Session 3.5

The Growth Model Across Algorithms



Zhigang Feng

© Zhigang Feng
Redistribution prohibited without permission

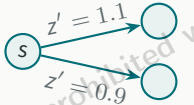
One Model, One Interface

the states, on the grid



one number per (k, z)

one period from s, after choosing k'

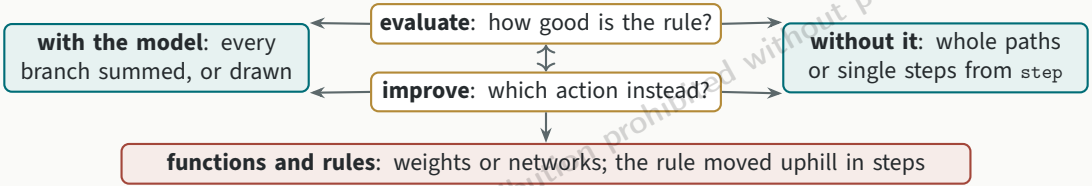


- **State** $s = (k, z)$: capital, and a productivity shock with two values that persists. **Action**: next period's capital k', or the share of output saved. **Reward**: $\ln c$.
- **Three sizes**: a 7×2 grid for the tables; continuous k with the same two shocks for the functions (Deck 3.4); Session 2's deterministic version for deep value iteration.
- **Two calls**: `branches` returns every z' with its probability, `step` one drawn z' and r. From Deck 3.2 on we pretend not to know P_z: learners call only `step`.
- **Every algorithm below learns a value or a rule from one of these two calls.**

Source: Session 3 lab, block A (the two calls); $\alpha = 0.36$, $\gamma = \beta = 0.95$, depreciation 0.10, $u = \ln c$, $z \in \{0.9, 1.1\}$ switching w.p. 0.2.

© Zhiheng Feng · Redistribution prohibited without permission

Every Algorithm Answers Two Questions



© Thiago Pena. Redistribution prohibited without permission

- Every algorithm makes four choices: what it **stores** (a table, a basis, a network); what stands in for the **expectation** over z' (a sum, one draw, a path); how many real periods its **target** counts before it trusts the stored estimate; how it **improves** the rule (a $\max$, ϵ-greedy, a gradient step).
- **Each page: on top what is stored and what one update reads; below, the update, and in bold the one change.**

Session 3.5

With the Model in Hand

Every branch of tomorrow summed with $P_{z'}$ or drawn: Decks 3.1 and 3.4



© Zhigang Feng Redistribution prohibited without permission

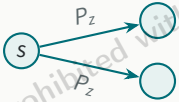
Value Function Iteration and Policy Iteration

stores



one number per (k, z)

one update reads, at every grid point



every z', weighted: `branches`

- **Stores** a table $V(k, z)$, one number per grid state. **Reads** u, f and P_z at every grid point: the whole model.
- **One update**, at every grid point, with $c = y(k, z) - k'$:

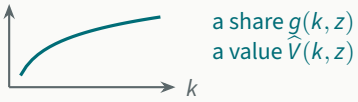
$$V(k, z) \leftarrow \max_{k'} \left\{ u(c) + \gamma \sum_{z'} P_z(z' \mid z)\, V(k', z') \right\}.$$

- **Improves** by that $\max$: $g(k, z)$ is the best k'; one backup, then improve. Policy iteration instead values the current rule all the way to convergence before improving it: Howard's method.
- **The benchmark: exact, and the answer key for every page after this.**

© Zhigang Feng • Redistribution prohibited without permission

The Simplest Actor–Critic (Deck 3.4)

stores



one update reads, at states drawn in a band



- **Stores** two functions of continuous k: a saving share $g(k,z)$ and a value $\widehat{V}(k,z)$. **Reads** the model whole, at states drawn afresh in a band.
- **One update** (critic): fit $\widehat{V}$ to the one-period backup

$$B(s) = u(c) + \gamma \sum_{z'} P_z(z' \mid z)\, \widehat{V}^{-}(k', z').$$

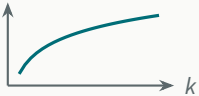
- **Improves** (actor) by one small step: save more where tomorrow's marginal value of capital beats today's marginal utility, less where it falls short. The max is taken in steps, with no search over k'.
- **The first change: no grid, no sweep, no search over k'; the model stays.**

Source: Session 3 lab, block B.

