PP-GYM: UNIFYING ENVIRONMENTS, LEARNING, AND EVALUATION FOR PERFORMATIVE PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Performative prediction (PP) studies learning when deployed decisions reshape subsequent data, as prices change demand. Existing implementations often tie algorithms to particular response models and feedback assumptions, making reuse and comparison across settings depend on custom integration. We introduce PP-Gym, a unified framework that separates environments, observations, learning algorithms and evaluation through a common deployment–feedback interface. It supports reusable algorithm implementations and comparisons with controlled information access, population responses and deployment counts. Across 13 tasks spanning immediate and gradual responses and individual-record or aggregate feedback, we evaluate 26 methods and variants. Building on this shared interface, we introduce the Trust-Region Cross-Loss Surrogate (TR-CLS), a simple learner that models deployment-dependent loss and selects new decisions from individual records or aggregate feedback. Under the shared protocol, TR-CLS significantly lowers average loss during learning on nine tasks relative to baselines selected on development data, including a 14.3% increase in revenue in simulated ten-product pricing. These results demonstrate the value of studying shared learning strategies across PP settings; PP-Gym provides the experimental infrastructure to develop them and test where their gains persist.

1 INTRODUCTION

A retailer adjusts prices using sales records, but each new price also changes demand and the data informing its next update. **Performative prediction (PP)** studies learning under such decision-induced distribution shifts (Perdomo et al., 2020; Hardt & Mendler-Dünner, 2023). A **stateless response** depends on the current decision, whereas a **stateful response** retains effects of past decisions (Brown et al., 2022; Izzo et al., 2022). Available **feedback** may also differ: a learner may observe individual transactions or only aggregate revenue (Chen et al., 2024; Hikima et al., 2025).

Existing PP methods address different response and feedback assumptions (Perdomo et al., 2020; Mendler-Dünner et al., 2020; Izzo et al., 2021; Brown et al., 2022; Izzo et al., 2022; Chen et al., 2024; Hikima et al., 2025), but their implementations often bind the learning rule to a particular simulator and evaluation procedure. Testing a method under new conditions requires custom integration. This makes it difficult to determine which algorithmic gains persist beyond the original setting. Researchers need to vary population responses and available feedback without rebuilding the experiment, while comparing methods using the same information and number of deployment trials. A shared experimental foundation would support both algorithm reuse and systematic investigation of these differences.

We introduce **PP-Gym** (Figure 1), a unified framework for independently reusable environments and compatible learning algorithms, following Gymnasium's approach to interoperability (Towers et al., 2025). Its 13 tasks vary population memory and feedback visibility under a common deployment–feedback protocol. This structure supports both comparisons of existing methods and development of strategies shared across settings. We illustrate the latter with **TR-CLS**, a simple learner that models deployment-dependent loss from individual records or aggregate feedback. It significantly reduces average loss during learning on nine tasks against development-selected baselines. These results show that a common loss-modeling strategy can work across distinct PP settings.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

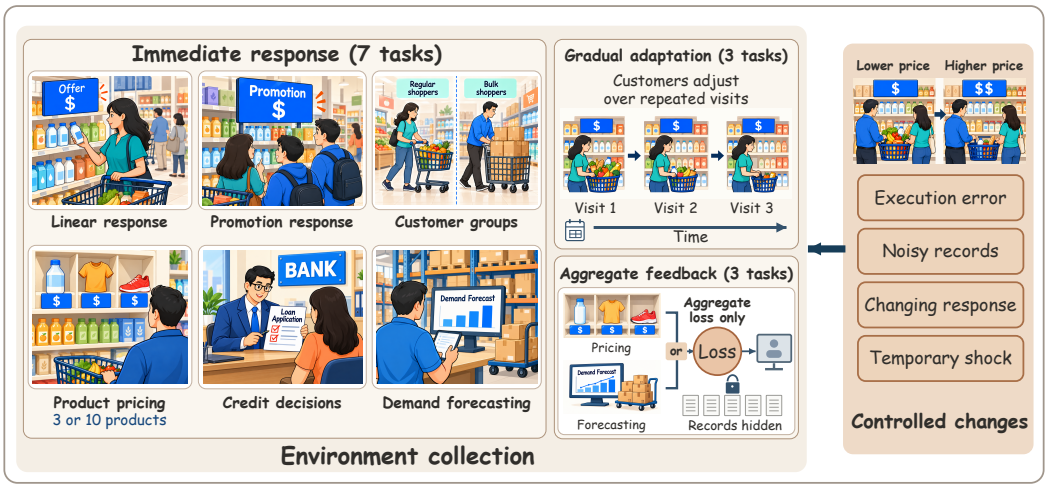


Figure 1. Environments and controlled changes in PP-Gym. Application scenes illustrate synthetic deployment-induced responses: seven tasks respond immediately, three pricing tasks adapt gradually, and three reveal aggregate loss while hiding records. Product counts specify jointly chosen prices; the retail vignette shows purchasing changes after a price update. Controls vary execution, observation, response relationships and temporary shocks.

Our contributions are threefold:

1. **Interchangeable PP components.** A common interface separates response simulation, observations, learning and evaluation, allowing compatible components to be combined without rewriting the experiment loop.
2. **Controlled experiments across settings.** Thirteen tasks and 26 methods and variants yield 297 compatible combinations for studying the effects of population memory, feedback and environmental changes.
3. **Reusable learning and empirical findings.** TR-CLS provides a shared loss-surrogate strategy across these conditions. Evaluation during learning and after updates stop reveals how the evaluation stage changes method rankings.

2 A Unified Interaction Framework

2.1 From Pricing to Composable Experiments

PP-Gym separates four components. The **simulation environment** generates demand and retains persistent price effects. The **observation interface** returns transaction records or aggregate loss. The selected **learning algorithm** chooses the next price; options include TR-CLS, repeated risk minimization (RRM) and performative gradient descent (PerfGD). The **evaluation protocol** sets initial conditions, sample counts, deployment budgets and scoring. Classifier or regression parameters can replace prices without changing this loop.

Algorithms affect population state through submitted rules and access only permitted observations; the evaluator separately records expected performance. Reusing batches is internal computation, whereas every request for new data counts as a deployment. Researchers can compare algorithms under fixed population responses and feedback, or hold the algorithm fixed and vary either condition.

2.2 How Deployment Produces Data and Observations

At interaction t, the selected learner submits θ_t using its own observation history. The environment executes $\widetilde{\theta}_t$, updates population state from s_t to s_{t+1} and generates batch B_t. In pricing, purchases determine revenue and inform the learner's next price (Figure 2).

Execution errors change actual prices; response changes alter demand at the same price; observation errors corrupt sales records. PP-Gym controls these mechanisms separately to identify the source of performance changes.

Let $D_t(s, \theta)$ denote the data distribution given population state s and rule θ. A record z is a demand vector or feature–label pair. The loss $\ell(\theta, z)$ evaluates a rule on a record; L_t is its batch average at the executed rule. The observation map returns o_t, including observed loss L_t^{obs}, which equals L_t without observation noise:

$$\begin{aligned} \widetilde{\theta}_t &= A_t(\theta_t), & s_{t+1} &= F_t(s_t, \widetilde{\theta}_t, \xi_t), \\ B_t &\sim D_t(s_{t+1}, \widetilde{\theta}_t)^{n_t}, & o_t &= O_t(B_t, \widetilde{\theta}_t). \end{aligned} \tag{1}$$

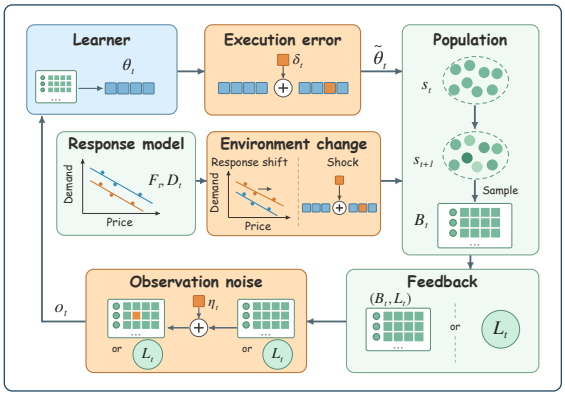


Figure 2. The shared deployment–feedback loop. The learner uses available observations to choose the next rule. Orange modules intervene in execution, response and observations; δ_t and η_t denote execution and observation errors. Squares depict vector components or sample fields. Feedback contains samples and loss or loss alone.

Here, A_t maps planned to executed rules and F_t models population adaptation with randomness ξ_t. Both A_t and O_t may be random. The distribution D_t generates n_t conditionally independent post-transition samples, and O_t reveals records or loss alone. Batch-level external shocks modify this sampling law and its risk interpretation (Appendix E.1). In a fixed stateless environment, responses depend only on the current rule, giving $D(\theta)$ and performative risk $R(\theta) = \mathbb{E}_{z\sim D(\theta)}[\ell(\theta, z)]$ (Perdomo et al., 2020).

Each interaction is an actual deployment. Trying higher and lower prices to infer a profitable direction is zeroth-order optimization: search through loss queries without directly receiving gradients. The two trials consume two deployments, with population dynamics continuing between them. An acquired batch can evaluate other prices without generating new demand (Jagadeesan et al., 2022), separating computation from deployment costs.

2.3 Public API and Visible Information

PP-Gym implements Gymnasium's `reset`, `step` and space declarations (Towers et al., 2025). `reset` initializes an experiment; `step` deploys one rule. Actions specify admissible prices or model parameters. Observations contain the executed rule, observed loss, relative time and a sample array. Visible batches populate the array; aggregate-only feedback hides individual records through an observation mask.

Retraining and sample-gradient methods use the batch; black-box methods can use loss alone. Some response-gradient methods also require the distribution score, the derivative of log probability density with respect to distribution parameters. PP-Gym marks and evaluates this extra information separately, admitting methods when tasks satisfy their requirements. Matching seeds fixes initial random streams, but different deployed rules still generate different subsequent data.

2.4 Evaluating Learning and the Final Rule Separately

Pricing must balance revenue during exploration with the quality of the final price. **Online performance** aggregates learning losses, including exploration costs; **continued-deployment performance** evaluates the final rule after updates stop. Evaluator **population risk** is expected loss under the current data distribution, whereas observed loss comes from a finite, potentially corrupted batch.

For T deployments with batches B_t of size n_t, define post-response expected loss as $r_t = \mathbb{E}_{z\sim D_t(s_{t+1},\widetilde{\theta}_t)}[\ell(\widetilde{\theta}_t, z)]$. With uncorrupted observations, average online observed loss and popu-

lation risk are

$$\widehat{L}_T = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n_t}\sum_{z\in B_t}\ell(\widetilde{\theta}_t, z), \qquad J_T = \frac{1}{T}\sum_{t=1}^{T} r_t. \tag{2}$$

Frozen evaluation on the unperturbed tasks retains population state and holds the final executed rule $\theta_f = \widetilde{\theta}_T$ fixed for another H interactions. For $k = 1, \ldots, H$, state $\bar{s}_k$ evolves from $\bar{s}_0 = s_{T+1}$ as $\bar{s}_k = F_{T+k}(\bar{s}_{k-1}, \theta_f, \bar{\xi}_k)$, with state-update randomness $\bar{\xi}_k$. Average frozen risk is

$$J_{T,H}^{\text{freeze}} = \frac{1}{H}\sum_{k=1}^{H}\mathbb{E}_{z\sim D_{T+k}(\bar{s}_k,\theta_f)}[\ell(\theta_f, z)], \qquad \bar{s}_0 = s_{T+1}. \tag{3}$$

Freezing fixes the rule while the population keeps adapting, so finite-horizon frozen risk need not equal stationary risk after full adaptation (Ray et al., 2022). Under observation noise, algorithms update from corrupted information while evaluators use the data-generating distribution, separating observation error from actual performance.

