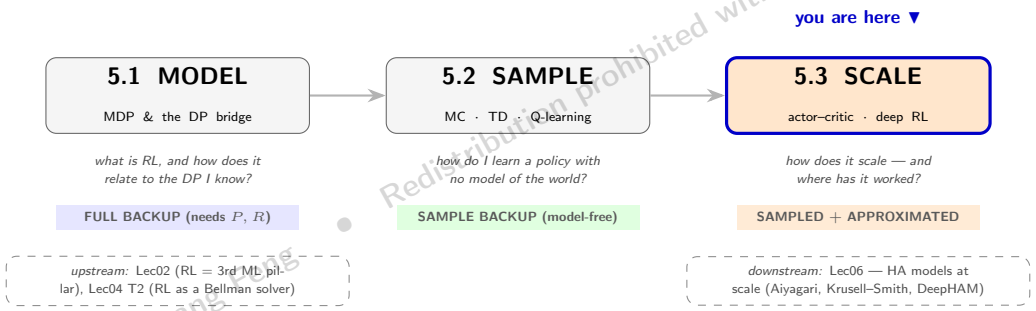
AI for Economic Research: Dynamic Models, Language, and Agents
Lecture 5.3: Actor-Critic Methods, Applications, and the AlphaManager Case

Zhigang Feng

2026-07-08 01:00:22

Lecture 5 in One Map — Act 5.3

Acts 5.1–5.2 executed departure #1 (sample the backup) — but still store values in tables. This act executes departure #2: approximate the function — neural networks for V, Q, π — and shows where the recipe has worked.



The internal flow of this act: tables → **function approximation** *(the deadly triad, DQN) →* **actor–critic** *and the policy-gradient family (A2C, DDPG, PPO, SAC) →* **applications** *(AlphaGo, RLHF) → two economic cases:* **AlphaManager** *(corporate finance) and* **Krusell–Smith** *(heterogeneous agents) → synthesis and the hand-off to Lec06.*

Actor–Critic Methods

From tables to neural networks — and how to make them stable

Motivation: Beyond Tabular Methods

Recall: Lec04 T2 previewed actor-critic as a Bellman solver; here we formalize it as the policy-gradient family with convergence theory and the A2C / A3C / PPO / DDPG / SAC variants.

Tabular RL:

- Stores a value for each *discrete* state (or state–action) pair.
- Infeasible in **large or continuous** state spaces.

Function Approximation:

- Replace tables with **parameterized models** (e.g., neural networks).
- Generalize from observed states to unseen states.

Continuous Action Spaces:

- Economic decisions are typically **continuous**: how much to consume, invest, price.
- Cannot enumerate all actions $\Rightarrow$ need a **continuous policy** $\pi_\theta(a|s)$.
- Actor-critic methods handle this naturally.

From Tables to Function Approximation

Replace the table $V(s)$ / $Q(s,a)$ with a parameterized function $V_w(s)$ — linear $w^\top \phi(s)$, then neural.

- **Semi-gradient TD(0):** $w \leftarrow w + \alpha\, \delta_t\, \nabla_w V_w(s_t)$, with the bootstrap target held fixed.
- **Why:** parameters **generalize** across states; storage grows with dimension, not as m^d — though this *mitigates*, not *eradicates*, the curse.
- **Shared with DP:** the same projection idea economists already use (Chebyshev, splines); the neural net is its nonlinear generalization.

The deadly triad (Sutton & Barto): **off-policy** + **function approximation** + **bootstrapping**, together, can *diverge* — one overestimate feeds its own target and spreads via generalization. Tabular RL never had this; the classical contraction guarantees no longer apply.

DQN: Making Neural Q-Learning Work

Mnih et al. (2015), *Nature*, "Human-level control through deep RL"

Plugging a neural net $Q_\theta(s, a)$ straight into Q-learning **ignites the triad** (off-policy + NN + bootstrap) — it oscillates or diverges. **DQN** stabilizes it with two engineering patches:

- **Experience replay:** store transitions in a buffer, sample mini-batches — breaks temporal correlation, restores (approx.) i.i.d. for SGD, reuses data.
- **Target network** θ^-: a *lagged* copy used in the target $y_t = r_{t+1} + \gamma \max_{a'} Q_{\theta^-}(s_{t+1}, a')$, synced every C steps — freezes the bootstrap loop so the net fits a *stable* target.