© Zhigang Feng
Redistribution prohibited without permission

Its Variants: One Drawn Branch, and the Walk

stores



the same two functions

one update reads



one drawn z'; u and f still read

- **Stores** the same two functions. **Reads** u, f and one drawn z', not every z' summed.
- **One update** (critic), toward one drawn branch: right on average, noisy on any one draw:

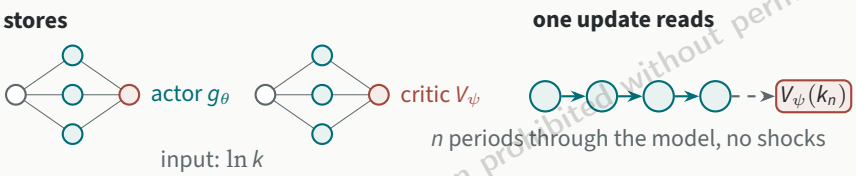
$$\widehat{B}(s) = u(c) + \gamma \widehat{V}^{-}(k', z'), \qquad z' \text{ drawn; the actor's step uses the same draw.}$$

- T **periods ahead**, the **walk**: run the rule N times for T periods from s and average, so the critic's target holds T real periods before the estimate.
- **The bridge: a run that ends on $\gamma^T \widehat{V}(s_T)$ is n-step TD; a run to the end is Monte Carlo.**

Source: Session 3 lab, block B.

© Zhigang Feng • Redistribution prohibited without permission

Deep Value Iteration (Deck 3.4)



- **Stores** two networks on continuous k: an actor g_θ (the share) and a critic V_ψ. **Reads** Session 2's deterministic model for n periods, at states drawn in a band.
- **Improves** first (actor): one gradient step up on J_n, through all n budget constraints:

$$J_n(k) = \sum_{t<n} \gamma^t u(c_t) + \gamma^n V_\psi(k_n).$$

- **One update** (critic): regress $V_\psi(k)$ on $J_n(k)$, recomputed and held fixed as a label.
- **The second change: both functions networks, both steps n periods deep; no theorem guarantees the pair.**

Source: Session 3 lab, block C.

© Zhigang Feng Registration prohibited without permission

Session 3.5

Without the Model

We pretend not to know P_z: whole paths, single steps, then functions and rules (Decks 3.2–3.4)



© Zhigang Feng · Redistribution Prohibited without permission

Monte Carlo Evaluation





© Zhigang Feng Redistribution prohibited without permission

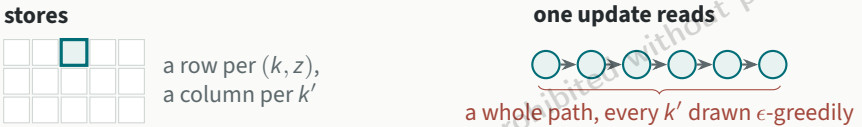
- **Stores** a table $\widehat{Q}(k, z, k')$ of running means, and visit counts C. **Reads** only what `step` returns: rewards and states.
- **One update**, after a whole simulated path under a rule g: each visited cell $x = (k, z, k')$ moves toward the discounted utility G that followed it,

$$\widehat{Q}(x) \leftarrow \widehat{Q}(x) + \big(G - \widehat{Q}(x)\big)/C(x).$$

- **Improves**: not yet; it values a fixed rule g. The luck of one whole path replaces the sum over z'.
- **The third change: simulation replaces integration.**

Source: Session 3 lab, block D.

Monte Carlo Control



- **Stores** $\widehat{Q}(k, z, k')$, visit counts. **Reads** whole paths, every k' drawn by the rule.
- **One update**: the path's returns averaged into $\widehat{Q}$, as on the last page. **Improves** after each path, **ϵ-greedy** in the new $\widehat{Q}$:

$$k' = \arg\max_{a} \widehat{Q}(k, z, a) \text{ with probability } 1 - \epsilon, \text{ a random feasible } k' \text{ otherwise.}$$

- **The fourth change: the rule improves from its estimates and keeps exploring.**

Source: Session 3 lab, block D.

