). **MDS:** $\{w_t\}$ where $\mathbb{E}[w_t \mid \mathcal{F}_t] = 0$ (zero-mean noise conditional on past)

Supermartingale: $\mathbb{E}[M_{t+1} \mid \mathcal{F}_t] \le M_t$ (expected value decreases or stays same). Used in convergence proofs to bound error terms.

Stochastic Approximation (SA)

Goal: Find $\theta^\star = h(\theta^\star)$ using noisy $H(\theta,\xi_t)$ where $\mathbb{E}[H(\theta,\xi_t)] = h(\theta)$.
Update: $\theta_{t+1} \leftarrow \theta_t + \eta_t(H(\theta_t,\xi_t) - \theta_t)$

Decomposition: $H(\theta_t,\xi_t) - \theta_t = (h(\theta_t) - \theta_t) + w_t$ where w_t is MDS. At fixed point: $h(\theta^\star) = \theta^\star$

TD(0) as SA: $V_{t+1} \leftarrow V_t + \eta_t E_t\circ(\mathcal{T}_\pi V_t - V_t + w_t)$ where E_t is one-hot at s_t, $w_t := \mathcal{T}_\pi^E V_t - \mathcal{T}_\pi V_t$

Robbins-Monro Stepsize

$$\underbrace{\sum_{t=0}^{\infty}\eta_t = \infty}_{\text{Updates non-negligible}} \quad \text{and} \quad \underbrace{\sum_{t=0}^{\infty}\eta_t^2 < \infty}_{\text{Noise diminishes}} \quad \left(e.g.\eta_t := \frac{1}{t+1}\right)$$

Convergence Insights

Max Norm Convergence: RM stepsizes + unbiased noise + bounded variance ($\mathbb{E}[w_t^2(s) \mid \mathcal{F}_t] \le A + B\|x_t\|^2$) + h is max-norm contraction $\Rightarrow \theta_t \to \theta^\star$

Segment Analysis: Split time into segments where $\prod_{t=t_i}^{t_{i+1}-1}(1-\eta_t) \le \frac{1}{2}$, then $\left|\mathbb{E}\left[x_{t_{i+1}}\right]\right| \le \frac{|\mathbb{E}[x_{t_i}]|}{2}$ (exponential convergence of bias).
Variance: $v_{i+1} \le \frac{1}{4}v_i + \varepsilon_i$ where $\varepsilon_i \to 0$. Both bias and variance $\to 0$

Expected Progress: If $\mathcal{T}_\pi$ is γ-contraction: $\mathbb{E}\left[\|V_{t+1} - V^\pi\|_\infty \mid \mathcal{F}_t\right] \le (1-\eta_t(1-\gamma)p_t)\|V_t - V^\pi\|_\infty$ where $p_t :=$ prob of updating worst state

Model-Free Control (L09)

L09: Q-learning, SARSA

SARSA vs Q-Learning

SARSA (On-Policy):

$$Q_{t+1}(s_t,a_t) \leftarrow Q_t(s_t,a_t) + \eta_t[r_t + \gamma Q_t(s_{t+1},a_{t+1}) - Q_t(s_t,a_t)]$$

where $a_{t+1} \sim \pi(\cdot \mid s_{t+1})$. Learns about current policy.

Q-Learning (Off-Policy):

$$Q_{t+1}(s_t,a_t) \leftarrow Q_t(s_t,a_t) + \eta_t\left[r_t + \gamma\max_{a'}Q_t(s_{t+1},a') - Q_t(s_t,a_t)\right]$$

Learns optimal $Q^\star$ regardless of behavior policy. Converges to $Q^\star$ under: coverage (all (s,a) visited infinitely often), RM stepsizes, bounded rewards.

Approximate Value-Based RL (L12-13)

L12: Function approximation, approximate policy evaluation – Parameterized value functions when tabular representation is infeasible or suboptimal.

