Anton Nesterov · an@vski.sh · July 23, 2026

Reflex Bandit: An Agent with Drift Detection for Active Exploitation Regimes

Machine Learning · Bandits · Concept Drift

An agent whose sensors are noiseless but whose meaning must be learned. When the world's shape drifts, ensemble disagreement is not enough — the agent has to watch its own prediction error. A tree-ensemble contextual bandit with two-timescale memory and an EWMA drift detector, in a regime where value-function RL is the wrong tool.

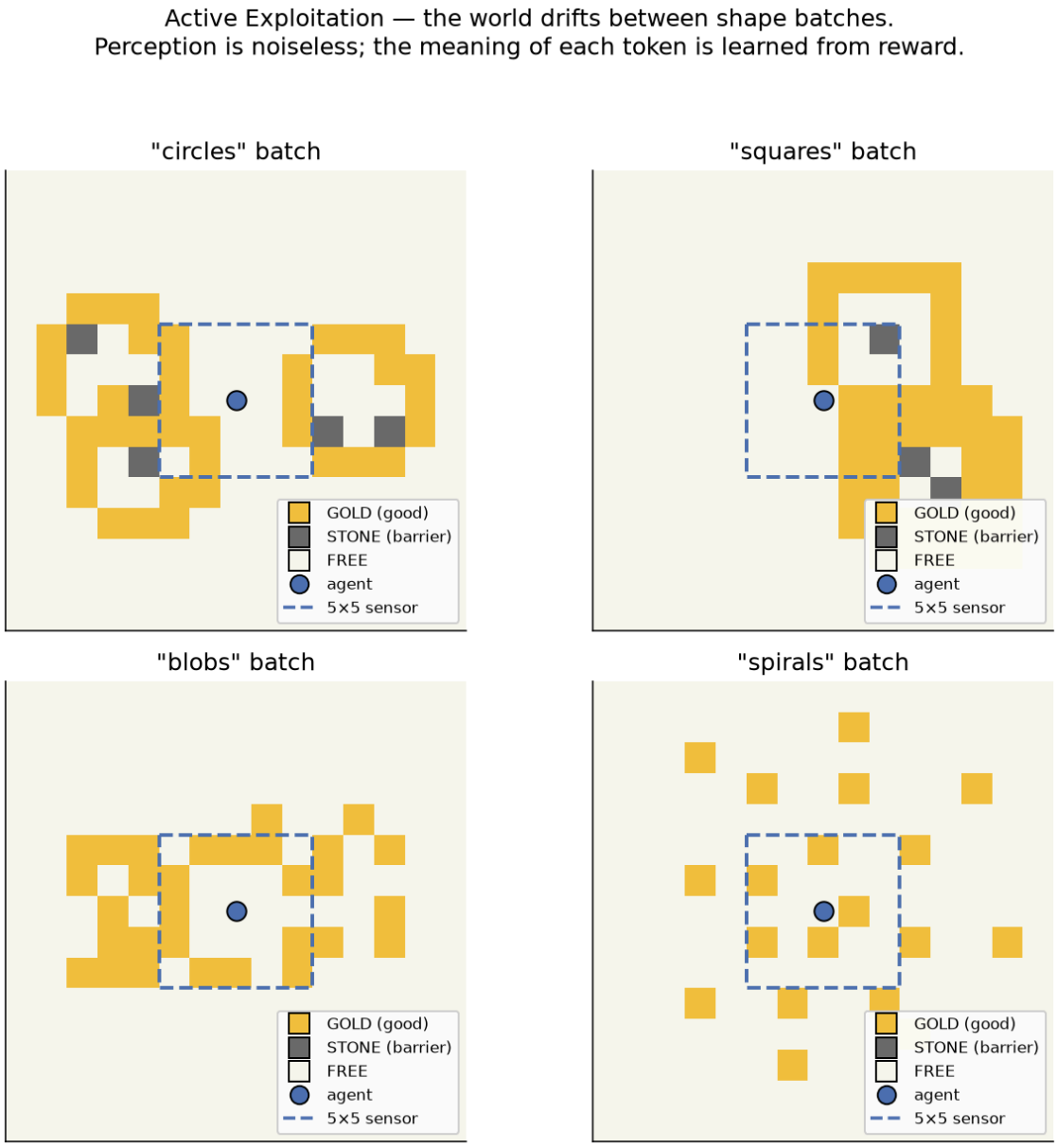
doi: 10.5281/zenodo.21501699

Assets & Materials

Reflex Bandit — Research Notebook. The tree-ensemble contextual bandit with two-timescale memory and EWMA drift detection, trained end-to-end.	https://blog.vski.ai/notebooks/notebooks/index.html?path=bandits_active_exploitation.ipynb
Online Demo: Active Exploitation Reflex Agent	https://blog.vski.ai/reflex/
Online Demo (Source Code)	https://vski.sh/x/reflex
MoB Queen: An Agent for Active Exploration (Opposite Regime)	https://blog.vski.ai/posts/mob-queen/