© Zhigang Feng Redistribution prohibited without permission

TD(0)

stores



one number per (k, z)

one update reads



one draw from `step`: r and $s' = (k', z')$

Redistribution prohibited without permission

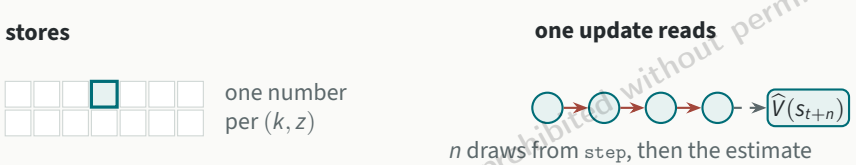
- **Stores** a table $\widehat{V}(k, z)$ for a fixed rule g, visit counts C. **Reads** one call of `step`.
- **One update**, after each call: today's reward plus the stored value of where it landed, against the stored value of where it was,

$$\widehat{V}(k, z) \leftarrow \widehat{V}(k, z) + \eta\big[r + \gamma \widehat{V}(k', z') - \widehat{V}(k, z)\big].$$

- **Improves**: not yet; it values g. The rest of the path is borrowed from $\widehat{V}$.
- **The fifth change: one real reward, then the estimate. No waiting for the path.**

© Zhigang Feng

Between One Step and the Whole Path: n-Step TD and TD(λ)



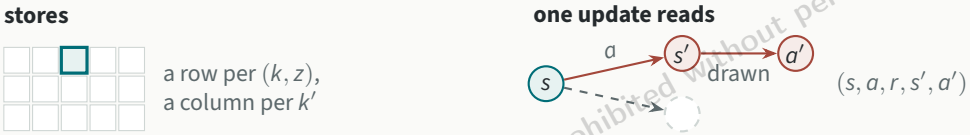
© Zhirang Feng · Redistribution prohibited without permission

- **Stores** $\widehat{V}(k, z)$; TD(λ) adds a memory of recent visits. **Reads** n transitions, or one at a time.
- **One update** (n-step), n periods after t: n real rewards, then the estimate,

$$\widehat{V}(k_t, z_t) \leftarrow \widehat{V}(k_t, z_t) + \eta \Big[\textstyle\sum_{j<n} \gamma^j r_{t+j+1} + \gamma^n \widehat{V}(k_{t+n}, z_{t+n}) - \widehat{V}(k_t, z_t) \Big].$$

- **TD(λ)** averages every n, with weights that fade geometrically, and does it as the path runs: each surprise is credited to the states visited recently, the latest most (Deck 3.3).
- **n and λ: one dial from TD(0) to Monte Carlo; turned up, less bias, more variance.**

SARSA



Redistribution prohibited without permission

- **Stores** $\widehat{Q}(k, z, k')$, and acts ϵ-greedily in it. **Reads** one transition and the next action, drawn: (s, a, r, s', a').
- **One update**: draw k'' at (k', z') from the ϵ-greedy rule, take it next, and set

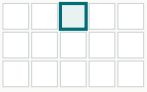
$$\widehat{Q}(k, z, k') \leftarrow \widehat{Q}(k, z, k') + \eta\big[r + \gamma \widehat{Q}(k', z', k'') - \widehat{Q}(k, z, k')\big].$$

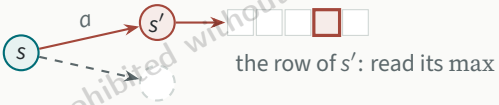
- **Improves** at every step: the rule follows the latest $\widehat{Q}$. It learns the value of the trembling rule it follows; to reach g^*, the trembles must fade slowly (Deck 3.3).
- **The sixth change: control on every step, valuing the rule it follows.**

© Zhuoer Feng

Source: Session 3 lab, block E.

Q-Learning





- **Stores** $\widehat{Q}(k, z, k')$, and acts ϵ-greedily in it. **Reads** (s, a, r, s'); the $\max$ is read off the row of (k', z'), nothing simulated.
- **One update**:

$$\widehat{Q}(k, z, k') \leftarrow \widehat{Q}(k, z, k') + \eta\big[r + \gamma \max_{k''} \widehat{Q}(k', z', k'') - \widehat{Q}(k, z, k')\big].$$

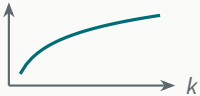
- **Improves** inside the target: the $\max$ values tomorrow under the greedy rule, so $\widehat{Q} \to Q^*$ for any fixed $\epsilon > 0$: value function iteration, one sampled transition at a time.
- **The seventh change: one line. The** $\max$ **replaces the drawn next action.**

Source: Session 3 lab, block E.