Patches, not a principled fix — DQN still needs careful tuning. And $\max_{a'}$ needs enumerable actions, which breaks for continuous controls ⇒ the policy-gradient / actor-critic route next.

Actor-Critic: The Architecture

Two sets of parameters, trained jointly:

Actor (Policy π_θ):

- Parameterized by θ (e.g., neural network weights).
- Produces actions: $a \sim \pi_\theta(a|s)$.
- Updated via **policy gradient**.

Critic (Value Function V_ϕ or Q_ϕ):

- Parameterized by ϕ.
- Estimates expected return: $V^\pi(s;\phi)$ or $Q^\pi(s,a;\phi)$.
- Provides feedback ("critique") to guide the actor.
- Updated via **TD learning**.

Why combine? Policy methods handle continuous actions; value methods reduce variance. Actor-critic merges both for efficient learning.

Policy Gradient Theorem

Sutton, McAllester, Singh & Mansour, *NIPS* 1999 (proc. 2000)

Objective: $J(\theta) = \mathbb{E}_{\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t r_t]$.

Policy Gradient Theorem:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim d^{\pi_\theta},\, a\sim\pi_\theta}\Big[\nabla_\theta \log \pi_\theta(a|s)\, Q^{\pi_\theta}(s,a)\Big].$$

- The **log-derivative trick**: we only differentiate $\log \pi_\theta(a|s)$.
- Multiplying by $Q^{\pi_\theta}(s,a)$ emphasizes actions with higher returns.
- In practice, the **critic** approximates Q (or V) with parameters ϕ.

Variance reduction: Replace $Q^\pi(s,a)$ with the **advantage**:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s).$$

Subtracting the baseline $V^\pi(s)$ keeps the estimate unbiased but lowers variance.

REINFORCE: the Score-Function Estimator

The environment drops out. Since $\nabla_\theta \log p_\theta(\tau) = \sum_t \nabla_\theta \log \pi_\theta(a_t|s_t)$, the transition terms *vanish* — the policy gradient is **model-free**:

$$\nabla_\theta J(\theta) = \mathbb{E}\Big[\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t)\,\big(G_t - b(s_t)\big)\Big].$$

- **REINFORCE** (Williams 1992): sample a trajectory, ascend this gradient. Unbiased, but **high variance**.
- **Baseline** $b(s_t) = V(s_t)$: keeps the gradient unbiased while cutting variance — exactly the **control-variate** trick from GMM. The bracket becomes the **advantage** $A_t \approx Q - V$.
- **Constraints:** a **masked softmax** sends infeasible-action logits to $-\infty$, so π_θ respects the budget set.

Actor-Critic Algorithm

At each time step:

1. **Actor** selects action: $a_t \sim \pi_\theta(\cdot|s_t)$.
2. Environment returns (r_{t+1}, s_{t+1}).
3. **Critic update** (TD learning on ϕ):

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}; \phi) - V(s_t; \phi)$$

$$\phi \leftarrow \phi + \alpha\, \delta_t\, \nabla_\phi V(s_t; \phi)$$

4. **Actor update** (policy gradient on θ):

$$\theta \leftarrow \theta + \beta\, \delta_t\, \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Intuition:

- $\delta_t > 0$: outcome was **better** than expected $\Rightarrow$ increase probability of a_t.
- $\delta_t < 0$: outcome was **worse** than expected $\Rightarrow$ decrease probability of a_t.

Why Actor-Critic for Economics?

Large / Continuous State Spaces:

- Capital stocks, prices, wealth distributions — high-dimensional and continuous.
- Function approximation with neural networks is essential.

Continuous Action Spaces:

- Monetary policy rates, consumption levels, investment amounts.
- Actor-critic handles continuous actions via a learned policy $\pi_\theta(a|s)$.