There are two ways to build an agent that has to *act* in a world it does not fully understand. The **active explorer**[*] has noisy sensors and a largely unknown world; its job is to *find out where* reward is. The **active exploiter** has noiseless sensors but does not know *what they mean* — `GOLD` does not read as "good" until reward teaches it, `STONE` does not read as "barrier" until it bumps one and is penalised. The tokens are fixed; the world's *shape* drifts — circles today, squares tomorrow, shapeless blobs the day after. The agent's job is to learn the reflexes each shape demands, exploit them while they hold, and **re-explore when they stop holding**.

We call this regime **active exploitation**. The explorer asks *where*; the exploiter asks *what*. They need different machinery.



The regime

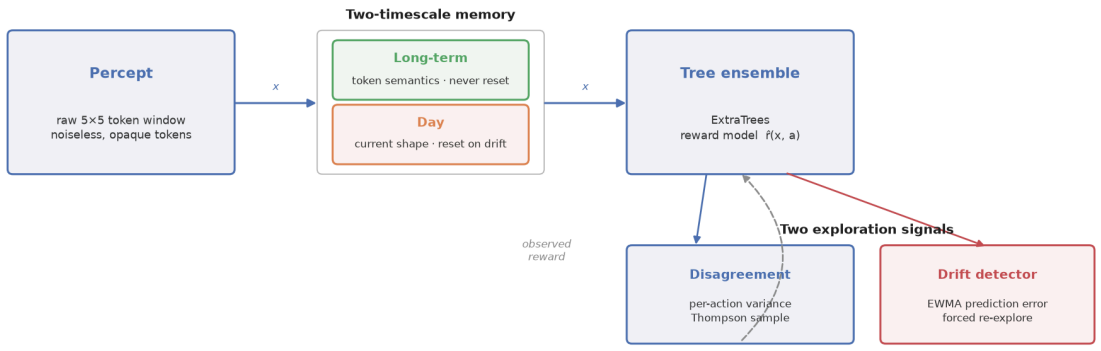
Active exploitation is the right framing whenever three conditions hold simultaneously.

Condition	What it means	Why it matters
Noiseless perception	The agent sees true state, not a noisy surrogate.	Eliminates the belief map; the agent's uncertainty is about *meaning*, not *where*.
Unknown semantics	The mapping from observation to value is not given; it must be learned from reward.	A rules engine cannot do this — that would be hand-coding, not learning.
Drifting structure	The world's shape changes over time, in batches or continuously.	A static supervised model overfits; a single policy that never adapts goes stale.

When all three hold, the problem stops being a Markov decision process and becomes a **contextual bandit**. Reward is observed *immediately* after each action — one step, one reflex — so there is no delayed credit assignment, no discount factor, no bootstrapping. The deadly triad that constrains every temporal-difference learner (bootstrapping + off-policy + function approximation) simply does not apply. This is why a value-function planner such as Fitted Q-Iteration — with its bootstrap target $Q \leftarrow r + \gamma \cdot \max_a Q(s', a')$ — is the *wrong tool* here. There is nothing to bootstrap through. The contextual bandit drops the $\gamma \cdot \max_a Q(s', a')$ term entirely and fits the reward function directly: $\hat{r}(x, a)$.

The architecture

The agent has three components, each addressing one thing the regime demands.



The reward model — a tree-ensemble contextual bandit

The reward model is $\hat{r}(x, a) =$ an extremely randomised trees ensemble. The context x is the agent's raw percept — a one-hot window of cell tokens around each candidate action's destination. Crucially, **the tokens are opaque to the agent**: `FREE`, `GOLD`, `STONE` are uninterpreted labels, and the ensemble has to discover, from reward alone, that the `GOLD` token is the one associated with positive return. A tree's axis-aligned splits are what make this

discovery possible — the first split on the `GOLD`-indicator feature is literally the agent learning *gold is the good token*.