© Zhuoran Feng Redistribution prohibited without permission

Linear TD

stores



$\psi^\top \phi(k, z)$:
five weights

one update reads



one draw from `step`:
r and $s' = (k', z')$

- **Stores** five weights ψ: $\widehat{V}_\psi(k, z) = \psi^\top \phi(k, z)$, a weighted sum of five fixed shapes in k and z (Deck 3.4). **Reads** one transition.
- **One update**, moving every state that shares those shapes:

$$\delta = r + \gamma \widehat{V}_\psi(k', z') - \widehat{V}_\psi(k, z), \qquad \psi \leftarrow \psi + \eta\, \delta\, \phi(k, z).$$

- **Improves**: not yet; it values a fixed rule. Where the shapes cannot bend, the fit is wrong at many states at once (Deck 3.4).
- **The eighth change: five weights stand in for the table.**

Source: Session 3 lab, block F.

© Zhigang Feng • Redistribution prohibited without permission

DQN

stores



$\widehat{Q}_\psi(k, z, \cdot)$: one output per k'; a lagged copy ψ^-; a buffer

one update reads



past (s, a, r, s') in a buffer: a minibatch, drawn

- **Stores** a network $\widehat{Q}_\psi$: input (k, z), one output per k'; a lagged copy ψ^- of its weights; and a buffer of past transitions. **Reads** a minibatch drawn from the buffer.
- **One update**, one gradient step on the minibatch:

$$\min_{\psi} \left(r + \gamma \max_{k''} \widehat{Q}_{\psi^-}(k', z', k'') - \widehat{Q}_\psi(k, z, k')\right)^2, \quad \text{the label from the lagged copy.}$$

- **Improves** as Q-learning does: ϵ-greedy acts and feeds the buffer; ψ is copied into ψ^- every so many steps, so the labels stand still between copies.
- **The ninth change: learned features, held still by a lagged target and replay.**

Source: Session 3 lab, block F.

© Zhiguo Feng Redistribution prohibited without permission

REINFORCE

stores, at each state



$g_\theta(k' \mid k, z)$

one update reads



a whole path from `step`, under g_θ

- **Stores** a rule $g_\theta(k' \mid k, z)$: at each state, a probability for every feasible k', started uniform; and a baseline $\widehat{V}(k, z)$. **Reads** whole paths.
- **One update**, after a whole path under g_θ, with G_t the discounted utility that followed period t:

$$\theta \leftarrow \theta + \eta \sum_t \gamma^t \nabla_\theta \ln g_\theta(k'_t \mid k_t, z_t) \left(G_t - \widehat{V}(k_t, z_t)\right).$$

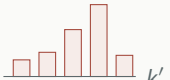
- **Improves** by that step: a k' that did better than the baseline becomes likelier, one that did worse less likely. No search over k', no model.
- **The tenth change: the $\max$ is taken in gradient steps on the rule.**

Source: Session 3 lab, block G.

© Zhigang Feng • Redistribution prohibited without permission

Actor–Critic Without the Model

stores



actor $g_\theta(k' \mid k, z)$
critic $\widehat{V}_\psi(k, z)$

one update reads



one draw from `step`:
r and $s' = (k', z')$

- **Stores** an actor $g_\theta(k' \mid k, z)$, probabilities over k', and a critic $\widehat{V}_\psi(k, z)$. **Reads** one transition; a restart at random now and then spreads the visits.
- **One update** (critic), then the actor, on the same transition and surprise:

$$\delta = r + \gamma \widehat{V}_\psi(k', z') - \widehat{V}_\psi(k, z),$$

$$\psi \leftarrow \psi + \eta_\psi \, \delta \, \nabla_\psi \widehat{V}_\psi(k, z), \qquad \theta \leftarrow \theta + \eta_\theta \, \delta \, \nabla_\theta \ln g_\theta(k' \mid k, z).$$

- **Improves** by the actor's step; each learner has its own step size.
- **The eleventh change: the TD error replaces the whole return.**

Source: Session 3 lab, block G.