Value Approximation/Parameterization

Tabular Value: Independent value for each state (or state-action pair)

Approximate/Parameterized Value (V_θ, Q_θ w/ params θ):
- **Must use:** Continuous/large state spaces (robotics, Atari, Go)
- **Should use:** Related states share features; sample efficiency; generalization to unvisited states
- Parameterized doesn't necessarily mean approximate (can restrict hypothesis space for efficiency)
- Convergence guarantees may be lost depending on parameterization

Common Parameterizations ($V_\theta \in \mathcal{F}$):
Linear:

$$V_\theta(s) = \theta^\mathsf{T}\phi(s) \quad \text{where} \quad \phi : \mathcal{S} \to \mathbb{R}^d$$

Features ϕ are hand-crafted, can be nonlinear (Fourier, RBF, polynomials). Convex optimization, convergence guarantees.

Neural Nets: $V_\theta(s) = \mathrm{NN}_\theta(s)$ - nonlinear in features and weights, learned jointly. Powerful but no convergence guarantees.

Approximate Policy Evaluation (Linear)

With non-tabular values, we make analogs to Model-Free Policy Evaluation methods:

Approximate MC/TD(1): Supervised learning on full returns

$$\hat{\theta}_n \leftarrow \arg\min_\theta\frac{1}{n}\sum_{i=1}^{n}(V_\theta(s_i) - R_i)^2 \quad \text{(Offline)}$$

$$\hat{\theta}_{i+1} \leftarrow \hat{\theta}_i - \alpha_i\nabla_\theta(V_\theta(s_i) - R_i)^2 \quad \text{(Online)}$$

where $R_i = \sum_{t=0}^{T_i}\gamma^t r_{t,i}$ (discounted return). Very sample inefficient, no one uses.

Approximate TD(0): Semi-gradient descent (freeze bootstrapped target)

$$\hat{\theta}_{t+1} \leftarrow \hat{\theta}_t + \alpha_t\delta_t\nabla_\theta V_\theta(s_t)\,|_{\theta=\hat{\theta}_t}$$

where $\tilde{R}_t := r_t + \gamma V_{\hat{\theta}_t}(s_{t+1})$ and $\delta_t := \tilde{R}_t - V_{\hat{\theta}_t}(s_t)$

Why "semi-gradient"? We treat $\tilde{R}_t$ as constant, ignoring $\nabla_\theta V_\theta(s_{t+1})$ term. Online, efficient.

LSTD (Least Squares TD): Model-based, closed-form solution
- **Key idea:** Find fixed point of $\Pi\mathcal{T}_\pi$ (projection composed with Bellman operator)
- **Why:** True V^π may lie outside function class; seek V_θ such that $\Pi\mathcal{T}_\pi V_\theta = V_\theta$
- **Requirements:** Linear parameterization, stationary distribution ρ^π (or estimate from samples), model/batch samples
- **Solution:** Solve weighted least squares

$$\theta^\star = \arg\min_\theta\sum_s\rho^\pi(s)(\mathcal{T}_\pi V_\theta(s) - V_\theta(s))^2$$

which gives closed-form: $\theta^\star = (\Phi^\mathsf{T}D\Phi)^{-1}\Phi^\mathsf{T}D\mathcal{T}_\pi V_\theta$ where $D = \mathrm{diag}(\rho^\pi)$, Φ is feature matrix
- **Convergence:** Converges to unique fixed point of $\Pi\mathcal{T}_\pi$

Linear TD: Model-free iterative version of LSTD

$$\theta_{t+1} \leftarrow \theta_t + \eta_t\phi(s_t)\left(r_t + \gamma\phi(s_{t+1})^\mathsf{T}\theta_t - \phi(s_t)^\mathsf{T}\theta_t\right)$$

Can be written as iterative estimation of $A\theta = b$:

$$A = \mathbb{E}\left[\phi(s)(\phi(s) - \gamma\phi(s'))^\mathsf{T}\right]; \quad b = \mathbb{E}[\phi(s)r]$$

Solution: $\theta^\star = A^{-1}b$ (same as LSTD). Converges to LSTD fixed point under RM stepsizes and coverage.