Efficiency and Stability:

- Critic reduces variance in policy updates.
- More data-efficient than pure policy gradient or pure value-based methods.

Already seen in Lec04: The simple actor-critic applied to the optimal growth model (policy NN + value NN trained jointly).

One Ladder: MC Control → SARSA → Q → REINFORCE → AC

1. **MC control** — full-return evaluation; high variance, needs episode end.
2. + one-step bootstrap ⇒ **SARSA** (on-policy TD control).
3. + $\max$ / off-policy ⇒ **Q-learning** (targets the optimum).
4. + direct policy gradient ⇒ **REINFORCE** (continuous actions; high variance).
5. + a TD critic as the baseline ⇒ **Actor-Critic** (low-variance gradient).

Actor-Critic ≈ online two-step estimation. The **critic** is the first-stage *nuisance* parameter (the value); the **actor**, the second-stage structural parameter (the policy) — the rolling version of the two-step M-estimator economists know (Newey–McFadden 1994), interleaved rather than waiting for stage one to converge.

Advanced Actor–Critic Algorithms

A2C/A3C, DDPG, PPO, SAC — and how to choose among them

A2C / A3C: Advantage Actor-Critic

A3C: Mnih et al. (2016), *ICML*, "Asynchronous Methods for Deep RL"; A2C: OpenAI Baselines (2017)

Key idea: Use the advantage function $A_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ in the policy gradient:

$$\nabla_\theta J(\theta) \approx \sum_t A_t \, \nabla_\theta \log \pi_\theta(a_t|s_t).$$

A3C (Asynchronous) — the original method (Mnih et al. 2016):

- Multiple **workers** run parallel environment instances.
- Workers update **asynchronously** — faster but gradients may become stale.

A2C (Synchronous) — the synchronous variant popularized by OpenAI Baselines (2017):

- Workers synchronously send gradients to a central learner — simpler and often more stable.

For economists: parallel workers = running multiple economy simulations simultaneously, each feeding back to improve a shared policy.

DDPG: Deep Deterministic Policy Gradient

Lillicrap et al. (2016), *ICLR*, "Continuous Control with Deep RL"

For continuous action spaces with a deterministic policy $\mu_\theta(s)$:

- **Actor:** $a = \mu_\theta(s)$ — deterministic mapping from state to action.
- **Critic:** $Q_\phi(s, a)$ — approximates expected return for (s, a).

Policy gradient (chain rule through the critic):

$$\nabla_\theta J \approx \frac{1}{N} \sum_i \nabla_a Q_\phi(s_i, a)\big|_{a=\mu_\theta(s_i)} \cdot \nabla_\theta \mu_\theta(s_i).$$

Exploration: Add noise to actions: $a_t = \mu_\theta(s_t) + \text{noise}$.

Economic applications:

- Continuous portfolio allocation adjustments.
- Optimal consumption/investment with continuous controls.

PPO: Proximal Policy Optimization

Schulman, Wolski, Dhariwal, Radford & Klimov (2017), arXiv:1707.06347

Problem: Large policy updates can destabilize learning.

Solution: Clip the policy ratio to prevent drastic changes:

$$L^{\text{clip}}(\theta) = \mathbb{E}_t\left[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}A_t,\ \text{clip}\left(\frac{\pi_\theta}{\pi_{\theta_{\text{old}}}},\ 1-\epsilon,\ 1+\epsilon\right)A_t\right)\right].$$

- $\epsilon \approx 0.2$: bounds how much the policy can change per update.
- If $A_t > 0$ (good action): increase its probability, but not too much.
- If $A_t < 0$ (bad action): decrease its probability, but not too much.

Why popular? Simple, stable, good performance across diverse tasks. Widely used in training LLMs via RLHF (Reinforcement Learning from Human Feedback).