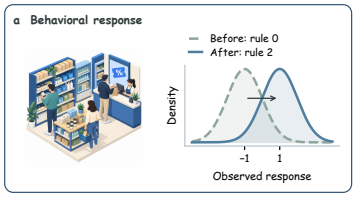
3 Environment Library and Composable Components

3.1 Simulation Tasks: How Rules Change Data

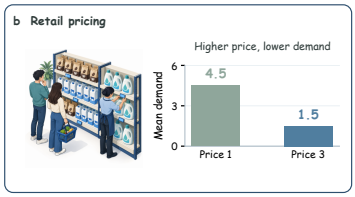
PP-Gym provides 13 stochastic simulation tasks: seven stateless tasks with sample feedback, three stateful pricing tasks and three with aggregate loss alone. Deployed parameters control responses; sampled batches provide data, and losses define learning objectives. The paired illustrations show application scenes and responses computed from the task models.

The **dimension** d counts deployed parameters: one for location response or regression, two for classification, and three or ten prices. Task suffixes 1D, 2D, 3D and 10D encode these counts. Six groups distinguish response mechanisms, memory and feedback.

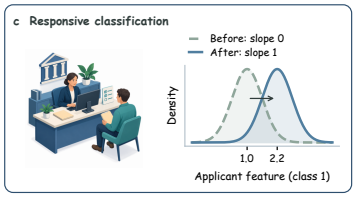
Location response (three tasks). Linear, nonlinear and mixture responses depend on scalar rule θ, with sampled response x and loss θx. The first two are Gaussian with means $\theta - 1$ and $\sqrt{1+\theta}$. The mixture combines two equally weighted Gaussian components whose means shift at different rates. In the linear example, changing the rule from 0 to 2 shifts the mean from -1 to 1.



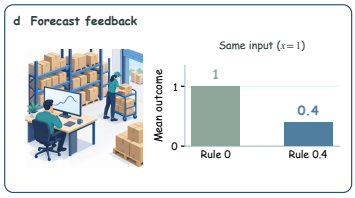
Immediate pricing (two tasks). Prices θ induce Gaussian demand x with mean $b - \varepsilon\theta$ and loss $-\theta^\top x$, where b is baseline demand and ε is price sensitivity. Demand is continuous and untruncated. For one product with $b = 6$ and $\varepsilon = 1.5$, raising price from 1 to 3 reduces mean demand from 4.5 to 1.5.



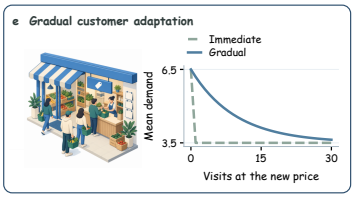
Responsive classification (one task). Deployment sets a linear classifier's intercept and slope for two equally likely classes, evaluated by regularized logistic loss. Class 0's feature distribution stays fixed; the class-1 mean shifts with the deployed slope. Increasing the slope from 0 to 1 moves that mean from 1 to 2.2.



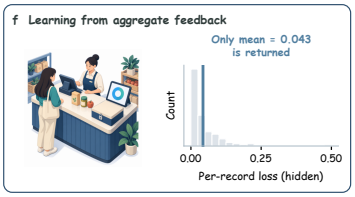
Responsive regression (one task). Standard Gaussian features generate labels $y = (1 - 1.5\theta)x + 0.3\epsilon$, with independent standard Gaussian noise ϵ. The rule predicts θx under squared loss and also changes the target. At $x = 1$, changing θ from 0 to 0.4 lowers the mean target from 1 to 0.4.



Gradual pricing (three tasks). Without execution error, demand is sampled around mean $s_{t+1} = (1-\alpha)s_t + \alpha(b - \varepsilon\theta_t)$. Fast and slow three-product tasks and ten-product pricing use $\alpha = 0.4$, 0.1 and 0.2, respectively. Smaller rates preserve historical price effects longer. The illustration compares immediate response with $\alpha = 0.1$, at baseline demand 6.5 and new price 2.



Aggregate loss only (three tasks). Three- and ten-product pricing and single-parameter regression reveal batch-average loss while hiding records. Learners explore through deployments because unavailable samples cannot be reused to evaluate other rules. The illustration aggregates 128 regression records into one observed loss.



3.2 Composing and Extending Experiments

PP-Gym combines response simulation, visible feedback and a compatible learner in one loop. Switching learners or changing adaptation, feedback and perturbations preserves the interaction and evaluation semantics in Section 2.

Independent experimental controls. Following Robust Gymnasium (Gu et al., 2025), modular interventions perturb executed rules and observations, change population responses, or introduce temporary demand and location shocks. Response strength, adaptation rate, sample size and dimension are independently configurable. Evaluation accounts for executed actions and response changes; Appendix E.1 defines shock-adjusted evaluation.

Interchangeable learning algorithms. The 26 methods and variants span retraining and sample gradients (Perdomo et al., 2020; Mendler-Dünner et al., 2020), response estimators such as PerfGD (Izzo et al., 2021), black-box search (Spall, 1992; Auer et al., 2002a; Srinivas et al., 2009; Hansen, 2016; Flaxman et al., 2004; Auer et al., 2002b) and loss surrogates. All read observations and submit rules. Scalar-loss methods also run when records are visible; sample-based methods require records, and some response estimators require additional structure or distribution information. Compatibility checks yield 297 executable method–task pairs.

A reusable experiment loop. The same `reset–act–step–observe` loop runs compatible task–learner pairs (Appendix G.3). For example, simultaneous perturbation stochastic approximation (SPSA) (Spall, 1992) uses losses at randomly perturbed prices to update a three-product pricing rule. The example uses 120 deployments and 128 samples per deployment; reward is negative observed loss.

New tasks specify parameter bounds, response dynamics and observations; new learners implement action selection and feedback updates. The runner supplies seed splits, deployment records and aggregation. We validate implementations against numerical results from 14 paper and code targets. We also test 16 component combinations and the interfaces of all 13 shared tasks.

3.3 TR-CLS: Learning to Optimize Deployment-Dependent Loss

TR-CLS predicts deployment-dependent loss from observed history and selects the next rule to reduce performative risk. Both feedback modes use surrogate fitting and local search across response and information settings.

Comparing candidate rules on one batch. Normalize rules to $[-1,1]^d$; $\vartheta(u)$ maps u back to original coordinates (Appendix F.1). The submitted rule satisfies $\theta_t = \vartheta(u_t)$. Stateless cross-loss $L(u,v) = \mathbb{E}_{z\sim D(\vartheta(u))}[\ell(\vartheta(v), z)]$ separates data generation at u from evaluation at v, with $R(\vartheta(u)) = L(u,u)$. TR-CLS fits losses on the visible batch B_t^{obs} in a polynomial basis $\phi(v)$:

$$\widehat{L}_t(v) = \frac{1}{|B_t^{\text{obs}}|} \sum_{z \in B_t^{\text{obs}}} \ell(\vartheta(v), z) \approx c_t^\top \phi(v). \tag{4}$$

Without observation noise, $B_t^{\text{obs}} = B_t$. The coefficients c_t weight constant, linear and low-order products in $\phi(v)$. Batch reuse requires no new deployments; changes in c_t capture response history.

Estimating performative risk from history. After deployment t, TR-CLS fits $c_i \approx \rho_t c_{i-1} + W_t^\top \psi(u_i)$ using retained record pairs up to t, where $\psi(u) = (1, u^\top)^\top$. The fitted W_t models response and $0 \le \rho_t < 1$ models persistence. At fixed u, this recursion has equilibrium coefficients $W_t^\top \psi(u)/(1-\rho_t)$. Their predicted loss guides local search:

$$\widehat{R}_t(u) = \left[\frac{W_t^\top \psi(u)}{1-\rho_t}\right]^\top \phi(u), \quad u_{t+1}^{\text{cand}} \approx \underset{u\in[-1,1]^d,\ \|u-u_t\|_\infty \le \Delta_t}{\arg\min}\ \widehat{R}_t(u). \tag{5}$$

The trust-region radius Δ_t bounds local search. Search approximately solves equation (5), followed by exploratory perturbations. Predicted and observed improvements adjust the radius, following model-based trust-region optimization (Conn et al., 2009). This equilibrium prediction guides decisions; online and frozen risks evaluate their performance.

Aggregate feedback and environmental change. Without samples, TR-CLS predicts observed loss using polynomial features of the rule and the previous loss. Local search uses the model's equilibrium prediction (Appendix F.3). Memory estimation, trust-region search and change detection remain. Observations since the latest reset are weighted by the distance between their rules and the current rule. Sustained large prediction residuals trigger a history reset and reopen the search region. Both paths use permitted feedback to select the next deployment; PP-Gym supplies the response and evaluation protocol shared with competing learners.

4 Shared Benchmark and Cross-Method Findings

4.1 Cross-Method Performance under Shared Conditions

We compare 26 methods and variants in 297 compatible pairs across 13 tasks. Configurations are selected on four development seeds and tested on 20 seeds, each with 120 deployments and 128 samples per deployment. Every query counts as a deployment. Comparisons match visible information; additional-score methods are evaluated separately. We pair complete trajectories by seed and compute 95% Student t intervals for loss differences. Holm correction (Holm, 1979) applies within predefined comparison families.

Table 1. TR-CLS and baseline performance. Mean online observed loss (20 seeds; lower is better). Pricing loss is negative revenue. Tasks follow Section 3.1; (3)/(10) indicate product counts. Development loss selects public-feedback baselines outside the surrogate family. Gain is $100(L_{\text{base}} - L_{\text{TR}})/|L_{\text{base}}|$. Green/red *: significant gains/losses (Holm correction, 13 paired tests); gray ns: no separation. Paired intervals: Table D4.

Task	Baseline	Baseline loss	TR-CLS loss	Gain (%)
Linear response	Guided ZO	−0.2287	−0.2397	**+4.8***
Nonlinear response	Guided ZO	−0.3593	−0.3753	**+4.5***
Mixture response	RGD	−0.0177	−0.0191	+8.1^{ns}
Immediate pricing (3)	Guided ZO	−23.4934	−24.4073	**+3.9***
Immediate pricing (10)	GP-LCB	−72.2496	−82.6016	**+14.3***
Classification	Anderson RRM	0.1258	0.1275	**−1.3***
Responsive regression	RGD	0.0496	0.0534	**−7.7***
Fast pricing (3)	SPSA	−18.3840	−20.4938	**+11.5***
Slow pricing (3)	UCB	−21.9633	−21.7510	−1.0^{ns}
Gradual pricing (10)	UCB	−67.7067	−70.7753	**+4.5***
Aggregate pricing (3)	SPSA	−23.6255	−24.1599	**+2.3***
Aggregate regression	Two-point ZO	0.1626	0.1094	**+32.7***
Aggregate pricing (10)	GP-LCB	−72.2496	−81.0350	**+12.2***

Table 1 summarizes Figure 3's primary comparisons: nine significant improvements, two losses and two comparisons not separated. Gains reach 32.7% in aggregate regression. Selecting comparators from additional-score methods as well yields eight improvements, four losses and one comparison not separated.