Trade-off: LSTD is faster (closed-form) but needs model/batch; Linear TD is slower (iterative) but model-free.

Deadly triad: Function approximation + bootstrapping + off-policy can cause divergence

Performance Loss with Approximation: If policy π satisfies:

$$\mathcal{T}_\pi V_\pi(i) \ge \mathcal{T}V_\pi(i) - \delta \quad \forall i$$

Then: $V^\star(i) - V_\pi(i) \le \frac{\delta}{1-\gamma} \quad \forall i$

Shows approximate greedy policies still perform well.

L13: Approximate VI, fitted Q iteration, DQN, DDQN

Although typically used interchangeably – **Parameterized** (value with parameters) $\supset$ **Approximate** (approximation error even at optimum) $\supset$ **Fitted** (refers to the value approximation but also the process by which it is updated – uses supervised learning instead of analytical projection)

Approximate Value Iteration (AVI)
- **Normal VI:** $V_{k+1} \leftarrow \mathcal{T}V_k$, contraction implies $V_k \to V^\star$
- **Approximate VI:** VI on any parameterized value function. After each Bellman update: $V_{k+1} \leftarrow \mathcal{A}\mathcal{T}V_k$ where generic function approximation operator $\mathcal{A}$ is often $:= \arg\min_{V\in\mathcal{F}}\|f - V\|_\infty$, maps any function to the representable class $\mathcal{F}$ (linear basis, kernels, deep nets, ...) (Π^ρ before was an example)
  - Note: This is more of an analytical definition, in practice $\mathcal{F}$ is huge, you can't do the approximation/projection $\mathcal{A}$ directly. That's why we need DQNs or Fitted Q-Iteration. Fitted Q-iteration is offline (fixed dataset). Q-learning is online (continuous interaction)

Fitted Q-Iteration (FQI): Offline supervised learning approach to AQI

$$\hat{\theta}_{i+1} \leftarrow \hat{\theta}_i - \alpha_i \nabla_\theta \tilde{\mathcal{L}}(s_{1:n}, a_{1:n}, y_{1:n}; \theta)$$
$$= \hat{\theta}_i - \alpha_i \sum_t \left(Q_\theta(s_t, a_t) - r_t - \gamma \max_{a'} Q_{\theta_i}(s_{t+1}, a') \right) \nabla_\theta Q_\theta(s_t, a_t)$$

where $\tilde{\mathcal{L}}$ is empirical loss over dataset, y_t is TD-error. Uses target network Q_{θ_i} (frozen during update) for stability.

Performance Bound: After K iterations with greedy policy π_K:

$$\|V^\star - V^{\pi_K}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \max_{0 \leq k < K} \|\mathcal{T} V_k - \mathcal{A}\mathcal{T} V_k\|_\infty + \frac{2\gamma^{K+1}}{1-\gamma} \|V^\star - V_0\|_\infty$$

Properties: If $\mathcal{A}$ is L_∞-projection, then non-expansive and $\mathcal{A}\mathcal{T}$ is contraction with unique fixed point $\tilde{V}$ where $\tilde{V} = \mathcal{A}\mathcal{T}\tilde{V}$

Deep Q-Networks

DQN: Q-learning (also kind of FQI) with neural nets $Q_\theta(s, a)$. Stabilization techniques:

- **Replay buffer:** Store transitions (s, a, r, s'); sample i.i.d. batches to break temporal correlation
- **Target network:** Frozen Q_{θ^-} updated periodically for consistent targets $y = r + \gamma \max_{a'} Q_{\theta^-}(s', a')$
- Issue: Overestimates Q due to $\max_{a'} Q(s', a')$ over noise

Double DQN: Decouple action selection (online net) from evaluation (target net):

$$y = r + \gamma Q_{\theta^-}\left(s', \arg\max_{a'} Q_\theta(s', a')\right)$$

Reduces overestimation bias by not using same network for both selection and evaluation.