SAC: Soft Actor-Critic

Haarnoja, Zhou, Abbeel & Levine (2018), *ICML*, "Soft Actor-Critic"

Key innovation: Add **entropy regularization** to the objective.

$$J_{\text{SAC}} = \mathbb{E}\left[\sum_t \gamma^t \big(r_t + \alpha\, \mathcal{H}(\pi(\cdot|s_t))\big)\right],$$

where $\mathcal{H}(\pi) = -\sum_a \pi(a|s) \log \pi(a|s)$ measures policy "spread."

- α large $\Rightarrow$ more exploration (high-entropy policy).
- α small $\Rightarrow$ more exploitation (near-deterministic).
- Prevents premature convergence to local optima.

Practical features: Twin Q-networks (reduce overestimation), soft target updates (stability), stochastic policy (reparameterization trick).

When to use: Uncertain environments requiring robust exploration.

Choosing an Algorithm: Summary

Algorithm	Key Feature	Action Space	Best For
Q-Learning	Off-policy, tabular	Discrete	Small problems
A2C/A3C	Advantage + parallel	Both	Parallel simulations
DDPG	Deterministic policy	Continuous	Single best action
PPO	Clipped updates	Both	Stable improvements
SAC	Entropy regularization	Continuous	Robust exploration

For economists solving macro models:

- **Known model, continuous controls:** Actor-critic (DDPG or PPO).
- **Unknown environment, need exploration:** SAC or PPO.
- **Discrete actions, small state space:** Q-learning suffices.
- **Training LLMs (RLHF):** PPO is the standard.

Deep RL in Practice: Four Honest Challenges

1. **Sample inefficiency:** 10^5–10^7 environment steps — vs. ∼200 quarterly macro observations.
2. **Instability:** same code, different random seed ⇒ different outcomes. Report ≥ 5 seeds with intervals.
3. **Exploration:** sparse rewards give near-zero gradient; needs entropy bonuses, curiosity, etc.
4. **Evaluation:** a good training curve $\neq$ a good policy — especially when training states aren't the test-relevant ones.

The workflow for economists: real macro data are too short and too broken by structural breaks (Volcker, 2008, 2020) to train on — so *train in a calibrated structural simulator* (DSGE / Aiyagari / HANK), *validate on a held-out period*, and benchmark against a small-grid DP solution. **RL's credibility inherits the calibrated model's credibility.**

Applications and Breakthroughs

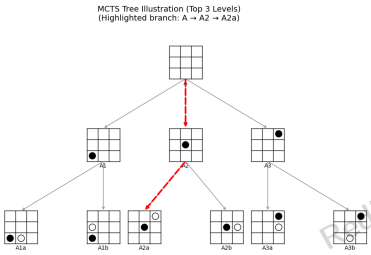
Where the recipe — neural approximation + sampling — has actually worked

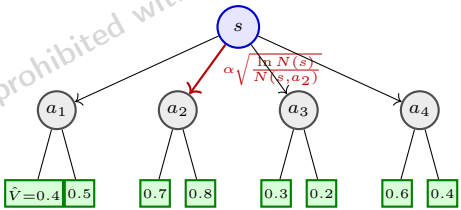
RL Breakthroughs: AlphaGo and Beyond

- **AlphaGo** (Silver et al. 2016, *Nature*, "Mastering the Game of Go. . ."): first program to defeat a human Go champion.
  - Combined deep NNs with RL (Monte Carlo Tree Search).
  - **Policy network:** estimates probability of each move.
  - **Value network:** predicts win/loss from current position.
- **AlphaZero** (Silver et al. 2018, *Science*, "A General RL Algorithm. . ."): learned Go, chess, and shogi entirely through **self-play**, without human knowledge.
- **RLHF for LLMs:** PPO used to align ChatGPT, Claude, etc. with human preferences.
- **Robotics:** RL enables robots to learn manipulation, navigation from trial-and-error.

Common thread (the payoff of T1's "why RL"): in a $\sim 10^{170}$-state game, *neural value+policy approximation* + *self-play* solve exploration and credit assignment — deep RL discovering strategies beyond human expertise. The same recipe scales to economic policy design.