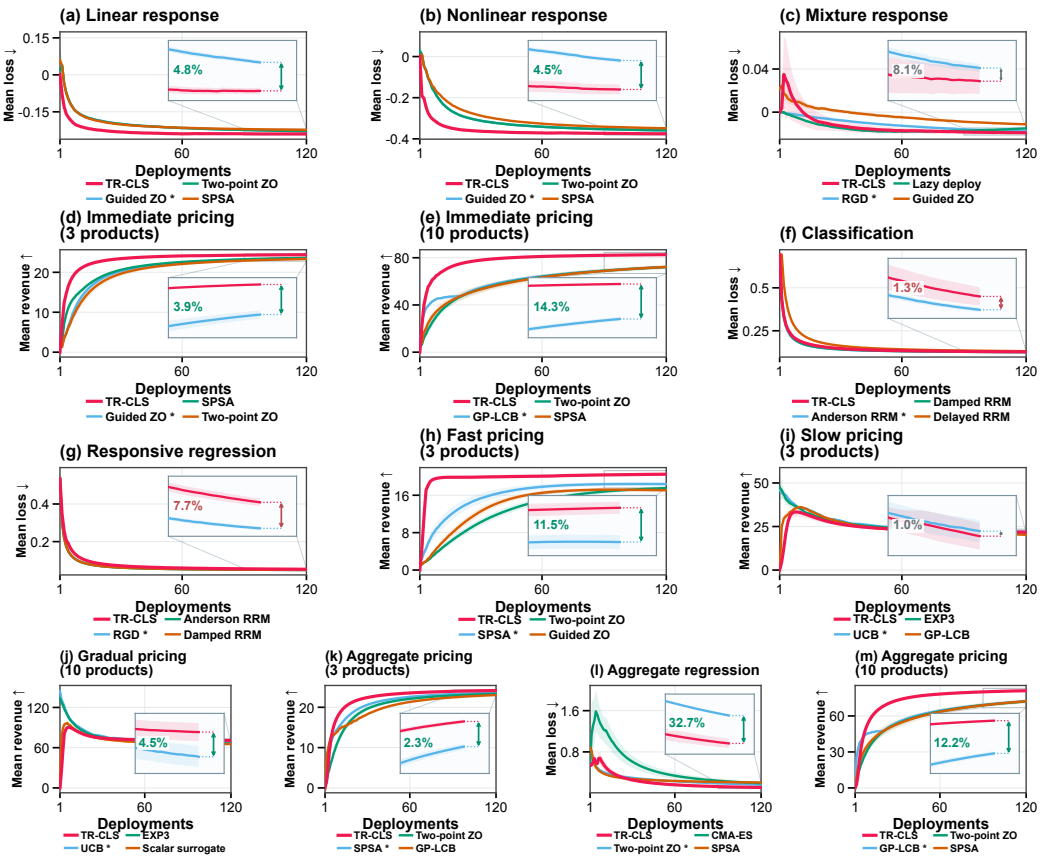


Figure 3. TR-CLS and the three public-feedback baselines with lowest development loss per task. Curves average observed loss over deployments 1–t; pricing shows mean revenue. Bands: pointwise 95% Student t intervals over 20 seeds. Insets enlarge deployments 90–120 with task-specific vertical ranges. Arrows compare TR-CLS with the primary baseline (*) using Table 1's gains and significance colors.

Across the full public-feedback leaderboard, TR-CLS shared the lowest test mean on three-product aggregate pricing and aggregate regression. TR-CLS or one of its three variants attained the lowest test mean on 11 tasks. Anderson RRM attained the remaining minima: 0.1258 in classification and 0.0493 in sample-feedback regression.

4.2 Isolating Environmental Effects and Learning Mechanisms

Figure 4 varies one environmental factor at a time. Algorithms interact with each modified environment, while evaluation records population risk to isolate its effects.

Response and memory change method rankings. Increasing response strength from 1.5 to 3.0 changed Damped RRM's mean risk from -7.5054 to 3.2718 and TR-CLS's from -24.4205 to -12.0784. In gradual pricing, UCB had lower mean risk than TR-CLS at adaptation rate 0.05 (-24.1060 versus -23.5293). At rate 0.8, TR-CLS had lower mean risk (-20.0191 versus -13.9963).

Observations, execution and sample size have distinct effects. Corrupting records changes algorithm inputs, whereas execution error changes customer-facing prices. TR-CLS reached risk -24.3946 at the largest observation-noise level and -11.6360 at the largest execution-error level. In sample-feedback regression, increasing batch size from 16 to 256 changed Damped RRM's risk from 0.0514 to 0.0496 and TR-CLS's from 0.0709 to 0.0536.

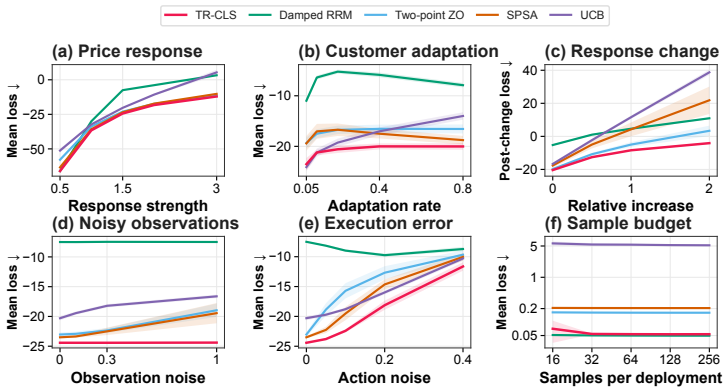


Figure 4. Controlled comparisons over 10 seeds (95% intervals), with clean-task configurations. Tasks: (a,d,e) three-product immediate pricing; (b,c) gradual pricing; (f) sample-feedback regression. Averages span 120 deployments, except (c): 60 post-change deployments. In (c), +1 doubles response strength. Both axes in (f) are logarithmic.

Mechanisms that transfer across conditions. In fixed-configuration ablations, removing memory raised risk by 12.2204 in slow-response three-product pricing and 38.1881 in ten-product pricing. Disabling resets after response changes raised risk by 10.3176 and 10.3487 in immediate and gradual pricing; retaining only recent history raised aggregate-regression risk by 0.0531. These differences were significant after Holm correction across 19 paired comparisons with 20 seeds. Clean-task reset and fixed-radius ablations showed no significant differences. Independent controls thus separate the benefits of memory, change detection and data retention.

4.3 Learning Costs and Final-Rule Performance

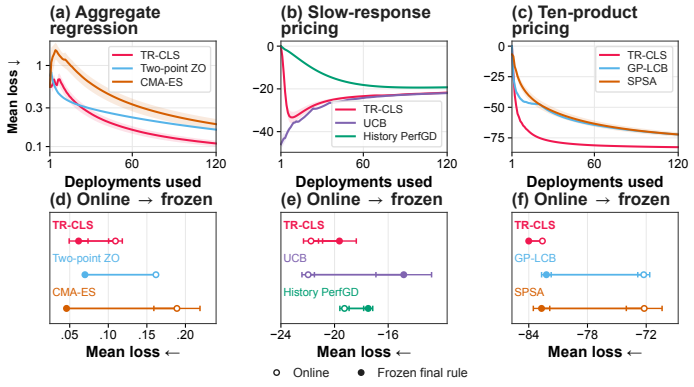


Figure 5. Learning and continued deployment across three feedback–response settings. Top: prefix-averaged expected loss on aggregate regression, slow-response pricing and ten-product immediate pricing. Bottom: online versus 40-deployment frozen loss for the same methods. Means and 95% intervals use 20 seeds. Regression uses a logarithmic learning axis. Frozen rules stop updating while population dynamics continue.

Figure 5 separates learning costs from final-rule quality. In scalar-feedback regression, online population risks were 0.1093 for TR-CLS and 0.1890 for CMA-ES. Frozen risks were 0.0615 and 0.0458, reversing their ranking. CMA-ES's final rule approached the risk floor of 0.045; its online score included earlier search costs.

Population memory further changes frozen performance. In slow-response pricing, UCB and TR-CLS had online risks -21.9673 and -21.7556, versus frozen risks -14.8384 and -19.6350. In ten-product immediate pricing, TR-CLS and GP-LCB had online risks -82.6050 and -72.2516, and frozen risks -84.0031 and -82.1997. The mean advantage persisted but narrowed.

Online and frozen mean-risk leaders differed on 12 of 13 tasks. This comparison excludes Scalar surrogate, the cross-loss family and additional-score methods. Spearman correlations ranged from 0.728 to 0.939. Leaders switched from Two-point ZO to CMA-ES in scalar-feedback regression and from UCB to History PerfGD in slow-response pricing. Rankings depend on evaluation stage, with individual gaps subject to seed uncertainty.

5 Related Work

Performative prediction. PP studies learning when deployed decisions change the data distribution (Perdomo et al., 2020). Retraining and stochastic optimization target stable rules that are optimal on their induced distributions (Mendler-Dünner et al., 2020; Drusvyatskiy & Xiao, 2020). Performative risk minimization instead accounts for the distributional response when choosing a rule (Miller et al., 2021). PerfGD estimates distribution sensitivity (Izzo et al., 2021), while plug-in and robust approaches address distribution modeling and misspecification (Lin & Zrnic, 2023; Xue & Sun, 2024). Stateful formulations capture historical effects and gradual adaptation (Brown et al., 2022; Izzo et al., 2022; Li & Wai, 2021). Bandit and zeroth-order methods address restricted feedback, including Markovian responses (Chen et al., 2024; Hikima et al., 2025; Liu et al., 2023). Further work studies neural networks and nonconvex losses (Mofakhami et al., 2023; Li & Wai, 2024), network and multiplayer interactions (Wang et al., 2023; Narang et al., 2022; 2023; Góis et al., 2025), recommendation (Eilat & Rosenfeld, 2023) and binary outcomes (Tsoy et al., 2025).

Simulation frameworks. WhyNot provides simulators for causal inference and dynamic decision-making, including classifier-induced feature responses (Miller et al., 2020). performativeGYM pairs extensible JAX distribution maps with optimizers for Gaussian responses, pricing and strategic classification (Sanguino Bautiste et al., 2026; 2025). Its optimizers update from batches and may query the distribution map internally. PP-Gym makes every new-data query a budgeted deployment and separates persistent population state, feedback access and evaluation. This enables comparisons across sample and aggregate feedback, immediate and gradual adaptation, and online and frozen evaluation under one protocol.

Standardized learning environments. Gym and Gymnasium standardize reinforcement-learning environments (Brockman et al., 2016; Towers et al., 2025). DeepMind Control Suite covers continuous control (Tassa et al., 2018), and PettingZoo provides a common interface for multi-agent learning (Terry et al., 2020). Robust Gymnasium modularizes perturbations (Gu et al., 2025), while GEM unifies environments for interactive language-model learning (Liu et al., 2026). HydroGym connects fluid solvers through a Gymnasium-compatible interface and provides validated flow-control benchmarks (Lagemann et al., 2026). Its shared environments support algorithm comparison and policy transfer across flow conditions.

6 Conclusion

PP-Gym connects separate PP experiments through reusable environments, interchangeable learners and explicit deployment and information semantics. Within this framework, TR-CLS learns deployment-dependent loss across feedback and memory conditions, with significant improvements in nine preselected public-feedback comparisons. Controlled studies reveal how population memory, environmental changes and evaluation stage alter algorithm performance. This combination of composable experiments and shared algorithm development provides a foundation for cumulative PP research.