Policy Gradient Methods (L14-15)

L14: Policy gradient, variance reduction Instead of a tabular policy π with independent states, lets also parameterize it π_θ for the same reasons we might want to approximate value. Now, policy iteration breaks.

- MUST use a parameterized policy when ACTION space is continuous or extremely large
- SHOULD use a parameterized policy when your STATE space is large enough that maintaining separate policy parameters for each state is unwieldy (Can still have large state space and use a linear parameterization of the policy $\pi(a \mid s) = \text{softmax}\left((\theta^a)^\top \varphi(s)\right)$ but doing so requires a full parameter vector for each action)

Policy Gradient Theorem

- μ is the initial state distribution.
- $V_\mu(\pi_\theta) = \mathbb{E}_{s_0 \sim \mu}[V^{\pi_\theta}(s_0)]$

$$\nabla_\theta V_\mu(\pi_\theta) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}} \left[R(\tau) \cdot \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]$$

where $R(\tau) \coloneqq \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$

Intuition: Increase log-probability of actions proportional to their return

Causality: Past rewards don't affect gradient at time t, so can replace $R(\tau)$ with future returns:

$$\sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}$$

Actor-Critic Form:

$$\nabla_\theta V_\mu(\pi_\theta) = \mathbb{E}_{\tau \sim P} \left[\sum_{t=0}^{\infty} \gamma^t Q^{\pi_\theta}(s_t, a_t) \cdot \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]$$

Discounted State Distribution Form:

$$\nabla_\theta V_\mu(\pi_\theta) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_\mu^{\pi_\theta}} \mathbb{E}_{a \sim \pi_\theta} [Q^{\pi_\theta}(s, a) \cdot \nabla_\theta \log \pi_\theta(a \mid s)]$$

where $d_\mu^{\pi_\theta}(s) \coloneqq (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \Pr[s_t = s \mid \pi_\theta]$

REINFORCE ($\nabla_\theta V_\theta$ using MC/TD(0) Policy Evaluation)

Algorithm: Monte Carlo policy gradient

1. Collect episode τ under π_θ
2. Compute returns: $G_t \coloneqq \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$
3. Update:

$$\theta \leftarrow \theta + \eta \sum_{t=0}^{T} G_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)$$

Properties: On-policy, high variance, unbiased gradient estimates

Variance Reduction with Baseline:

$$\theta \leftarrow \theta + \eta \sum_{t=0}^{T} (G_t - b(s_t)) \nabla_\theta \log \pi_\theta(a_t \mid s_t)$$

Common choice: $b(s_t) \coloneqq V(s_t)$ (value function baseline)

Best: Use advantage $A(s_t, a_t) \coloneqq Q(s_t, a_t) - V(s_t)$ for variance reduction

L15: Actor-critic methods – Compatible function approximation, A2C, A3C, DDPG, SAC

Actor-Critic Methods

Idea: Combine value-based and policy-based approaches

- **Actor:** Policy network π_θ that takes actions
- **Critic:** Value function V_φ or Q_φ that estimates values to reduce variance

A2C (Advantage Actor-Critic)

Update: Synchronous version of A3C

$$\theta \leftarrow \theta + \eta \nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{A}_t$$

where advantage $\hat{A}_t \coloneqq \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n V_\varphi(s_{t+n}) - V_\varphi(s_t)$ (n-step TD)

Properties: On-policy, synchronous (waits for all workers), stable training

A3C (Asynchronous Advantage Actor-Critic)

Key difference: Multiple workers train asynchronously on different environment copies, update shared parameters without waiting

Benefits: Decorrelates data (like replay buffer), faster wall-clock time, more exploration diversity

DDPG (Deep Deterministic Policy Gradient)

For: Continuous action spaces. **Policy:** Deterministic $\mu_\theta(s)$

Actor update: $\nabla_\theta \mathbb{E}\left[Q_\varphi(s, \mu_\theta(s))\right] = \mathbb{E}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q_\varphi(s, a) \mid_{a=\mu_\theta(s)}\right]$