© Zhigang Feng
Redistribution prohibited without permission

Session 3.5

Side by Side

Every algorithm against the same questions, and when to use which



© Zhigang Feng Redistribution prohibited without permission

With the Model, Side by Side

algorithm	stores	tomorrow, by	looks ahead	improves by
value function iteration	a table V	every z', summed	one period	a $\max$ over k'
policy iteration	a table V	every z', summed	until the rule's value settles	a $\max$ over k'
the simplest actor–critic (3.4)	two functions	every z', summed	one period	a derivative step
one drawn branch	two functions	one z', drawn	one period	a derivative step
the walk	two functions	the rule run N times	T periods	a derivative step
deep value iteration	two networks	no shocks to draw	n periods, both	a step through n periods

© Anirang Feng · Redistribution prohibited without permission

- All read u and f. The first three sum over P_z, the next two draw from it, and deep value iteration has no shocks.
- **Down the rows: a draw for the sum, functions for the table, steps for the $\max$.**

Without the Model, Side by Side

algorithm	stores	the target uses	looks ahead	improves by
Monte Carlo	a table $\widehat{Q}$	a whole path's return	to the end	ϵ-greedy, per path
TD(0); n-step, TD(λ)	a table $\widehat{V}$	real rewards, then $\widehat{V}$	1 or n periods	(it evaluates)
SARSA	a table $\widehat{Q}$	a drawn k''	one period	ϵ-greedy, per step
Q-learning	a table $\widehat{Q}$	the row's $\max$	one period	a $\max$ in the target
linear TD	five weights ψ	one reward, then $\psi^\top \phi$	one period	(it evaluates)
DQN	a network, a copy	the $\max$, from the copy	one period	a $\max$ in the target
REINFORCE	a rule θ, a baseline	a whole path's return	to the end	a step on the rule
actor–critic	a rule θ, a critic	one reward, then $\widehat{V}_\psi$	one period	a step on the rule

© Zhaoyang Feng. Unauthorized distribution prohibited without permission

- All read only (s, a, r, s') from `step`. We pretend not to know P_z; in Session 4 the distribution's law of motion is usually truly unknown to the household.
- **Down a column, what changed; along a row, what one algorithm is.**

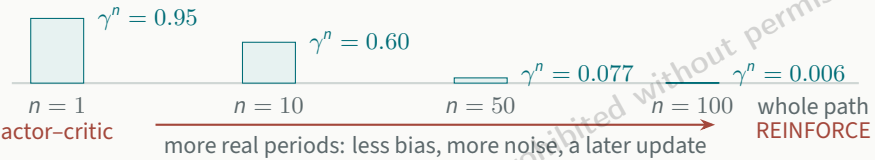
VFI, Q-Learning, Actor–Critic: Which Dynamic Programme Each Samples

	value function iteration	Q-learning	actor–critic
stores	a table $V(k, z)$	a table $\widehat{Q}(k, z, k')$	a rule g_θ and a critic $\widehat{V}_\psi$
values	V^*	Q^*, from any behaviour	V^g of today's rule, which moves
improves by	a $\max$: a jump	a $\max$ inside the target	a step on θ, pushed by δ
data	none: the model	any rule's path (off-policy)	today's rule's path (on-policy)
suits	a small grid, P_z known	a few discrete k'	a share in $(0, 1)$, many actions

- **Q-learning is VFI relaxed three ways**: one drawn z' for the sum, a step $\eta < 1$ for the replacement, the visited cell for the sweep. Restore all three and it is VFI.
- **Actor–critic is policy iteration done partially**: one TD step values today's rule, one gradient step improves it, so its critic chases a moving rule. Q-learning's target ignores behaviour; its data does not.
- **The $\max$ decides: a few k' to compare, Q-learning; a share in $(0, 1)$, actor–critic, climbing in steps.**