Limitations and outlook. The benchmark covers one- to ten-dimensional continuous synthetic tasks. Separately validated neural, recommendation, network and multiplayer settings are not all included. Data-driven response models and new environments can broaden applications, constraints and data-generation mechanisms. Budgets and tuning grids are fixed; perturbations retain clean-task configurations. Longer horizons, perturbation-specific tuning and combined perturbations may change rankings. Frozen evaluation preserves population state for 40 deployments; stationary risk also depends on adaptation rates and transitions. TR-CLS's low-order loss representations and empirical memory models depend on response structure and dimension. Larger models increase batch-reuse and search costs. Beyond deployments and sample counts, higher-dimensional computation, response-estimation error and runtime need joint evaluation.

Reproducibility Statement

Appendix A specifies environment distributions, losses and dynamics. Appendices B and F describe algorithm mechanisms, hyperparameter grids and final configurations. Appendix C covers independent numerical reproduction and interface validation. Appendices D and E provide complete comparisons, statistical rules and perturbation settings. Appendix G provides self-contained component-integration examples, and Appendix H summarizes seeds, deployment budgets and computational settings.

Generative AI Use Statement

Large language models were used to improve the clarity and fluency of the manuscript, including sentence polishing, wording refinement, and grammar checking. They were not involved in developing the research ideas, designing the model architecture, constructing the dataset, conducting experiments, or analyzing the results. All manuscript content, experimental results, figures, and references were thoroughly reviewed and verified by the authors, who assume full responsibility for the submitted work.

References

Peter Auer, Nicolò Cesa-Bianchi, Paul Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. Machine Learning, 47(2):235–256, 2002a.

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, Robert E. Schapire. The Nonstochastic Multiarmed Bandit Problem. SIAM J. Comput., 32:48-77, 2002b.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John D. Schulman, Jie Tang, Wojciech Zaremba. OpenAI Gym. ArXiv, abs/1606.01540, 2016.

Gavin Brown, Shlomi Hod, Iden Kalemaj. Performative Prediction in a Stateful World. Proceedings of The 25th International Conference on Artificial Intelligence and Statistics, 151:6045–6061, 2022.

Richard H. Byrd, Pei-Huang Lu, Jorge Nocedal, Ci-You Zhu. A Limited Memory Algorithm for Bound Constrained Optimization. SIAM J. Sci. Comput., 16:1190-1208, 1995.

Yatong Chen, Wei Tang, Chien-Ju Ho, Yang Liu. Performative Prediction with Bandit Feedback: Learning through Reparameterization. Proceedings of the 41st International Conference on Machine Learning, 235:7298–7324, 2024.

Andrew R. Conn, Katya Scheinberg, Luís Nunes Vicente. Introduction to Derivative-Free Optimization. MPS-SIAM series on optimization, 2009.

Dmitriy Drusvyatskiy, Lin Xiao. Stochastic Optimization with Decision-Dependent Distributions. Math. Oper. Res., 48:954-998, 2020.

John C. Duchi, Michael I. Jordan, Martin J. Wainwright, Andre Wibisono. Optimal Rates for Zero-Order Convex Optimization: The Power of Two Function Evaluations. IEEE Transactions on Information Theory, 61:2788-2806, 2013.

Itay Eilat, Nir Rosenfeld. Performative Recommendation: Diversifying Content via Strategic Incentives. Proceedings of the 40th International Conference on Machine Learning, 202:9082–9103, 2023.

Abraham D. Flaxman, Adam Tauman Kalai, H. B. McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. ArXiv, cs.LG/0408007, 2004.

António Góis, Mehrnaz Mofakhami, Fernando P. Santos, Gauthier Gidel, Simon Lacoste-Julien. Performative Prediction on Games and Mechanism Design. Proceedings of The 28th International Conference on Artificial Intelligence and Statistics, 258:1855–1863, 2025.

Shangding Gu, Laixi Shi, Muning Wen, Ming Jin, Eric Mazumdar, Yuejie Chi, Adam Wierman, Costas Spanos. Robust Gymnasium: A Unified Modular Benchmark for Robust Reinforcement Learning. International Conference on Learning Representations, 2025:102014–102041, 2025.

Nikolaus Hansen. The CMA Evolution Strategy: A Tutorial. ArXiv, abs/1604.00772, 2016.

Moritz Hardt, Celestine Mendler-Dünner. Performative Prediction: Past and Future. ArXiv, abs/2310.16608, 2023.

Yuya Hikima, Hiroshi Sawada, Akinori Fujino. Guided Zeroth-Order Methods for Stochastic Nonconvex Problems with Decision-Dependent Distributions. Proceedings of the 42nd International Conference on Machine Learning, 267:23235–23260, 2025.

Arthur E. Hoerl, Robert W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. Technometrics, 42:80 - 86, 2000.

Sture Holm. A Simple Sequentially Rejective Multiple Test Procedure. Scandinavian Journal of Statistics, 6:65-70, 1979.

Zachary Izzo, Lexing Ying, James Zou. How to Learn when Data Reacts to Your Model: Performative Gradient Descent. Proceedings of the 38th International Conference on Machine Learning, 139:4641–4650, 2021.

Zachary Izzo, James Zou, Lexing Ying. How to Learn when Data Gradually Reacts to Your Model. Proceedings of The 25th International Conference on Artificial Intelligence and Statistics, 151:3998–4035, 2022.

Meena Jagadeesan, Tijana Zrnic, Celestine Mendler-Dünner. Regret Minimization with Performative Feedback. International Conference on Machine Learning, 2022.

Christian Lagemann, Sajeda Mokbel, Miro Gondrum, Mario Rüttgers, Yu-Ning Wang, Pol Suárez, Ludger Paehler, Deniz A. Bezgin, Aaron B. Buhendwa, Jared L. Callaham, Samuel Ahnert, Nicholas Zolman, Xiao Shao, Jean-Christophe Loiseau, Nikolaus A. Adams, Matthias Meinke, Wolfgang Schröder, Kai Lagemann, Esther Lagemann, Ricardo Vinuesa, Steve Brunton. The HydroGym reinforcement learning platform for fluid dynamics. Nature, 657:369 - 376, 2026.

Qiang Li, Hoi-To Wai. State Dependent Performative Prediction with Stochastic Approximation. International Conference on Artificial Intelligence and Statistics, 2021.

Qiang Li, Hoi-To Wai. Stochastic Optimization Schemes for Performative Prediction with Nonconvex Loss. Advances in Neural Information Processing Systems, 37:8673–8697, 2024.

Li-Cong Lin, Tijana Zrnic. Plug-in Performative Optimization. ArXiv, abs/2305.18728, 2023.

Hai-Tong Liu, Qiang Li, Hoi-To Wai. Two-timescale Derivative Free Optimization for Performative Prediction with Markovian Data. International Conference on Machine Learning, 2023.

Zichen Liu, Anya Sims, Keyu Duan, Changyu Chen, Simon Yu, Xiangxin Zhou, Haotian Xu, Shaopan Xiong, Bo Liu, Chenmien Tan, others. **GEM: A gym for generalist LLMs**. The Fourteenth International Conference on Learning Representations, 2026.

Celestine Mendler-Dünner, Juan Perdomo, Tijana Zrnic, Moritz Hardt. Stochastic Optimization for Performative Prediction. Advances in Neural Information Processing Systems, 33:4929–4939, 2020.

John Miller, Chloe Hsu, amroid, Moritz Hardt, Yonadav Shavit. zykls/whynot: v0.12.0b. Zenodo, 2020.

John Miller, Juan C. Perdomo, Tijana Zrnic. Outside the Echo Chamber: Optimizing the Performative Risk. International Conference on Machine Learning, 2021.

Mehrnaz Mofakhami, Ioannis Mitliagkas, Gauthier Gidel. Performative Prediction with Neural Networks. Proceedings of The 26th International Conference on Artificial Intelligence and Statistics, 206:11079–11093, 2023.

Adhyyan Narang, Evan Faulkner, Dmitriy Drusvyatskiy, Maryam Fazel, Lillian Ratliff. Learning in Stochastic Monotone Games with Decision-Dependent Data. Proceedings of The 25th International Conference on Artificial Intelligence and Statistics, 151:5891–5912, 2022.

Adhyyan Narang, Evan Faulkner, Dmitriy Drusvyatskiy, Maryam Fazel, Lillian J. Ratliff. Multiplayer Performative Prediction: Learning in Decision-Dependent Games. Journal of Machine Learning Research, 24(202):1–56, 2023.

Juan Perdomo, Tijana Zrnic, Celestine Mendler-Dünner, Moritz Hardt. Performative Prediction. Proceedings of the 37th International Conference on Machine Learning, 119:7599–7609, 2020.

Mitas Ray, Dmitriy Drusvyatskiy, Maryam Fazel, Lillian J. Ratliff. Decision-Dependent Risk Minimization in Geometrically Decaying Dynamic Environments. ArXiv, abs/2204.08281, 2022.

Javier Sanguino Bautiste, Thomas Kehrenberg, Jose A. Lozano, Novi Quadrianto. Strategically Deceptive Model Deployment in Performative Prediction. The 2026 ACM Conference on Fairness, Accountability, and Transparency, 2025.

Javier Sanguino Bautiste, Thomas Kehrenberg, Carlos Rosety, Jose A. Lozano, Novi Quadrianto. Performative Prediction made practical. ICLR Blogposts 2026, 2026.

James C Spall. **Multivariate stochastic approximation using a simultaneous perturbation gradient approximation**. IEEE transactions on automatic control, 37(3):332–341, 1992.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, Matthias Seeger. **Gaussian process optimization in the bandit setting: No regret and experimental design**. arXiv preprint arXiv:0912.3995, 2009.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, Martin A. Riedmiller. DeepMind Control Suite. ArXiv, abs/1801.00690, 2018.

Justin K. Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis Paulo Santos, Rodrigo Raúl León Pérez, Caroline Horsch, Clemens Dieffendahl, Niall L. Williams, Yashas Lokesh, Praveen Ravi. PettingZoo: Gym for Multi-Agent Reinforcement Learning. 2020.

Mark Towers, Ariel Kwiatkowski, John Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Kallinteris Andreas, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, J Terry, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, Omar G. Younis. Gymnasium: A Standard Interface for Reinforcement Learning Environments. Advances in Neural Information Processing Systems, 38, Main Conference, 2025.

Nikita Tsoy, Ivan Kirev, Negin Rahimiyazdi, Nikola Konstantinov. On the Impact of Performative Risk Minimization for Binary Random Variables. Proceedings of the 42nd International Conference on Machine Learning, 267:60336–60361, 2025.

Homer F. Walker, Peng Ni. Anderson Acceleration for Fixed-Point Iterations. SIAM J. Numer. Anal., 49:1715-1735, 2011.

Xiaolu Wang, Chung-Yiu Yau, Hoi To Wai. Network Effects in Performative Prediction Games. Proceedings of the 40th International Conference on Machine Learning, 202:36514–36540, 2023.

Songkai Xue, Yuekai Sun. Distributionally Robust Performative Prediction. ArXiv, abs/2412.04346, 2024.

APPENDIX A COMPLETE INTERACTION DEFINITIONS AND ENVIRONMENT CONFIGURATIONS

A.1 INTERACTION HISTORY AND STATE

A trajectory is determined jointly by the initial population state and the random streams of the environment and learner. Before deployment t, the public history, starting with initial observation o_0, is $h_t = (o_0, \theta_1, o_1, \ldots, \theta_{t-1}, o_{t-1})$. The learner uses h_t to propose θ_t, and the action wrapper produces the executed action $\widetilde{\theta}_t$. The environment then updates state, samples data and computes loss. Public observations record the executed action for use in algorithm updates. A learner that instead fits responses to planned actions faces an identification problem with execution error.