How AlphaGo Works: MCTS + RL





Selection → *rollout* → *backup*. Same UCB logic as T2; deeper search for higher value.

Monte Carlo Tree Search: Connection to Economics

- **MCTS** evaluates moves by simulating future game progressions (*rollouts*).
- Four steps: **Selection** (UCB) $\rightarrow$ **Simulation** (rollout) $\rightarrow$ **Backpropagation** $\rightarrow$ **Move selection**.

MCTS vs. DP:

- DP enumerates **all** states and transitions.
- MCTS selectively simulates **promising branches**.
- Both update value estimates from future returns (Bellman-like).
- Key difference: MCTS works when $P(s'|s,a)$ is **not fully known**.

For economists: MCTS is like running targeted "what-if" simulations of policy scenarios, focusing computational effort on the most relevant paths through the economy.

Case Study — AlphaManager

Offline RL for corporate finance: Campello, Cong & Zhou (2023)

AlphaManager: Offline RL for Corporate Finance

Campello, Cong & Zhou (2023), working paper, "AlphaManager: A Data-Driven-Robust-Control Approach to Corporate Finance"

Campello, Cong & Zhou (2023) develop an AI-assisted framework (DDRC) for corporate finance:

Three challenges addressed:

- **Curse of dimensionality:** managers' decisions in high-dimensional action/state space.
- **Dynamic feedback:** decisions interact with the economic environment; market reactions feed back.
- **Evolving environment:** data distributions shift as markets evolve.

Solution approach:

1. **Predictive Environment Module (PEM):** estimate $\Delta X_{t+1} = f(X_t, u_t) + \varepsilon_{t+1}$ from historical data using deep neural networks.
2. **Policy Module:** offline RL (policy gradient) to find optimal $g(X_t|\theta)$.
3. **Ambiguity aversion:** penalize actions where model ensemble disagrees, addressing distribution shift.

AlphaManager: Policy Gradient and Ambiguity

Optimization:

$$\max_{g(X_t|\theta)} \mathbb{E}_{t_0} \sum_{t=t_0}^{t_0+T} r(X_t, u_t) \quad \text{s.t.} \quad \Delta X_{t+1} = f(X_t, u_t) + \varepsilon_{t+1},\ u_t = g(X_t|\theta).$$

Ambiguity via Boosting:

- Train ensemble $\{f^i\}_{i=1}^I$ with different initializations.
- Measure disagreement: $\text{BoostingError}(X_t, u_t)$.
- Penalize high-disagreement regions:

$$\sum_t \min_i r^i - \delta \cdot \max\{0, \max_t \text{BoostingError} - \theta(X_{t_0})\}.$$

Key insight: When $\delta \to \infty$, the agent becomes infinitely ambiguity-averse — it avoids novel, uncertain regions. Balancing δ is analogous to the exploration–exploitation tradeoff.

AlphaManager: Data, Robustness, Consistency Discipline

Data sources (offline, observational — no environment to interact with):

- **BoardEx** (governance covariates), **Execucomp** (managerial incentives), **Compustat** (state X_t: leverage, cash, payouts, ...).

Policy gradient under model ambiguity:

- Reward $r(X_t, u_t)$ encodes the corporate objective (e.g., long-run firm value).
- Ambiguity penalty δ makes the policy **robust** to misspecified f and r.
- Closely related to **robust control** (Hansen & Sargent) — familiar territory.

Three consistency checks that turn offline RL into a credibility-bearing artifact:

1. **Cross-validation:** hold-out *firms* — guards against firm-FE leakage.
2. **Out-of-sample:** hold-out *years* — guards against regime drift (pre/post-GFC).
3. **Interpretability subsamples:** actions on regulator-friendly slices (small / leveraged firms) checked against domain expectations.

The same triangulation logic resurfaces in Lec09 T5 on agent-assisted research — multi-agent debate as a consistency check on LLM-generated economic reasoning.

AlphaManager: Consistency and Extensions

Consistency question:

- PEM estimates $f(X_t, u_t)$ from historical data.
- Optimal policy g^* (given f) changes behavior $\Rightarrow$ new data distribution.
- Will the estimated f remain valid under the new policy?