© Zhiwei Feng — Unauthorized distribution prohibited without permission

REINFORCE and Actor–Critic: One Dial

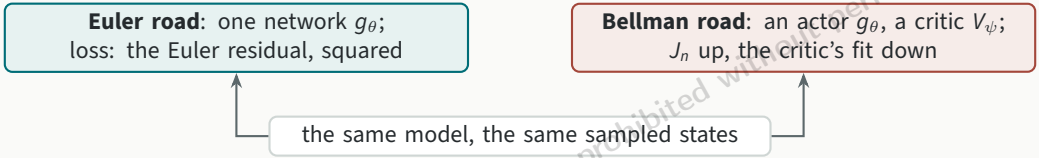


© Zhigang Feng. Unauthorized distribution prohibited without permission

- **The question.** Both move the rule by the score. Is actor–critic just a better REINFORCE?
- Credit k'_t with $A_t^{(n)} = \sum_{j<n} \gamma^j r_{t+j+1} + \gamma^n \widehat{V}(s_{t+n}) - \widehat{V}(s_t)$: $n = 1$ is actor–critic; to the path's end it is $G_t - \widehat{V}(s_t)$, REINFORCE with the critic as baseline.
- The critic's error enters scaled by γ^n (the bars); as a baseline alone it adds no bias.
- Each end loses somewhere: at half budget actor–critic collapsed on three seeds of five, REINFORCE on none; deep value iteration with $n = 1$ settled on a wrong rule. Practice averages over n.
- **One question, as for TD: how many real periods before trusting the critic.**

Source: Session 3 lab, blocks C and G (the half-budget pilot).

Euler or Bellman: Two Roads on Session 2's Model



Euler road: one network g_θ;
loss: the Euler residual, squared

Bellman road: an actor g_θ, a critic V_ψ;
J_n up, the critic's fit down

the same model, the same sampled states

- **Euler** (Session 2): drive $u'(c) - \gamma\, u'(c')\big[f'(k') + 1 - \delta_k\big]$ to zero at sampled states (f' the marginal product, δ_k depreciation). One network, a sharp condition, no value function.
- **Bellman** (Deck 3.4): two networks and no first-order condition, so a kinked value is no obstacle, and a value comes out for welfare comparisons. A discrete choice needs the actor of p. 19, which moves probabilities, not a derivative through the budget.
- **A decision, not a doctrine: take the Euler road when the first-order conditions are clean.**

Source: Session 3 lab, block C.

© Shigang Feng · Redistribution prohibited without permission

When Dynamic Programming, When Reinforcement Learning

	expectation computable	**expectation only simulable**
low dimension	DP: nothing beats it	tabular RL; coverage is the issue
high dimension	approximate DP: deep value iteration	deep RL's home ground

- Bottom left is where most structural economics lives: the model is written, the grid cannot be built. RL's sampled paths and fitted functions can win there too.
- In use: dynamic discrete choice, where Rust re-solves the Bellman equation inside the estimation; high-dimensional macro; interest-rate and tax rules.
- **The caveat: reinforcement learning scales the *solution*, not the *identification*. The data discipline lives in the calibration.**

© Zhiguo Feng. Unauthorized distribution prohibited without permission

RL's One Idea, and the Shared Toolkit

© Yiqiang Feng · Redistribution prohibited without permission

The substitution	Decks	also in DP, macro	What it costs
simulation for integration	3.2, 3.3	simulated moments	noise; states the paths rarely reach
stored functions for tables	3.4	projection, deep DP	approximation error; with a target built from itself and data from another rule, no guarantee
sampled states where the economy lives, for a sweep	3.1, 3.4	ergodic-set methods	blind to rare states
gradient steps for the search over actions	3.1, 3.4	Euler methods	step sizes and seeds; a pair that agrees can still be wrong
the price: the guarantees	3.1–3.4		an answer key; ranges over seeds

- **RL's own idea: the sampled transition its own rule generates. The rest is a toolkit shared with modern macro computation.** Still owed: an equilibrium, a government, a test.

Session 3.5

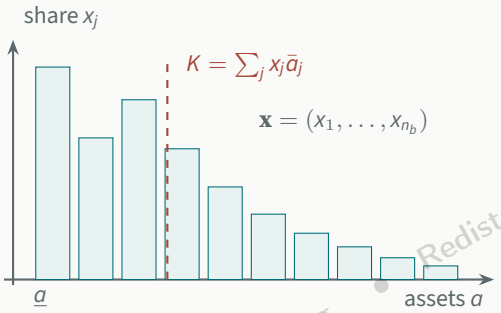
From One Household to an Economy

What Session 4 inherits: the distribution in the state



© Zhigang Feng · Redistribution prohibited without permission

Bridge (1): Put the Distribution in the State, as a Histogram

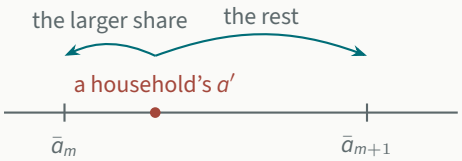


- The method starts by choosing what to carry of the wealth distribution. Krusell and Smith carry its mean. From here a is assets, Session 4's symbol; the choice is a'.
- Carry a **histogram** instead: the share x_j of households in each of n_b asset bins. No shape is assumed; the mass at the borrowing limit and the tail survive.
- Prices follow inside the model: K, then r and w from the firm's conditions.
- **The state is $n_b + 2$ numbers, which a network takes as one input vector.**

© Zhigang Feng Redistribution prohibited without permission

Source: Method of Feng, Han, Sargent & Zhu (2026, working paper), research in progress.