The initial observation contains a clipped zero action, zero loss, a zero-filled sample container and a no-sample mask. It establishes a valid observation structure and does not represent an actual deployment. The time observation is the number of completed deployments divided by the budget. Reaching the budget sets `truncated=True`; the tasks do not trigger `terminated` through an absorbing state. Further environment calls after budget exhaustion raise an error.

Proposition A.1 (Trajectory preservation under a common interface). Consider an interaction process with specified initialization, transition and sampling kernels, a loss function and an observation map. Suppose its Gym wrapper preserves these kernels, random-call order, transitions per action and public feedback. If the learner follows the same policy at corresponding histories, the two representations induce identical action–observation trajectory distributions and deployment counts.

Proof. The same initialization mechanism produces the initial observations, so initial history distributions coincide. Suppose corresponding history distributions agree before action t. The same learning policy gives the same conditional action distribution. Identical transition and sampling kernels then give identical conditional distributions for the next state, samples and observations, preserving the next history distribution. Induction establishes equality in distribution for every finite trajectory. Each action corresponds to exactly one transition, so deployment counts also agree. If corresponding transitions use the same deterministic functions of the coupled random draws, trajectories agree pathwise as well. This completes the proof.

This property specifies what the integration layer must preserve. Appendix C tests observations, randomness and accounting for the current tasks through interface checks.

A.2 STATELESS LOCATION RESPONSES

Gaussian-1D uses action domain $\Theta = [-2, 2]$, samples $x \sim \mathcal{N}(-1 + \theta, 0.5^2)$ after deployment and assigns loss $\ell(\theta, x) = \theta x$. Thus $R(\theta) = \theta^2 - \theta$, with optimum $\theta^* = 0.5$.

Nonlinear-1D uses $\Theta = [-0.95, 2]$ and $x \sim \mathcal{N}(\sqrt{1+\theta}, 0.5^2)$, retaining loss θx. The parameter domain keeps the square-root argument positive, and the objective is $R(\theta) = \theta\sqrt{1+\theta}$.

Mixture-1D uses two equally weighted components with means $-1 + \theta$ and $0.5 + 0.4\theta$, standard deviations 0.5 and 0.8, and action domain $[-2, 2]$. Its loss is θx, giving $R(\theta) = 0.7\theta^2 - 0.25\theta$. Component identities are hidden from the learner.

A.3 IMMEDIATE AND GRADUAL DEMAND

Immediate-demand tasks use price vectors $\theta \in [0, 5]^d$, with post-deployment demand and risk given by

$$x \sim \mathcal{N}(b - \varepsilon\theta, \sigma^2 I), \qquad \ell(\theta, x) = -\theta^\top x, \qquad R(\theta) = \varepsilon\|\theta\|_2^2 - b^\top\theta. \tag{A1}$$

The three-dimensional task uses $b = (6, 7, 8)^\top$; the ten-dimensional task uses evenly spaced components of b between 5 and 9. Here I is the identity matrix; defaults are $\varepsilon = 1.5$ and noise standard deviation $\sigma = 0.5$. Corresponding bandit tasks preserve these parameters while hiding samples.

Without execution error, gradual-demand tasks use mean demand as the state and evolve according to

$$s_{t+1} = (1 - \alpha)s_t + \alpha(b - \varepsilon\theta_t), \qquad x_t \sim \mathcal{N}(s_{t+1}, \sigma^2 I). \tag{A2}$$

The initial state is $s_1 = b$. Components of b are sampled uniformly from $[6, 7)$ and fixed by the environmental seed of each trajectory. Adaptation rates are 0.4, 0.1 and 0.2 for the fast three-dimensional, slow three-dimensional and ten-dimensional tasks, respectively. The default response matrix is the identity scaled by $\varepsilon = 1.5$, with noise standard deviation 0.5. Demand is not clipped to nonnegative values; this Gaussian demand model studies continuous response mechanisms.

For a fixed price θ, the stationary mean is $s^*(\theta) = b - \varepsilon\theta$. Current population risk is $-\theta_t^\top s_{t+1}$, and stationary risk is $-\theta^\top s^*(\theta)$. These quantities describe the current and fully adapted populations, respectively.

A.4 Classification and Regression

Classification-2D parameterizes the intercept and slope as $\theta = (\theta_{\text{int}}, \theta_{\text{slope}})^\top \in [-3, 3]^2$. Labels 0 and 1 are equally likely. Writing the class label as y, the raw feature a follows $\mathcal{N}(-1, 0.6^2)$ for label 0 and $\mathcal{N}(1 + 1.2\theta_{\text{slope}}, 0.6^2)$ for label 1. With design vector $x = (1, a)^\top$ and record $z = (x, y)$, the loss is

$$\ell(\theta; x, y) = \log(1 + \exp(x^\top\theta)) - yx^\top\theta + \frac{0.03}{2}\|\theta\|_2^2. \tag{A3}$$

Classification population risk is computed by trapezoidal integration over a fixed grid of 4,097 points on $[-8, 8]$. This is the protocol's numerical risk estimate; observed loss is computed directly from the current batch.

Regression-1D uses $\theta \in [-3, 3]$, $x \sim \mathcal{N}(0, 1)$ and $y = (1 - 1.5\theta)x + 0.3\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$. The loss is $\frac{1}{2}(\theta x - y)^2$, giving

$$R(\theta) = \frac{1}{2}\big((2.5\theta - 1)^2 + 0.09\big). \tag{A4}$$

The optimal parameter is 0.4, and minimum risk is 0.045. Bandit-regression-1D uses the same environment but reveals only scalar loss to the learner.

A.5 Finite-Horizon Frozen Risk and Stationary Risk

Proposition A.2 (The transient term in frozen evaluation). Fix a rule θ. Suppose the deterministic population update F_θ is contractive under the chosen norm, with factor $0 \le \gamma < 1$ and fixed point s^*. Assume the state-conditional risk $g_\theta(s) = \mathbb{E}_{z\sim D(s,\theta)}[\ell(\theta, z)]$ is L_g-Lipschitz in state. Starting from the pre-freeze state $\bar{s}_0$, define $\bar{s}_k = F_\theta(\bar{s}_{k-1})$. Then

$$\left|\frac{1}{H}\sum_{k=1}^{H} g_\theta(\bar{s}_k) - g_\theta(s^*)\right| \le \frac{L_g\|\bar{s}_0 - s^*\|}{H}\frac{\gamma(1 - \gamma^H)}{1 - \gamma}. \tag{A5}$$

Proof. Contractivity gives $\|\bar{s}_k - s^*\| \le \gamma^k\|\bar{s}_0 - s^*\|$. Lipschitz continuity bounds the absolute risk difference at each step by $L_g\gamma^k\|\bar{s}_0 - s^*\|$. Summing over $k = 1, \dots, H$ and applying the geometric-series formula yields the result. This completes the proof.

For the mean-demand process in equation (A2), $\gamma = 1 - \alpha$, and the Euclidean norm permits $L_g = \|\theta\|_2$. Frozen evaluation therefore depends on both the final action and the population state at the start of freezing. Slower adaptation makes the finite-horizon transient term more pronounced. This bound separates the transient effect of the initial state from stationary performance.

A.6 Analytical Risk References and Task Difficulty

When an environment's risk function is known, an attainable-risk reference gives learning performance an explicit scale. Table A1 lists optimal fixed rules in unperturbed environments. Sample and scalar tasks sharing a response model use the same reference. These values follow from the environment equations and are available to the evaluator; learners retain their declared feedback access.

Table A1. Optimal fixed rules and risks for analytical environments. Pricing assumes $\varepsilon > 0$ and action domain $[0, 5]^d$; Π denotes coordinatewise clipping. Stateful tasks fix b through each trajectory's environmental seed.

Environment	Optimal fixed rule	Corresponding risk
Gaussian-1D	$\theta^* = 1/2$	$R^* = -1/4$
Nonlinear-1D	$\theta^* = -2/3$	$R^* = -2/(3\sqrt{3})$
Mixture-1D	$\theta^* = 5/28$	$R^* = -5/224$
Immediate pricing and corresponding scalar tasks	$\theta^* = \Pi_{[0,5]^d}(b/(2\varepsilon))$	$R^* = \varepsilon\|\theta^*\|_2^2 - b^\top\theta^*$
Regression and corresponding scalar task	$\theta^* = 0.4$	$R^* = 0.045$
Stationary risk in stateful pricing	$\theta^* = \Pi_{[0,5]^d}(b/(2\varepsilon))$	$R^*_{\text{stat}} = \varepsilon\|\theta^*\|_2^2 - b^\top\theta^*$

The first three rows follow by differentiating the quadratic risk, $\theta\sqrt{1+\theta}$ and the quadratic mixture-mean risk, respectively. The nonlinear derivative is $(1 + 3\theta/2)/\sqrt{1+\theta}$, changing from negative to positive at $-2/3$. Each pricing coordinate is convex quadratic, so clipping the unconstrained solution gives its constrained minimum. For an interior solution, $R^* = -\|b\|_2^2/(4\varepsilon)$. The regression minimum follows directly from equation (A4).

Response shifts and execution errors. If immediate pricing's response coefficient changes to ε', replacing the table's coefficient with ε' gives the new objective and optimum. Suppose the actual price is $\theta + \delta$, with zero-mean execution noise of covariance Σ, and boundary clipping does not occur. Expected risk is then $R(\theta) + \varepsilon\,\mathrm{tr}(\Sigma)$, so execution errors themselves raise attainable risk. The benchmark uses bounded actions; when clipping occurs, the reference must follow the clipped action distribution. Observation noise leaves the underlying risk function unchanged, primarily altering the learner's ability to identify it.

Dynamic objectives. The stateful-pricing reference is stationary risk after a fixed rule has been deployed long enough for adaptation. Within finite horizons, population state, exploration and the freezing starting point also affect cumulative risk; equation (A5) characterizes the fixed-rule transient term. Classification population risk is evaluated numerically using equation (A3). The analytical references characterize fixed-rule objectives; the shared benchmark measures learning performance over finite deployment trajectories.

A.7 Task and Component Support

Table A2. Environment and component support. All instances support action noise, observation noise and response shifts. External-shock support follows the current mean-response interface. Compatible-method counts refer to combinations actually executed in this benchmark.

Task instance	Dimension	Feedback	Response	External shock	Compatible methods
Gaussian-1D	1	Samples	Immediate	Yes	25
Nonlinear-1D	1	Samples	Immediate	Yes	25
Mixture-1D	1	Samples	Immediate	Yes	25
Pricing-3D	3	Samples	Immediate	Yes	26
Classification-2D	2	Samples	Immediate	—	25
Regression-1D	1	Samples	Immediate	—	25
Stateful-fast-3D	3	Samples	Gradual	Yes	25
Stateful-slow-3D	3	Samples	Gradual	Yes	25
Bandit-pricing-3D	3	Scalar	Immediate	Yes	15
Bandit-regression-1D	1	Scalar	Immediate	—	15
Pricing-10D	10	Samples	Immediate	Yes	26
Stateful-10D	10	Samples	Gradual	Yes	25
Bandit-pricing-10D	10	Scalar	Immediate	Yes	15

A.8 Public Interaction and Evaluation Interfaces

Visible sample batches populate the observation array. Aggregate-only feedback zero-fills that array and sets its mask to zero, retaining the executed rule, observed loss and relative time.

Table A3. Public interaction and evaluation interfaces. Algorithms update from observations; the evaluator separately records the simulator's expected loss.