The encoding is **action-invariant**: every action's destination window lives in the *same* feature columns, plus a small one-hot action ID. The destination token (cell 12 of the window) occupies one fixed feature index regardless of which action is being scored, so the tree learns a single rule — *GOLD at the destination → reward* — that is shared across all eight action slots. The natural alternative (action-specific blocks, where action 0's gold indicator is a different feature from action 3's) fragments the gold signal into eight disjoint undersampled problems; the agent plateaus at ~75% gold-grab because a gold cell at slot 3 cannot benefit from gold observed at slots 0, 1, 2. Sharing the feature across slots pools the data and the rule is learned within one world.

Why trees, specifically, in the bandit setting? Three reasons.

- **Interpretability of the learned meaning.** The first split of the first tree reads off as the meaning the agent has learned. A neural bandit would learn the same mapping but hide it inside an embedding that no human can audit.
- **Native uncertainty quantification.** Each tree in the ensemble is an independent estimate of $\hat{r}(x, a)$; their disagreement is a calibrated proxy for epistemic uncertainty. This is the basis of exploration (below). A neural net needs bootstrap ensembles or Bayesian last layers to get the same thing.
- **Cheap online re-fitting.** At the scale of a deployed reflex layer (thousands, not millions, of transitions per regime), an ensemble re-fit is a few hundred milliseconds on CPU — fast enough to keep up with a drifting world without GPU infrastructure.

This is the tree-ensemble contextual bandit of Nilsson et al. (TMLR 2024) — ensemble disagreement as the exploration signal, exactly as Pathak et al. (ICML 2019) used it for intrinsic motivation in RL.

Two-timescale memory

The single biggest design decision in this regime is **what to forget, and when**. The agent has to learn two things at once, on two different timescales:

- **Token semantics** — `GOLD` is good, `STONE` is a barrier. This is constant across every regime the agent will ever see. It must *never* be forgotten; relearning it after every drift is a catastrophic waste.
- **Shape pattern** — today the gold sits in rings; yesterday it sat in square perimeters. This is the *current regime*. It must be forgotten the moment the regime changes, or the agent keeps acting on yesterday's shape and mispredicting today's.

One memory timescale cannot do both. A single buffer either holds the old shape too long (and mispredicts the new one) or flushes too aggressively (and forgets that `GOLD` is good). The agent keeps two buffers and concatenates them at fit time: a **long-term buffer** capped at a thousand or so transitions, never reset, that carries semantics across every regime; and a

day buffer that resets at each regime boundary and carries only the current shape pattern. This is the discounted/sliding-window bandit idea, made two-timescale so the discount applies to shape, not to meaning.

Two complementary exploration signals

Two signals, two scales of uncertainty.

Ensemble disagreement — the per-action standard deviation across trees — answers *"is the model confident about which action is best here?"* When the trees disagree, the agent samples a Thompson posterior and explores. This handles the within-regime case: a context the agent has not seen often enough to have a confident ranking over actions.

Prediction-error drift detection answers *"is the world still the world the model was trained on?"* It tracks the EWMA of $|r_{\text{observed}} - r_{\text{predicted}}|$ and fires when that error crosses a statistical threshold (the ECDD detector of Ross et al. 2013, the continuous-signal descendant of Gama's DDM 2004). When it fires, the agent opens a forced-random re-explore window to collect fresh post-drift data before trusting its model again.

These two signals are not redundant — they see different things, and the difference is the single most important architectural point in the design. **Ensemble disagreement tracks the model's own confidence; prediction error tracks whether the world still matches the model.** When token semantics are stable (and they almost always are — `GOLD` does not stop being good), the ensemble stays confident even during a shape drift, because every tree still agrees that `GOLD = reward`. The shape mismatch shows up as the model *confidently mispredicting* rewards it used to get right. Only the prediction-error signal catches that. Disagreement alone is blind to it.

Validation

The agent runs in the **Gold Digger** grid — a recommender engine stripped to its load-bearing geometry: an agent on a grid, reward painted on some cells, obstacles on others, and (here) a noiseless sensor that reads the raw token at each cell. The grid is a state space, not a map: each cell is a candidate item, configuration, or probe target; gold is a click, a conversion, a successful probe; stones are no-gos. The agent perceives a raw 5×5 token window around each candidate destination. Perception is noiseless; semantics are opaque. Reward is sparse: gold $+1$, bumping a *discovered* barrier -0.5, exploration free. Worlds are drawn from four shape families — **circles, squares, blobs, spirals** — and presented either as discrete batches (one shape per day, switched at the boundary) or as a continuous blend (one shape morphing into another over thirty-two episodes).