Critic update: TD learning for Q_φ with target networks and replay buffer (like DQN)

Exploration: Add noise $a_t = \mu_\theta(s_t) + \varepsilon_t$ where $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$ (or Ornstein-Uhlenbeck)

TD3 (Twin Delayed DDPG)

Improvements over DDPG:

1. **Twin Q-networks:** Use $\min\left(Q_{\varphi_1}, Q_{\varphi_2}\right)$ to reduce overestimation
2. **Delayed policy:** Update actor less frequently than critic
3. **Target smoothing:** Add noise to target actions $\tilde{a} = \mu_{\theta'}(s') + \varepsilon$ where $\varepsilon \sim \text{clip}(\mathcal{N}(0, \sigma^2), -c, c)$

SAC (Soft Actor-Critic)

Framework: Entropy-regularized RL (see L16) with actor-critic

Objective: $\max \mathbb{E}\left[\sum_t \gamma^t (r_t - \alpha \log \pi_\theta(a_t \mid s_t))\right]$ where α is temperature

Updates: Actor maximizes $\mathbb{E}\left[Q_\varphi(s, a) - \alpha \log \pi_\theta(a \mid s)\right]$; critic learns soft Q-function

Properties: Off-policy, stochastic policy, automatic exploration, stable

Advanced Policy Methods (L16)

L16: Conservative policy iteration, NRG, TRPO, PPO

Conservative Policy Iteration

Problem: Large policy updates can cause performance collapse

Idea: Constrain policy updates to stay close to current policy using KL divergence

General form: $\max_\theta \mathbb{E}[\ldots] \quad \text{s.t.} \quad \mathbb{E}_{s \sim d^\pi}[D_{\text{KL}}(\pi_{\theta'}(\cdot \mid s) \,\|\, \pi_\theta(\cdot \mid s))] \leq \delta$

TRPO (Trust Region Policy Optimization)

Objective: Maximize surrogate advantage while constraining KL divergence

$$\max_\theta \mathbb{E}_{s,a \sim \pi_{\text{old}}} \left[\frac{\pi_\theta(a \mid s)}{\pi_{\text{old}}(a \mid s)} A^{\pi_{\text{old}}}(s, a) \right]$$

subject to: $\mathbb{E}_{s \sim d^{\pi_{\text{old}}}}[D_{\text{KL}}(\pi_{\text{old}}(\cdot \mid s) \,\|\, \pi_\theta(\cdot \mid s))] \leq \delta$

Solution: Uses conjugate gradient and line search to approximately solve constrained optimization

Properties: Monotonic improvement guarantee, but computationally expensive

PPO (Proximal Policy Optimization)

Idea: Simpler alternative to TRPO - clip objective instead of hard constraint

PPO-Clip:

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min\left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon) \hat{A}_t \right) \right]$$

where $r_t(\theta) \coloneqq \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\text{old}}(a_t \mid s_t)}$ is probability ratio

Clipping: Prevents ratio from moving outside $[1-\varepsilon, 1+\varepsilon]$ (typically $\varepsilon = 0.2$)

PPO-Penalty: Add KL penalty to objective instead of clipping

$$\mathcal{L}^{\text{KL}}(\theta) = \mathbb{E}_t \left[r_t(\theta) \hat{A}_t - \beta D_{\text{KL}}(\pi_{\text{old}}(\cdot \mid s_t) \,\|\, \pi_\theta(\cdot \mid s_t)) \right]$$

Properties: Simple to implement, sample efficient, widely used (default RL algorithm for many applications)