Interface or object	Meaning in PP-Gym	Research use
`reset(seed)`	Reset environment, random streams and deployment count; return initial observation	Comparisons and replay with the same configuration and seed
`step(action)`	Deploy once, advance the response and return observations; reward is negative observed loss	Run different update rules through a common loop
Action and observation spaces	Parameter bounds, sample shape, observation mask and time	Check inputs and visible information
Termination and truncation	Current tasks truncate at the deployment budget and then require a reset	Match actual deployment counts
Independent evaluation entry point	Record population risk, stationary risk and sample counts	Compare performance and resource use

A.9 Perturbation Controls

Table A4. Implemented perturbations and control parameters. Action noise, observation noise and external shocks support per-step triggering probabilities. A response shift occurs once at a specified deployment and persists thereafter.

Module	Affected object	Configurable conditions	Evaluation
Action noise	Executed parameters, then clipped to action bounds	Relative magnitude, trigger probability, seed	Evaluate the executed action
Observation noise	Public loss and visible samples	Noise magnitude, trigger probability, seed	Evaluate the uncorrupted data distribution
Response shift	Distribution-response coefficients; retain current population state in stateful tasks	Coefficient change, change time	Record risk before and after the change
External shock	Current demand or location in mean-response families	Shock magnitude, trigger probability, seed	Include the shock in actual deployment loss

Appendix B Methods, Information Requirements and Configuration Selection

B.1 Method Mechanisms and Implementations

Table B1 lists every method. S requires scalar feedback, F requires samples, and F+ additionally requires a parametric score. F/S selects the sample or scalar construction according to feedback mode. All environment queries count toward the deployment budget.

Table B1. Methods and information requirements. Adaptation means integrating an update mechanism with the benchmark's normalized coordinates, action constraints and accounting protocol. Numerical reproduction of individual paper protocols is reported separately in Appendix C.

Method	Information	Update mechanism and implementation
Hold	S	Repeatedly deploy the clipped zero action
Random search	S	Explore randomly and retain the action with the lowest observed loss
SPSA	S	Sequential positive and negative Rademacher perturbations with a decaying step size (Spall, 1992)
One-point DFO	S	One-point random-direction estimate, optionally subtracting a moving-average loss (Flaxman et al., 2004)
Two-point ZO	S	Two sequential spherical-direction queries with decaying step size and perturbation radius (Hikima et al., 2025; Duchi et al., 2013)
EXP3	S	Exponential weights over discrete actions; losses scaled by the observed range (Auer et al., 2002b)
UCB	S	Upper-confidence-bound selection over the same discrete action set (Auer et al., 2002a)
GP-LCB	S	Radial-basis-function Gaussian process and lower-confidence-bound candidate search (Srinivas et al., 2009)

Method	Information	Update mechanism and implementation
CMA-ES	S	Adapt mean, covariance and scale; actually deploy every candidate (Hansen, 2016)
Scalar surrogate	S	Quadratic scalar surrogate with one-step temporal dependence
Damped RRM	F	Damped mixture of the current rule and current-batch empirical minimizer (Perdomo et al., 2020)
Delayed RRM	F	Recompute the empirical minimizer after a fixed number of deployments (Brown et al., 2022)
RGD	F	Repeated gradient descent using direct sample gradients (Perdomo et al., 2020; Mendler-Dünner et al., 2020)
Lazy deploy	F	Update candidate parameters by batch gradients; deploy every few rounds (Mendler-Dünner et al., 2020)
Momentum RGD	F	Sample-gradient updates with momentum
Anderson RRM	F	Anderson mixing of the empirical-minimizer map (Walker & Ni, 2011)
Guided ZO	F	Sample-loss gradients guide two-point query directions; a mechanism adaptation (Hikima et al., 2025)
History PerfGD	F	Backend historical-difference response estimation, restricted to demand tasks (Izzo et al., 2021)
Cross-loss (no memory)	F	Quadratic cross-loss surrogate without memory
Cross-loss (diagonal)	F	Cross-loss surrogate with a diagonal quadratic basis
Cross-loss	F/S	Cross-loss surrogate with temporal dependence; scalar construction on the scalar path
TR-CLS	F/S	Response and memory fitting, local search and change resets
TR-CLS (no memory)	F/S	Fix the memory coefficient at zero
TR-CLS (no reset)	F/S	Disable anomaly-triggered resets
TR-CLS (forgetting)	F/S	Replace hard resets with adaptive forgetting weights
Parametric PerfGD + score	F+	Backend parametric PerfGD using additional score structure (Izzo et al., 2021)

The ten S methods and five F/S methods run on all 13 tasks. Nine sample-based methods run on the ten sample-feedback tasks. History PerfGD requires demand samples and runs on five pricing tasks; Parametric PerfGD + score requires additional distribution information and runs on seven stateless sample-feedback tasks. These requirements give $15 \times 13 + 9 \times 10 + 5 + 7 = 297$ combinations. Tables D1–D3 report their complete online-loss results.

EXP3 and UCB use 11 or 21 evenly spaced actions in one dimension. In multiple dimensions, they use 12 or 24 Latin-hypercube actions including the center. GP-LCB standardizes losses and fits fixed kernel parameters. Its candidates comprise 256 global random points, 64 perturbations near the incumbent and the incumbent itself. CMA-ES forms each generation through consecutive deployments; action clipping and resulting losses are part of its actual interaction. In stateful environments, the positive and negative queries of two-point methods advance the population sequentially.

B.2 Development Grids

The tuples in Table B2 specify joint candidates, not Cartesian products of individual parameter values. Each candidate uses development seeds 20, 21, 22 and 23. The final configuration is selected by mean online loss within each task. Standard parameters use normalized coordinates, while native backend learners follow their published update rules.

Table B2. Development candidates for methods without a trust region.

Method	Candidate configurations
Hold	Fixed policy; no hyperparameter selection
Random search	Exploration probability: 0.1, 0.2, 0.4
SPSA	(Step size, perturbation): (0.005,0.1), (0.02,0.1), (0.08,0.1), (0.02,0.3)
One-point DFO	(Step size, control variate): (0.003,on), (0.01,on), (0.03,on), (0.01,off); perturbation 0.1
Two-point ZO	The same four step-size and perturbation pairs as SPSA; update decay 0.98 per round
EXP3	(Mixing rate, number of actions): (0.1,12), (0.2,12), (0.4,12), (0.2,24)
UCB	(Exploration coefficient, number of actions): (0.5,12), (1,12), (2,12), (1,24)
GP-LCB	(Kernel length scale, confidence coefficient): (0.2,2), (0.4,2), (0.8,2), (0.4,4)
CMA-ES	Initial scale 0.2, 0.3 or 0.5; an additional candidate uses initial scale 0.3 and population size 6
Damped RRM	Update rate: 0.1, 0.3, 0.6, 1.0
Delayed RRM	Update interval: 2, 3, 5, 8
RGD	Step size: 0.01, 0.03, 0.1, 0.3
Lazy deploy	(Step size, deployment interval): (0.03,3), (0.1,3), (0.3,3), (0.1,6)
Momentum RGD	(Step size, momentum): (0.03,0.5), (0.1,0.5), (0.03,0.9), (0.1,0.9)
Anderson RRM	(History length, mixing rate): (2,1), (3,1), (5,1), (3,0.5)
Guided ZO	(Step size, guidance strength): (0.005,0.75), (0.02,0.75), (0.08,0.75), (0.02,0.5); perturbation 0.1, decay 0.98, guidance decay 0.5
History PerfGD	(Step size, history length): (0.01,25), (0.03,25), (0.1,25), (0.03,80)
Parametric PerfGD + score	(Step size, history length): (0.01,20), (0.03,20), (0.1,20), (0.03,80)
Scalar surrogate and the three Cross-loss methods	(Ridge coefficient, exploration perturbation): (0.001,0.02), (0.01,0.02), (0.1,0.02), (0.01,0.08); history length 80

TR-CLS shares the following settings across candidates: maximum radius 0.5, minimum radius 0.02, ridge coefficient 0.01 and relative predicted-improvement threshold 10^{-3}. Reset monitoring begins after 8 residual records; 3 consecutive absolute residuals above 4 times the reference scale trigger a reset. The local bandwidth has a lower bound of 0.25. The five candidates are listed below.

Table B3. TR-CLS candidate configurations. Global probing constructs batch-loss probe points across the full action range; local probing uses radius 0.3. A diagonal quadratic basis is used above three dimensions.

Config.	Initial radius	Perturb. cap	Batch probes	Basis	Additional initialization
C1	0.2	0.03	Global	Quadratic	Active exploration when under-determined
C2	0.1	0.03	Local	Quadratic	Default
C3	0.2	0.01	Global	Quadratic	Default
C4	0.3	0.01	Local	Cubic	Default
C5	0.5	0.01	Local	Cubic	Early empirical-loss initialization

In the main benchmark, the no-memory and no-reset variants are each tuned over C1 to C3. The forgetting variant is tuned over C1 to C5 with hard resets disabled. These comparisons assess each method after its own hyperparameter selection. The matched ablations in Appendix E inherit the configuration selected for full TR-CLS and change one component at a time.

Table B4. Final configurations of full TR-CLS for each task. ERM denotes empirical risk minimization.

Task	Config.	Initial radius	Perturb. cap	Probes	Low-dimensional basis	Early ERM
Gaussian-1D	C2	0.1	0.03	Local	Quadratic	Off
Nonlinear-1D	C5	0.5	0.01	Local	Cubic	On
Mixture-1D	C1	0.2	0.03	Global	Quadratic	Off
Pricing-3D	C4	0.3	0.01	Local	Cubic	Off
Classification-2D	C5	0.5	0.01	Local	Cubic	On
Regression-1D	C3	0.2	0.01	Global	Quadratic	Off
Stateful-fast-3D	C3	0.2	0.01	Global	Quadratic	Off
Stateful-slow-3D	C3	0.2	0.01	Global	Quadratic	Off
Bandit-pricing-3D	C1	0.2	0.03	Global	Quadratic	Off
Bandit-regression-1D	C3	0.2	0.01	Global	Quadratic	Off
Pricing-10D	C4	0.3	0.01	Local	Cubic	Off
Stateful-10D	C4	0.3	0.01	Local	Cubic	Off
Bandit-pricing-10D	C1	0.2	0.03	Global	Quadratic	Off

APPENDIX C NUMERICAL REPRODUCTION AND COMPONENT INTEROPERABILITY

C.1 INDEPENDENT NUMERICAL REPRODUCTION

PP-Gym independently implements environments, algorithms and protocols from 14 research lines (Table C1). Reproduction covers experiments with publicly specified numerical protocols and results, comparing metrics and trajectories under the corresponding settings. Reference values come from published numerical results or public author implementations; PP-Gym generates its results through independent components.

Table C1. Research problems covered by the independent backend. Conference and journal papers from the same research line are merged into one target, yielding 14 targets. These targets validate the underlying implementation; Appendix A specifies the tasks used in the shared benchmark.

Research problem	References	Numerical validation scope
Repeated risk minimization	(Perdomo et al., 2020)	Distribution response and repeated training in credit decisions
Stochastic PP optimization	(Mendler-Dünner et al., 2020)	Gaussian responses, online data and deployment schedules
Response-gradient estimation	(Izzo et al., 2021)	Parametric responses and gradient-learning trajectories
Stateful PP	(Brown et al., 2022)	Delayed deployment and gradual population adaptation
Gradual distribution response	(Izzo et al., 2022)	Response protocols and optimization trajectories
Bandit PP	(Chen et al., 2024)	Reparameterization and two-point zeroth-order feedback
Guided zeroth-order optimization	(Hikima et al., 2025)	Estimation and updates in pricing and strategic classification
Neural PP	(Mofakhami et al., 2023)	Neural predictors and repeated resampling for training
Network effects	(Wang et al., 2023)	Risk expressions and interactions in multi-layer networks
Multiplayer learning	(Narang et al., 2022; 2023)	Monotone games and ridesharing games
Recommendation feedback	(Eilat & Rosenfeld, 2023)	Content diversity and preference-encoding updates
Non-convex optimization	(Li & Wai, 2024)	Finite data streams, optimizers and deployment processes
Games and mechanism design	(Góis et al., 2025)	Fixed, repeated-training, trust-policy and neural learners
Binary risk learning	(Tsoy et al., 2025)	Binary population responses and learning rules

Table C2 reports numerical agreement for the experiments summarized in Table C1. Errors compare PP-Gym values with their reference values under the same numerical protocol. Tolerances follow the precision of program outputs or published rounded values.