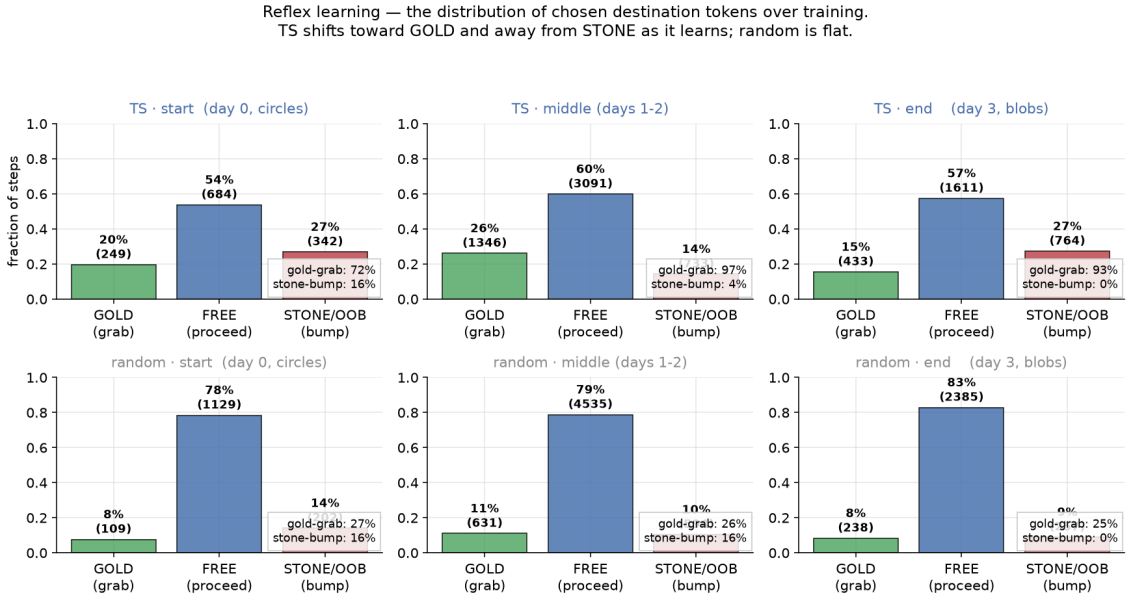
Three arms are compared on a matched-pair protocol (identical world and sensor seeds across arms): **TS** (Thompson Sampling over the tree ensemble, with two-timescale memory, cold-start floor, and drift detector), **greedy** (same ensemble, mean prediction, same memory), and **random** (no model).

Reflex learning

On the first batch of worlds the agent has ever seen, with no labelled semantics, TS collects **2.38× the gold of random**. By the second circle batch (after a squares batch in between), TS collects 24.5 gold/episode against random's 12.8 — the long-term memory has carried the token semantics across the intervening batch. The meaning was learned from reward alone.

Reflex confusion — grab gold, evade stones

Gold collected measures outcomes, but not whether the agent has actually learned the *two distinct reflexes* the regime demands: grab `GOLD`, evade `STONE`. The diagnostic classifies every step's chosen destination token into {GOLD, FREE, STONE/OOB} and pools steps across three training phases. A trained agent shifts toward GOLD and away from STONE; random is flat.



The carryover test is the load-bearing result: the same `circles` batch, seen first on day 0 and again on day 2 after a `squares` batch intervened.