Bridge (2): an Analytic Transition Matrix, Not a Simulated Panel



its mass is split between
the two neighbouring bins,
the nearer one getting more

$\mathbf{x}_{t+1} = \mathbf{x}_t \, \mathbf{M}_g$: a matrix built from the rule

- A household's a' lands between two grid points; its mass is split between them by nearness. Summed over households and shocks: tomorrow's histogram, exact, as one matrix $\mathbf{M}_g$.
- It works under *any* rule, as Session 4's government needs, and changes smoothly with the rule; a simulated panel jumps when a household crosses a bin edge.
- **Deck 3.1's promise kept: Γ_g has no closed form, yet a period can always be run, here as one matrix product.**

© Zhigang Feng
Redistribution prohibited without permission

Source: The two-point split is Young's (2010); using it as the law of motion a learner trains against: Z. Feng, J. Han, T. Sargent & S. Zhu (2026, working paper), research in progress.

Bridge (3): the Household's Actor–Critic

- **The question.** A household in this economy must forecast prices, so the histogram $\mathbf{x}$ is part of its state. How does it solve its problem with the distribution as an input?
- **The same two learners**, both networks reading $\mathbf{x}$ as one input: a critic $V_\psi(a, z, \mathbf{x})$ and an actor $a' = g_\theta(a, z, \mathbf{x})$, trained at distributions drawn near the current one.
- **The critic's step**: regress V_ψ on the one-period backup at $(a', z', \mathbf{x}')$, the label from a frozen copy of the critic, refreshed at intervals.
- **The actor's step**: one gradient step up the same backup, through the budget constraint (Deck 3.4).
- **The distribution's motion is taken as given**: $\mathbf{x}' = \mathbf{x}\,\mathbf{M}_g$ enters as a number, not differentiated; one household cannot move it. Stop when Euler residuals are small and the pushed-forward aggregates settle.
- **Without results: Session 4 owns equilibrium, the government, validation.**

Source: Feng, Han, Sargent & Zhu (2026, working paper), research in progress.

© Jesse Perla · Redistribution prohibited without permission

Using Reinforcement Learning on Your Model: Five Decisions

decide	this session's growth model	Session 4's household
1 the state	(k, z); enlarge it until tomorrow depends on it alone (Deck 3.1)	assets, shock and the distribution: a histogram, moments, or a history
2 the action	grid: k', infeasible ones masked; continuous: a share between 0 and 1	next assets, from an actor network
3 reward, discount	$u(c)$; γ is the economist's β	the same
4 what it may read	the model: sum, draw, roll forward (3.1, 3.4); `step`: sample, explore (3.2–3.4)	the model, written; the law of motion rebuilt each date
5 the grade	an answer key, DP's g^*; seeds as a range	a benchmark where one exists; else residuals, aggregates

© Zixiang Feng · Redistribution prohibited without permission

- Decisions 1 and 4 are where macro departs: a distribution in the state, and a model written in full whose law of motion for it has no closed form (Deck 3.1).
- **Decide these five first: decision 4 picks the half of this deck you need; decision 5 tells you whether it worked.**

What Session 4 Inherits

Built in Session 3	Used in Session 4
an answer key before any learner	the stationary equilibrium by classical methods first
actor and critic, model known	household policy and value, distribution as an input
gradients through the model	only the government differentiates through $\mathbf{x}'$
a grade in units of consumption	consumption-equivalent welfare across tax regimes
small residual, not a right answer	judge a solution by its simulated aggregates

- **Every algorithm this session ran on a growth model chosen so that an answer key exists**: a printable grid, a fine dynamic programme, a closed form. The point was to see how each one works and how each is graded; none of it is a finding about an economy.
- **Session 4 starts on the problems these tools are for: Aiyagari, Krusell–Smith, transitions, taxation — where no answer key exists and a solution is judged by its simulated aggregates.**

© Zhigang Feng Redistribution prohibited without permission