Table C2. Numerical reproduction errors. Counts refer to compared metrics or trajectory values. All listed comparisons satisfy their individual tolerances. For targets with multiple tolerances, the final column gives their largest tolerance; recommendation uses metric-specific tolerances as detailed below.

Research line	Compared values	Maximum absolute error	Tolerance
Repeated risk minimization (Perdomo et al., 2020)	4,016	3.782×10^{-10}	10^{-8}
Stochastic PP optimization (Mendler-Dünner et al., 2020)	9,897	2.776×10^{-17}	10^{-12}
Response-gradient estimation (Izzo et al., 2021)	48,814	1.445×10^{-11}	10^{-9}
Stateful PP (Brown et al., 2022)	1,192,068	1.021×10^{-14}	10^{-8}
Gradual distribution response (Izzo et al., 2022)	13,328	2.430×10^{-7}	10^{-6}
Bandit PP (Chen et al., 2024)	10,200	0	10^{-12}
Guided ZO: program outputs (Hikima et al., 2025)	41,392	1.847×10^{-13}	10^{-10}
Guided ZO: rounded paper values (Hikima et al., 2025)	60	4.860×10^{-3}	5.1×10^{-3}
Neural PP (Mofakhami et al., 2023)	900	4.441×10^{-16}	10^{-9}
Network effects (Wang et al., 2023)	332	0	10^{-12}
Multiplayer learning (Narang et al., 2022; 2023)	107,094	5.002×10^{-12}	10^{-10}
Recommendation feedback (Eilat & Rosenfeld, 2023)	1,270	9.382×10^{-2}	Metric-specific
Non-convex optimization (Li & Wai, 2024)	814,220	5.436×10^{-8}	10^{-5}
Games and mechanism design (Góis et al., 2025)	1,672,357	2.980×10^{-7}	10^{-6}
Binary risk learning (Tsoy et al., 2025)	251,252	3.608×10^{-16}	10^{-12}

Guided ZO also matches 120 reported significance indicators exactly. For recommendation, 976 values match exactly; the largest error is 0.093815 against a tolerance of 0.2, and no comparison uses more than 46.91% of its tolerance. The games and binary-risk counts include finite values; respectively 3,990 and 16 jointly undefined values are excluded. These results validate the numerical implementations underlying the shared benchmark.

C.2 Cross-Component Execution

Table C3 demonstrates component interoperability across three response environments. Each compatible pair uses two seeds, four deployments, dimension 3 and 24 samples per deployment. All 16 pairs complete the requested trajectories and return finite risks through the common comparison interface.

Table C3. Composition checks through the public interface. A dash indicates that the combination was not run in this short check. Learners requiring additional response or state gradients are labeled explicitly.

Environment	ERM	RRM	RGD	Lazy deploy	PerfGD (response gradient)	Stateful PerfGD (state gradient)
Gaussian location shift	Pass	Pass	Pass	Pass	Pass	—
Gaussian location–scale shift	Pass	Pass	Pass	Pass	Pass	—
Gradual location shift	Pass	Pass	Pass	Pass	Pass	Pass

The response- and state-gradient learners in this test use structural information beyond the public feedback available to the main benchmark baselines.

C.3 Gym Interaction Semantics

All 13 tasks satisfy the declared observation spaces, feedback masks, evaluator separation, reward signs, executed-action records, deployment limits and sample counts. Two independent instances with seed 731 produce the same trajectory under five fixed central actions and 16 samples per deployment.

Stateless tasks draw 80 observations; stateful tasks draw 96, including one initialization batch hidden from the learner. The environment rejects deployments after the budget is exhausted. These results confirm the interaction semantics in Appendix A.1.

Appendix D Complete Main Results and Statistical Definitions

D.1 Online Loss by Task

Tables D1 to D3 report mean online observed loss for all 26 methods across 13 tasks, with 20 seeds per valid cell. Dashes indicate combinations outside the declared applicability rules.

Table D1. Location-response, pricing and classification tasks.

Method	Gaussian-1D	Nonlinear-1D	Mixture-1D	Pricing-3D	Classification-2D
Hold	0.0000	0.0000	0.0000	0.0000	0.6931
Random search	0.0402	-0.1666	0.0934	-19.0993	0.3926
SPSA	-0.2231	-0.3487	-0.0072	-23.6255	0.1899
One-point DFO	-0.2005	-0.3362	0.0060	-20.4717	0.1774
Two-point ZO	-0.2287	-0.3591	-0.0111	-23.3377	0.1540
EXP3	1.0700	0.5802	0.7944	-17.6189	0.8084
UCB	0.5590	0.1579	0.4694	-20.5140	0.5340
GP-LCB	-0.1974	-0.3324	0.0158	-23.0366	0.1908
CMA-ES	-0.1998	-0.3272	-0.0054	-19.2997	0.2240
Scalar surrogate	-0.1213	-0.2847	0.0473	-23.0398	0.2655
Damped RRM	-0.0759	-0.2422	0.0045	-7.4893	0.1261
Delayed RRM	3.7372	-0.2089	2.6169	2.3382	0.1315
RGD	-0.2027	-0.2926	-0.0177	-19.1209	0.1570
Lazy deploy	-0.1261	-0.2415	-0.0151	-9.3276	0.1633
Momentum RGD	-0.0631	-0.2252	-0.0084	-5.6866	0.1538
Anderson RRM	0.0340	-0.2118	0.0608	-4.5388	0.1258
Guided ZO	-0.2287	-0.3593	-0.0113	-23.4934	0.1483

Method	Gaussian-1D	Nonlinear-1D	Mixture-1D	Pricing-3D	Classification-2D
History PerfGD	—	—	—	-20.4560	—
Cross-loss (no memory)	-0.1370	-0.3045	0.0506	-23.0737	0.2266
Cross-loss (diagonal)	-0.1371	-0.3048	0.0516	-23.5702	0.2513
Cross-loss	-0.1371	-0.3048	0.0516	-23.0713	0.2308
TR-CLS	-0.2397	-0.3753	-0.0191	-24.4073	0.1275
TR-CLS (no memory)	-0.2416	-0.3697	-0.0203	-24.3526	0.1406
TR-CLS (no reset)	-0.2397	-0.3738	-0.0191	-24.3620	0.1396
TR-CLS (forgetting)	-0.2401	-0.3756	-0.0200	-24.4118	0.1274
Parametric PerfGD + score	-0.2419	-0.3699	-0.0209	-23.8719	0.2113

Table D2. Regression, stateful and three-dimensional bandit tasks.

Method	Regression-1D	Stateful-fast-3D	Stateful-slow-3D	Bandit-pricing-3D	Bandit-regression-1D
Hold	0.5466	0.0000	0.0000	0.0000	0.5466
Random search	1.4076	-11.9181	-17.8673	-19.0993	1.4076
SPSA	0.2057	-18.3840	-18.5023	-23.6255	0.2057
One-point DFO	0.3412	-15.9503	-17.3477	-20.4717	0.3412
Two-point ZO	0.1626	-17.5300	-18.5624	-23.3377	0.1626
EXP3	8.5421	-17.3044	-21.6335	-17.6189	8.5421
UCB	5.1866	-17.8246	-21.9633	-20.5140	5.1866
GP-LCB	0.4223	-15.7255	-20.2349	-23.0366	0.4223
CMA-ES	0.1921	-11.3205	-11.2231	-19.2997	0.1921
Scalar surrogate	0.7630	-16.0939	-14.1056	-23.0398	0.7630
Damped RRM	0.0495	-6.3394	-7.3389	—	—
Delayed RRM	20.6906	-6.8695	-12.2017	—	—
RGD	0.0496	-16.5167	-18.0077	—	—
Lazy deploy	0.0591	-7.1870	-7.0002	—	—
Momentum RGD	0.0527	-3.2985	-4.9912	—	—
Anderson RRM	0.0493	-4.1870	-6.9387	—	—
Guided ZO	0.1620	-17.0719	-17.1316	—	—
History PerfGD	—	-17.7336	-19.2535	—	—
Cross-loss (no memory)	0.7359	-17.9235	-14.2709	—	—
Cross-loss (diagonal)	0.7541	-18.6035	-16.1700	—	—
Cross-loss	0.7541	-16.3077	-13.3456	-23.0398	0.7630
TR-CLS	0.0534	-20.4938	-21.7510	-24.1599	0.1094
TR-CLS (no memory)	0.0576	-20.1040	-10.0420	-24.1505	0.1252
TR-CLS (no reset)	0.0534	-20.5020	-22.0514	-24.1599	0.1094
TR-CLS (forgetting)	0.0532	-20.5600	-22.0119	-24.1540	0.1414
Parametric PerfGD + score	0.0523	—	—	—	—

Table D3. Ten-dimensional tasks.

Method	Pricing-10D	Stateful-10D	Bandit-pricing-10D
Hold	0.0000	0.0000	0.0000
Random search	-59.7463	-56.8649	-59.7463
SPSA	-72.2043	-59.0252	-72.2043

Method	Pricing-10D	Stateful-10D	Bandit-pricing-10D
One-point DFO	-47.6720	-46.9330	-47.6720
Two-point ZO	-72.0283	-49.5876	-72.0283
EXP3	-57.0115	-66.0404	-57.0115
UCB	-69.1126	-67.7067	-69.1126
GP-LCB	-72.2496	-61.6014	-72.2496
CMA-ES	-59.6724	-46.2319	-59.6724
Scalar surrogate	-49.1120	-65.4042	-49.1120
Damped RRM	-29.5413	-18.5122	—
Delayed RRM	3.8853	-31.3700	—
RGD	-64.9848	-56.9737	—
Lazy deploy	-36.1403	-23.9799	—
Momentum RGD	-25.7130	-10.6342	—
Anderson RRM	-20.6170	-17.3054	—
Guided ZO	-68.6904	-58.1666	—
History PerfGD	-62.6472	-55.3730	—
Cross-loss (no memory)	-49.1120	-65.4042	—
Cross-loss (diagonal)	-71.6849	-57.2465	—
Cross-loss	-49.1120	-65.4042	-49.1120
TR-CLS	-82.6016	-70.7753	-81.0350
TR-CLS (no memory)	-82.4309	-34.4294	-81.0441
TR-CLS (no reset)	-82.3868	-70.9656	-81.0350
TR-CLS (forgetting)	-82.6364	-71.1915	-81.0394
Parametric PerfGD + score	-75.6150	—	—

D.2 Paired Statistics and Multiple Comparisons

For task j and seed k, let $d_{j,k} = \widehat{L}^{\mathrm{TR}}_{T,j,k} - \widehat{L}^{\mathrm{cmp}}_{T,j,k}$ be TR-CLS minus its selected comparator in mean online observed loss (equation (2)). Tables D4 and D5 report its seed mean $\bar{d}_j$ and interval $\bar{d}_j \pm t_{0.975,19} s_{d,j}/\sqrt{20}$, where $s_{d,j}$ is the sample standard deviation of the 20 paired differences and $t_{0.975,19}$ is the 97.5th percentile of Student's t distribution with 19 degrees of freedom. Within each comparison family, the m raw p-values are sorted in ascending order. Writing $p_{(i)}$ for the ith ordered value, the Holm-adjusted value (Holm, 1979) is

$$\widetilde{p}_{(i)} = \min\left\{1, \max_{1\le k\le i}(m-k+1)p_{(k)}\right\}. \tag{D1}$$

The public-feedback comparisons and the comparisons allowing additional score information each form a separate family with $m = 13$; matched ablations use $m = 19$. In all tables, "lower" and "higher" depend jointly on the adjusted value and the sign of the difference. Time points from the same trajectory are correlated. We therefore compute pointwise intervals across independent seeds and do not treat deployments as independent samples.