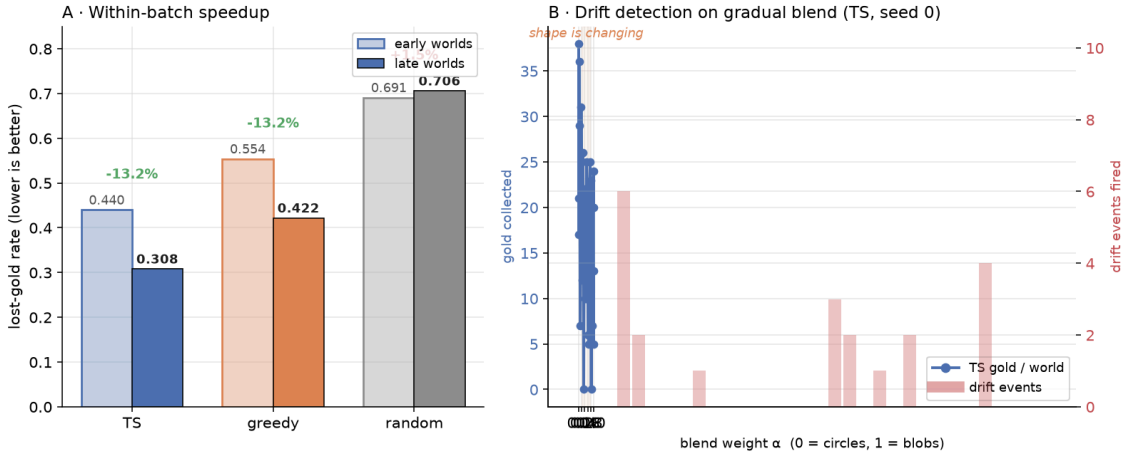
Circles batch	gold-grab	stone-bump
Day 0 (first time seen)	85%	15%
Day 2 (after squares intervene)	**97%**	**4%**

Token semantics survive the regime boundary in the long-term memory. By day 2 the reflex is essentially perfect — 97% gold-grab, 4% stone-bump — even though the agent has not seen circles since day 0 and spent the intervening batch learning a different shape.

Within-batch speedup

Within a batch, the lost-gold rate (fraction of available gold the agent fails to collect) drops as the agent sees more worlds from the same shape:

Arm	Early worlds	Late worlds	Delta
TS	0.418	**0.304**	**−11.4%**
greedy	0.424	0.295	−12.8%
random	0.691	0.706	+1.6% (flat, as it should be)



Random is flat — the improvement is not the world getting easier, it is the agent learning. The two-timescale memory is doing its job: the shape pattern accumulates in the day buffer and the agent's reflexes tighten on it.

Drift re-explore

The hardest test. The world blends continuously from circles to blobs over thirty-two episodes. The drift detector fires **5–10 drifts and 15–27 warnings per seed**, and they cluster exactly where they should — in the middle of the blend, where the shape is actually changing, not at the settled endpoints. Crucially, the detector keeps firing across all three seeds and never fires on the random arm (which has no predictions to be wrong about) — a clean signal that what it is detecting is genuine model-mismatch, not noise.

This is the result that the disagreement signal *could not have produced on its own*. During the blend, the ensemble stays confident because the token semantics are stable; only the prediction-error detector sees the shape drift.

Negative results

- **Disagreement alone misses shape drift.** When semantics are stable, disagreement is low throughout the drift — every tree agrees that `GOLD` is good even when the gold has moved. The drift detector is not an optimisation; it is *necessary*.

- **EWMA lags abrupt switches.** The detector is built for gradual drift. On the discrete day-boundary schedule it fires late — after a few forced-random actions that cost some gold. The agent's performance on later discrete batches reflects this cost. On the gradual schedule the detector was designed for, the cost is negligible.
- **Saturation hides the detector's performance value.** A world with abundant gold and a generous step cap lets random collect most of it, so the three arms tie in aggregate. The detector's *signal* value (firing correctly when the shape changes) is the load-bearing result; the *gold* value is masked by the environment's permissiveness. Per-band-of-blend breakdowns, where TS leads by 20–30% in the turbulent middle, are the honest signal.

Where this pattern actually lives

The regime maps onto a long list of deployed problems that are routinely — and wrongly — treated as sequential RL.

Anomaly and intrusion detection

The agent sees telemetry perfectly — a request, a packet, a login. It does not know which patterns are benign and which are hostile until an incident label arrives. The attack shape drifts: today's exfiltration looks different from yesterday's cryptomining. The reflex to learn is *see this signature → escalate*.

Adaptive manufacturing controllers

Sensors read process state perfectly, but the *meaning* of a vibration signature or a temperature drift — acceptable, tool-wearing, catastrophic — is learned from outcome data, and the meaning shifts as tooling wears and batches change.

Robotics reflex layers

A robot's cameras and proximity sensors are noiseless at the relevant timescale, but the robot must *learn* that a particular visual configuration is a step (climb), a wall (evade), or a target (grab) — and the test site changes weekly.