Possible extension — toward full RL:

1. Use solved $g(X|f)$ to generate new data via bootstrap.
2. Mix new (out-of-sample) data with historical data.
3. Re-estimate $\hat{f}$ with a learning rate for new observations; repeat until f and $\hat{f}$ converge.

This moves from **offline RL** (fixed dataset) toward **online RL** (iterative environment learning) — improving robustness and discovering strategies invisible in historical data alone.

Forward pointer: Lec09 T5 returns to consistency mechanisms in agent-assisted research — Du et al. (2023) on multi-agent debate; Irving, Christiano & Amodei (2018) on AI safety via debate. AlphaManager's checks are a domain-specific instance.

Application — Krusell–Smith via Deep RL

The heterogeneous-agent workhorse, solved with a deep actor–critic (the bridge to Lec06)

Application: Krusell–Smith with Deep Actor-Critic

Krusell & Smith (1998); Feng, Han, Sargent & Zhu (2025)

The hard part: in a heterogeneous-agent model the **wealth distribution Φ is a state** — infinite-dimensional, and its law of motion $\Phi' = \mathcal{T}(\Phi)$ has no closed form.

- **Histogram representation:** bin assets into shares $\widehat{f}_a = [x_1, \ldots, x_N]$, $\sum_i x_i = 1$ — a finite vector a neural net can read.
- **Actor** $\pi_\theta(a' \mid a, z, Z, \widehat{f}_a)$ = household savings policy; **Critic** $V_w(\cdot)$ = household value.
- **Prices clear internally:** $K = \sum_i x_i \bar{a}_i \Rightarrow (r, w)$ from the firm's FOCs each period — no external forecast-rule iteration (the KS shortcut).

Three properties tame the dimensionality:

- NN first-layer parameters grow **linearly** in the input dimension (not m^d).
- Monte-Carlo simulation concentrates samples on the **ergodic set** — where households actually are.
- The neural critic gives **grid-free** interpolation over the simulated "point cloud."

The AlphaGo lesson, applied to macro: *neural value+policy approximation + sampling* reaches models classical grid-based DP cannot — but the curse is **mitigated, not eradicated**, and credibility still rests on the calibrated model.

Summary

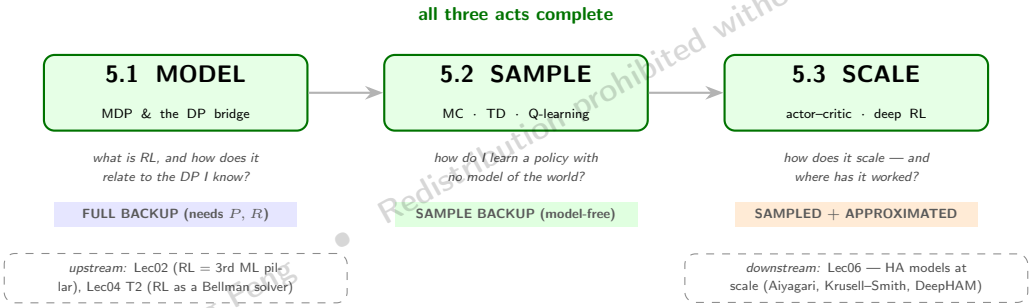
Three acts, one backbone — and the one caveat that keeps it credible

Lecture Summary

1. **RL = DP + Learning**: same Bellman foundations, but replaces full expectation backup with **sample-based** updates.
2. **Monte Carlo**: learns from complete episodes; model-free but slow and high-variance.
3. **TD / Q-Learning**: incremental updates after each step; balances bias and variance; handles continuing tasks.
4. **Actor-Critic**: combines policy optimization (actor) with value estimation (critic); handles continuous actions and large state spaces.
5. **Advanced algorithms** (PPO, SAC, DDPG): stability, entropy regularization, deterministic policies for different use cases.
6. **For economists**: RL enables solving dynamic models that are too high-dimensional for classical DP, and environments where transition probabilities are unknown or endogenous.