Entropy-Regularized RL

Framework: Add entropy bonus to encourage exploration

Boltzmann Policy: $\pi_Q^{\mathcal{B}}(a \mid s) \coloneqq \overline{\pi}(a \mid s) \cdot \frac{\exp\left(\frac{Q(s,a)}{\tau}\right)}{\mathbb{E}_{a \sim \overline{\pi}}\left[\exp\left(\frac{Q(s,a)}{\tau}\right)\right]}$. **Soft Value:** $V_{\pi_Q^{\mathcal{B}}}(s) = \tau \log \mathbb{E}_{a \sim \overline{\pi}}\left[\exp\left(\frac{Q(s,a)}{\tau}\right)\right]$

Soft Q-Learning

Learn Q function for entropy-regularized objective. Alternate: (1) **Policy improvement:** $\pi \leftarrow \pi_Q^{\mathcal{B}}$; (2) **Policy evaluation:** Update Q toward soft Bellman target

Single-Sample Q-Update: Minimize $\frac{1}{2}\|Q_\theta(s_t, a_t) - y_t\|^2$ where $y_t \coloneqq r_t + \gamma V_{\pi_Q^{\mathcal{B}}}(s_{t+1})$ (semi-gradient). Uses soft value instead of max in Q-learning

PART 3: Special Topics

Multi-Armed Bandits (L17)

L17: Multi-arm bandits – Exploration vs exploitation with horizon = 1

Setup & Regret

At each round t, select $a_t \in \mathcal{A}$, receive $r_t(a_t)$[1]

Notation:[2]

$$\mu(a) \coloneqq \mathbb{E}[r_t(a)]; \quad a^\star \coloneqq \arg\max_a \mu(a); \quad T_n(a) \coloneqq \sum_{t=1}^{n} \mathbb{1}\{a_t = a\}$$

$$A \coloneqq |\mathcal{A}|; \quad \Delta(a) \coloneqq \mu(a^\star) - \mu(a)$$

Regret:

$$R_n = \max_a \mathbb{E}\left[\sum_{t=1}^{n} r_t(a)\right] - \mathbb{E}\left[\sum_{t=1}^{n} r_t(a_t)\right] = \sum_{a \neq a^\star} \mathbb{E}[T_n(a)] \Delta(a)$$

Explore-then-Commit

Algorithm:

1. **Explore:** Pull each arm K times (total $\tau = KA$ rounds)
2. **Commit:** Choose $\hat{a}^\star = \arg\max_a \hat{\mu}_\tau(a)$ and pull for remaining $n - \tau$ rounds

where $\hat{\mu}_\tau(a) = \frac{1}{K} \sum_{t:a_t=a}^{\tau} r_t(a)$ is empirical mean

Regret:[3]

$$R_n \leq \underbrace{\tau}_{\text{exploration cost}} + \underbrace{\mathcal{O}\left(\frac{\sqrt{A \log n}}{\tau} n\right)}_{\text{exploitation error}}$$

ε-Greedy

Algorithm: At each round t:

- With probability ε_t: explore (pick random action)
- With probability $1 - \varepsilon_t$: exploit (pick $\arg\max_a \hat{\mu}_t(a)$)

Online Regret: $R_t \leq \tilde{\mathcal{O}}\left(t^{\frac{2}{3}}\right)$ with $\varepsilon_t = t^{-\frac{1}{3}} (A \log t)^{\frac{1}{3}}$

Upper Confidence Bound (UCB)

Idea: Optimism under uncertainty - construct upper confidence bound $B_t(a)$ for each action and pick action with highest bound

Algorithm:

1. Compute estimates: $\hat{\mu}_t(a) = \frac{1}{T_t(a)} \sum_{s=1}^{t} r_s \mathbb{1}\{a_s = a\}$
2. Evaluate uncertainty: $|\hat{\mu}_t(a) - \mu(a)| \leq \sqrt{\frac{\log \frac{2}{\delta_t}}{2T_t(a)}}$ w.h.p.
3. Optimism (exploration bonus):

$$B_t(a) = \hat{\mu}_t(a) + \underbrace{\rho}_{\text{tuning param}} \sqrt{\frac{\log \frac{2}{\delta_t}}{2T_t(a)}}$$

4. Select: $a_t = \arg\max_a B_t(a)$

Actions with high uncertainty get larger exploration bonus

Regret: With $\rho = 1$, $\delta_t = \frac{1}{t}$:

$$R_n = \mathcal{O}\left(\sum_{a \neq a^\star} \frac{\log n}{\Delta(a)}\right)$$

(near-optimal, almost matches lower bounds)

Tuning ρ: $\rho < 1 \longrightarrow$ polynomial regret; $\rho \geq 1 \longrightarrow$ logarithmic regret. Larger $\rho \longrightarrow$ more exploration.