Trading and market-making

Order-book state is observed exactly, but the *regime* — trending, mean-reverting, shocked — drifts continuously. The reflex is *see this book shape → place this order*, learned from realised PnL and relearned as the regime rotates.

Personalisation under concept drift

Content or product selectors whose users' tastes shift seasonally or with life events. The recommendation is a reflex against a context; the meaning of each context feature is learned and periodically relearned.

When it fails

Do not use it when perception is genuinely noisy (you need a belief map and the active-explorer machinery: Bayesian updates, a value function, Fitted Q-Iteration), when reward is delayed (you are back in MDP territory and need a discount factor), or when the world is stationary (a plain supervised model is simpler, faster, and more accurate). It is also the wrong choice if the percept-token set is so large that tree splits cannot enumerate the relevant interactions — at that scale, a neural bandit with a Bayesian last layer becomes the better trade.

The cleanest tell that you are in this regime: **the model's confidence and the model's accuracy come apart.** The agent stays confident in its reflexes right up until the moment they stop working. That gap — confident-but-wrong — is exactly what the prediction-error drift detector exists to catch, and exactly what the explorer architecture, with its single disagreement signal over a noisy belief, cannot.

Conclusion

The active explorer and the active exploiter are duals. The explorer's job is to find out *where* reward is, under noisy perception, and its core machinery is a belief map updated one observation at a time. The exploiter's job is to find out *what* the patterns in its noiseless perception mean, and to relearn them when they drift; its core machinery is a tree-ensemble contextual bandit with two-timescale memory and a prediction-error drift detector.

The single most important architectural insight is that exploration in this regime needs **two signals, not one**. Ensemble disagreement handles the within-regime case — the model is uncertain about a context it has not seen enough. Prediction-error drift detection handles the across-regime case — the model is confident but the world has moved. Neither signal subsumes the other. Disagreement is blind to shape drift when semantics are stable; prediction error is blind to per-context uncertainty once the global error average settles. Together they cover the space, and neither alone is enough.

The second insight is that **memory must be two-timescale**. A single buffer cannot both remember what the tokens mean and forget what the shape looked like. Forgetting semantics on every drift is a catastrophe; never forgetting the shape is a slower one. Split the memory: long-term for meaning, short-term for shape.

Together, these two design moves take a regime that looks like RL — an agent acting in a world — and reframe it as what it actually is: a non-stationary contextual bandit with immediate reward. The right machinery for that is not a value function and a discount factor. It is a reward model, two memory timescales, and two exploration signals.

References

1. **Nilsson, H. et al.** *Tree Ensembles for Contextual Bandits.* (TMLR 2024) — tree ensembles as bandit reward models, with ensemble disagreement as the exploration signal. The canonical reference for the planner's core structure.
2. **Pathak, D., Gandhi, D. & Gupta, D.** *Self-Supervised Exploration via Disagreement.* (ICML 2019) — ensemble disagreement as an intrinsic exploration motive; the principle the disagreement signal is built on.
3. **Ross, G. J., Adams, N. M., Tasoulis, D. K. & Hand, D. J.** *Exponentially Weighted Moving Average Charts for Detecting Concept Drift.* (Pattern Recognition Letters, 2013) — ECDD, the EWMA drift detector the prediction-error signal uses.
4. **Gama, J., Medas, P., Castillo, G. & Rodrigues, P.** *Learning with Drift Detection.* (SBIA 2004) — the original DDM, of which ECDD is the continuous-signal variant.
5. **Cavenaghi, M. A. et al.** *Non-Stationary Multi-Armed Bandit: Empirical Evaluation of a Novel Approach.* (Entropy 2021) — f-DSW-TS, the discounted/sliding-window Thompson Sampling whose two-timescale memory split the agent generalises.
6. **Chen, Y., Lee, C.-W., Luo, H. & Wei, C.-Y.** *A New Algorithm for Non-stationary Contextual Bandits.* (COLT 2019) — the dynamic-regret theory for non-stationary contextual bandits, the formal setting this agent instantiates.
7. **Auer, P.** *Using Confidence Bounds for Exploitation-Exploration Trade-offs.* (JMLR 2002) — the bandit framing of exploration under uncertainty that both signals inherit.
8. **Sutton, R. S. & Barto, A. G.** *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, 2018 — Ch. 2 (bandits) and §2.9 (associative search = contextual bandits) for the foundational contrast between the bandit setting and the full MDP this regime deliberately steps out of. Full PDF.