The Whole Lecture in One Map

Three acts, one backbone. The Bellman / GPI skeleton never changed — only how we filled the expectation did.



5.1 MODEL: *known $P, R \Rightarrow$ full backup (classical DP).* **5.2 SAMPLE**: *no model $\Rightarrow$ sample backup (MC, TD, Q-learning).* **5.3 SCALE**: *sampled + function-approximated $\Rightarrow$ deep RL (actor–critic, DQN) for continuous, high-dimensional economies.*

What Economists Can Borrow from RL — and the One Caveat

One skeleton, two ways to fill the expectation. DP and RL share the Bellman/GPI backbone; DP is RL's *structural* special case (known P, small $|\mathcal{S}|$). So we **extend** DP, not abandon it — and RL is *not* merely a last resort ("deep DP" helps even when P is known).

- **Sample backup** $\rightarrow$ solve high-dimensional / distribution-valued models.
- **Function approximation** $\rightarrow$ scale (a *shared* tool; the threat is the deadly triad).
- **Actor-critic** $\rightarrow$ the online two-step / GMM estimator economists already know.
- **Off-policy** + **importance sampling** $\rightarrow$ counterfactual policy evaluation (coverage = overlap).

The caveat that makes it principled: *RL scales the solution, not the identification.* Train in a calibrated structural model, validate out of sample. "Isn't RL just DP plus sampling?" — yes in *structure*; no in *reach* (the information and computation dimensions).

The RL Landscape for Economists

Method	Model Required?	Update Style
Value Function Iteration (DP)	Yes (full model)	Full backup
Monte Carlo	No	End of episode
TD(0) / SARSA	No	Every step
Q-Learning	No	Every step (off-policy)
Actor-Critic	No	Every step (policy + value)
Deep RL variants		
DQN / DDPG / PPO / SAC	No	Mini-batch gradient

As models grow in complexity, RL becomes not a replacement for DP, but its necessary generalization.

Key Takeaways

- Economists already know the theory behind RL — it's the **Bellman equation**.
- What RL adds: the ability to **learn** value functions and policies from **simulated experience** rather than solving them analytically.
- The exploration–exploitation tradeoff has direct economic interpretations (R&D, pricing, portfolio choice).
- Actor-critic methods are the workhorse for solving continuous-action dynamic models with deep learning.
- Modern breakthroughs (AlphaGo, RLHF for LLMs) rest on the same foundations covered today.

The one caveat that keeps it credible: RL scales the *solution*, not the *identification*. Train inside a calibrated structural model, validate out of sample — RL's credibility inherits the model's.

Bridge to Lec06: The Toolkit Doesn't Change — the State Space Does

Act 5.3 scaled RL from tables to neural networks. Lec06 turns exactly this machinery onto heterogeneous-agent models that were intractable a decade ago.

- **Lec06 T1 — Aiyagari computation:** the canonical HA workhorse gets the RL treatment; classical projection + iteration vs. DNN-based global solvers.
- **Lec06 T2 — Why ML for HA + global DNN:** Euler-equation training as a direct application of value/policy nets (Lec04 T3) at HA scale.
- **Lec06 T3 — Krusell–Smith via DeepHAM:** deep value-and-policy learning — the actor–critic family of Act 5.3 at high-dimensional aggregate states (Han, Yang & E, 2021; *Quantitative Economics* 2026).
- **Lec06 T4 — Deep Equilibrium Nets (DEQN) + OLG** with aggregate uncertainty (Azinovic, Gaegauf & Scheidegger, 2022).
- **Lec06 T5 — Mean-Field Games (HJB + KFE):** RL for the continuum-of-agents limit.

Hands-on (Session-4 lab): you build one of these deep HA solvers yourself — Krusell–Smith (1998) via Deep Equilibrium Nets in PyTorch.

Arc complete: 5.1 MODEL → 5.2 SAMPLE → **5.3 SCALE** $\Longrightarrow$ **Lec06***: heterogeneous agents at scale.*