Recommendation System (RS)

- Similar to linear value approximation, when the action space is extremely large, parameterize actions with features
- Contextual bandits (have a linear context variable, e.g. on a user)

Concentration Inequalities

Hoeffding's Inequality: Sample mean concentrates around true mean exponentially fast. For $i \in \{1:n\}$, let $X_i \in [a, b]$ be independent RVs with mean μ. Then

$$\Pr\left[\underbrace{|\overline{X}_n - \mu|}_{\text{Accuracy}} > \varepsilon\right] \leq 2 \exp\left(-\frac{2n\varepsilon^2}{(b-a)^2}\right)$$

where $\overline{X}_n \coloneqq \frac{1}{n} \sum_{i=1}^{n} X_i$. Prob of large deviation ($> \varepsilon$) decays exponentially with n.

Chernoff-Hoeffding: Equivalent form, setting δ instead of ε. showing confidence radius. For fixed failure prob δ, estimated means concentrate in radius shrinking as $\sqrt{n}$:

$$\Pr\left[|\overline{X}_n - \mu| > (b-a)\sqrt{\frac{\log \frac{2}{\delta}}{2n}}\right] \leq \delta$$

Deviation; Confidence Radius (shrinks with $\sqrt{n}$); Fixed failure prob

Both equivalent; Hoeffding shows tail probability, Chernoff-Hoeffding shows confidence radius

Hack: When you only have a one-sided tail, you can decompose the absolute value into one-sided tails

[1] The t subscript on r is only meant to indicate the actual reward received at that time(?) Not that the reward changes with time.

[2] Delta is called the "Gap"

[3] How to show exploiation error:

- Let confidence radius

$$r(a) = \sqrt{2 \ln(n)/K}$$

-

Post-Class Notes

Hierarchical Reinforcement Learning (HRL)

HRL Overview

Promises of HRL [Link]

1. Long-term credit assignment (faster learning and better generalization)
2. Structured exploration (sub-policies not primitive actions)
3. Transfer learning (hierarchy = different knowledge = better transfer)

Learn multiple policies operating at different levels of abstraction. Multiple frameworks for HRL. One involves thinking of them as options @hrl-options. Another involves thinking about them temporally @hrl-temporal. Others.

Options Framework: Option contains three components (better explained in link, but did not read that first)

- Initiation Set: When option can be invoked
- Policy: Actions to be taken
- Termination Condition: When option should end
- Option can be primitive (meaning base-level actions.)

1. **Subgoal Discovery**: Idk somehow helps with this
2. **Reward Shaping**: Assign rewards at different levels of the hierarchy.

Diagram

Temporal Framework: Macro-actions consisting of sequences of lower-level actions (primitive action is lowest level). [Image]

Hierarchical DQN (H-DQN)

Two levels:

1. High-Level Controller:[4] Selects subgoals.
2. Low-level Controller

Model-Specific

- **Dreamer**: Model-based RL
- **PPO**: Policy-based RL (Actor-Critic)
- **Rainbow**: Value-based RL (DQN-Enhanced)
- **Plan2Explore**: Model-based Exploration (Intrinsic Reward, often Plan2Explore-DreamerV{X})
- **RND**: Exploration Bonus (Intrinsic Reward Signal Strategy, often RND-PPO)

Research Techniques

- **Ablation**: Systematically remove or modify components of a model to assess their individual contributions to overall performance.

[4] Aka Meta-Controller