D.3 Primary Comparisons under Public Feedback

Table D4 reports absolute loss differences, paired intervals and Holm-adjusted conclusions for the public-feedback comparisons in Figure 3. For each task, the comparator has the lowest development-set online loss among methods outside the cross-loss surrogate family that use the task's public observations without additional score information.

Table D4. Paired differences under the same public feedback mode. Each row uses 20 seeds. Intervals are not adjusted for simultaneous coverage; the final column uses Holm correction.

Task	Comparator	Difference and 95% interval	Holm conclusion
Gaussian-1D	Guided ZO	-0.0110 [-0.0122, -0.0098]	Lower
Nonlinear-1D	Guided ZO	-0.0160 [-0.0182, -0.0139]	Lower
Mixture-1D	RGD	-0.0014 [-0.0027, -0.0001]	Not separated
Pricing-3D	Guided ZO	-0.9139 [-1.0398, -0.7880]	Lower
Classification-2D	Anderson RRM	+0.0016 [+0.0004, +0.0029]	Higher
Regression-1D	RGD	+0.0038 [+0.0034, +0.0042]	Higher
Stateful-fast-3D	SPSA	-2.1098 [-2.4473, -1.7724]	Lower
Stateful-slow-3D	UCB	+0.2123 [-0.2388, +0.6635]	Not separated
Bandit-pricing-3D	SPSA	-0.5344 [-0.6134, -0.4553]	Lower
Bandit-regression-1D	Two-point ZO	-0.0532 [-0.0618, -0.0445]	Lower
Pricing-10D	GP-LCB	-10.3520 [-10.9494, -9.7546]	Lower
Stateful-10D	UCB	-3.0686 [-3.7493, -2.3878]	Lower
Bandit-pricing-10D	GP-LCB	-8.7854 [-9.4209, -8.1498]	Lower

For the Gaussian-mixture task, the pointwise interval excludes zero, but the methods are not separated after Holm correction across the 13 comparisons. The interval describes an individual comparison; Holm correction controls the family-wise false-positive rate.

D.4 Comparisons Allowing Additional Score Information

Table D5 applies the same development-set selection rule while allowing comparators that use a parametric distribution score. TR-CLS and all candidate comparators retain their configurations and test trajectories. The expanded comparator pool defines a separate comparison from the public-feedback analysis in Figure 3 and Table D4.

Table D5. Paired comparisons allowing additional score information. Lower loss is better. Differences are TR-CLS minus comparator, with 20 paired seeds per row. F and S label the task's available sample and scalar feedback; F+ adds the comparator's score access. We report per-comparison 95% intervals and conclusions after Holm correction across 13 comparisons.

Task	Comparator (information)	Comparator	TR-CLS	Difference and 95% interval	Outcome
Gaussian-1D	PerfGD + score (F+)	-0.2419	-0.2397	+0.0021 [+0.0007, +0.0035]	Higher
Nonlinear-1D	PerfGD + score (F+)	-0.3699	-0.3753	-0.0055 [-0.0078, -0.0031]	Lower
Mixture-1D	PerfGD + score (F+)	-0.0209	-0.0191	+0.0018 [+0.0005, +0.0031]	Higher
Pricing-3D	PerfGD + score (F+)	-23.8719	-24.4073	-0.5354 [-0.6022, -0.4686]	Lower
Classification-2D	Anderson RRM (F)	0.1258	0.1275	+0.0016 [+0.0004, +0.0029]	Higher
Regression-1D	RGD (F)	0.0496	0.0534	+0.0038 [+0.0034, +0.0042]	Higher
Stateful-fast-3D	SPSA (F)	-18.3840	-20.4938	-2.1098 [-2.4473, -1.7724]	Lower
Stateful-slow-3D	UCB (F)	-21.9633	-21.7510	+0.2123 [-0.2388, +0.6635]	Not separated
Bandit-pricing-3D	SPSA (S)	-23.6255	-24.1599	-0.5344 [-0.6134, -0.4553]	Lower
Bandit-regression-1D	Two-point ZO (S)	0.1626	0.1094	-0.0532 [-0.0618, -0.0445]	Lower
Pricing-10D	PerfGD + score (F+)	-75.6150	-82.6016	-6.9866 [-7.2082, -6.7650]	Lower
Stateful-10D	UCB (F)	-67.7067	-70.7753	-3.0686 [-3.7493, -2.3878]	Lower
Bandit-pricing-10D	GP-LCB (S)	-72.2496	-81.0350	-8.7854 [-9.4209, -8.1498]	Lower

APPENDIX E PERTURBATIONS, FROZEN EVALUATION AND COMPLETE ABLATIONS

E.1 SWEEP CONFIGURATIONS AND FEEDBACK PATHS

Sweeps use seeds 520 to 529 and 120 deployments, with 128 samples per deployment except in the sample-size sweep. Table E1 lists the full grid. The six-task group comprises Pricing-3D, Stateful-fast-3D, Classification-2D, Regression-1D, Gaussian-1D and Bandit-pricing-3D. The shock group excludes classification and regression.

The sweep methods are Hold, SPSA, Two-point ZO, UCB, GP-LCB, Damped RRM, RGD, Anderson RRM, Guided ZO, History PerfGD, Parametric PerfGD + score and four TR-CLS methods. Each task filters methods according to the capability rules in Appendix B. The observation-noise sweep additionally excludes three methods that read native statistics. Appendix H summarizes the run counts for each sweep.

Table E1. Controlled sweeps. All levels use the configurations selected during development on the corresponding clean task.

Variation	Levels or values	Tasks	Additional settings
Action noise	0, 0.05, 0.1, 0.2, 0.4	Six-task group	Applied independently at each deployment
Observation noise	0, 0.1, 0.3, 1.0	Six-task group	Excludes methods that read clean native statistics
Response change	0, 0.5, 1.0, 2.0	Six-task group	Permanently changes the response coefficient after 60 deployments
External shock	0, 0.1, 0.3, 0.6	Four mean-response tasks	Occurs with probability 0.2 at each deployment
Response strength	0.5, 1.0, 1.5, 2.0, 3.0	Pricing-3D	Directly changes the demand-response coefficient
Adaptation rate	0.05, 0.1, 0.2, 0.4, 0.8	Stateful-fast-3D	Changes the population-state update rate
Samples per deployment	16, 32, 64, 128, 256	Three-dimensional pricing, regression and classification	Fixed deployment budget

Action noise adds independent Gaussian perturbations to each coordinate. Its standard deviation equals the noise level times the action-range width, and the result is clipped to the action space. The environment responds to the executed action, which is recorded in public feedback. Some learners still fit their histories using proposed actions, so this sweep also reveals how they handle execution error.

Observation noise adds zero-mean Gaussian noise to scalar losses. Its standard deviation equals the noise level times the larger of the absolute first-deployment loss before observation noise and 1. Each sample-feedback data entry also receives Gaussian noise with standard deviation equal to the noise level. Evaluation uses the true environment risk. The native feedback paths of RGD, History PerfGD and parametric PerfGD read clean underlying statistics, so these methods are excluded from the shared perturbed-feedback sweep.

A response change of relative magnitude κ multiplies the relevant coefficient by $1+\kappa$. Stateful environments modify the demand-response matrix while retaining the current population state. Stateless environments update distribution parameters while preserving the continuity of the random stream and counters.

External shocks act on an already sampled batch. With probability 0.2, an independent uniform sign $e_t \in \{-1, 1\}$ and strength ζ give amplitude $a_t = \zeta e_t(\operatorname{mean}|X_t| + 10^{-9})$, where X_t contains the pre-shock demand or feature entries and the mean is over all entries; otherwise $a_t = 0$. Let T_a add a to every feature or demand coordinate while retaining labels. The batch entering loss calculation and feedback is $B_t^{\mathrm{sh}} = T_{a_t}(B_t)$, with B_t sampled as in equation (1). Because a_t depends on the whole batch, the transformed records are generally dependent.

For a realized amplitude, evaluation holds a_t fixed and integrates over a fresh draw from the base distribution. This defines the shifted-distribution reference risk

$$r_t^{\mathrm{sh}} = \mathbb{E}_{z\sim D_t(s_{t+1},\widetilde{\theta}_t)}[\ell(\widetilde{\theta}_t, T_{a_t}(z))] = r_t + \begin{cases} -a_t \mathbf{1}^\top \widetilde{\theta}_t, & \text{pricing,} \\ +a_t \mathbf{1}^\top \widetilde{\theta}_t, & \text{location-response tasks.} \end{cases} \tag{E1}$$

Here $\mathbf{1}$ is the all-ones vector. This reference risk is distinct from conditioning the sampled batch on its own amplitude. The same additive loss shift gives the realized batch loss used for feedback; the

shock does not accumulate in the next population state. Native structural-statistic paths read pre-shock statistics, so shared-feedback shock comparisons use methods that update from the perturbed observations.

E.2 Budget Prefixes and Continued Deployment

Budget curves report prefix-averaged population risk through deployment t. Each prefix uses the same learning trajectory and full-budget configuration. Frozen evaluation reconstructs the post-learning population by replaying the learning actions, then holds the final rule fixed for 40 further deployments while the environment continues to respond. Learning and frozen risks are averaged separately, as defined in Section 2.4.

Table E2. Representative population risks during online and frozen evaluation. Entries are means with half-widths of per-comparison 95% intervals, using 20 seeds per cell.

Task	Method	Online population risk	Frozen population risk
Bandit-regression-1D	TR-CLS	0.1093 ± 0.0089	0.0615 ± 0.0122
Bandit-regression-1D	Two-point ZO	0.1618 ± 0.0001	0.0697 ± 0.0013
Bandit-regression-1D	CMA-ES	0.1890 ± 0.0299	0.0458 ± 0.0007
Stateful-slow-3D	TR-CLS	-21.7556 ± 0.5595	-19.6350 ± 1.2578
Stateful-slow-3D	UCB	-21.9673 ± 0.4644	-14.8384 ± 2.0750
Stateful-slow-3D	SPSA	-18.5093 ± 0.6803	-13.0876 ± 1.6097

E.3 Single-Component Ablations with Fixed Configurations

Ablations use seeds 600 to 619, with task and full-method configurations fixed as specified in Appendices A and B. The four clean conditions are Stateful-fast-3D, Stateful-slow-3D, Stateful-10D and Bandit-regression-1D. The two changing conditions are Pricing-3D and Stateful-fast-3D, where the response coefficient triples after deployment 60.

No memory fixes $\rho_t = 0$; No reset disables anomaly-triggered resets; Fixed radius sets both radius bounds to the selected initial radius. Recent 60 limits retained history to the latest 60 deployments in scalar regression only. All other parameters and implementation details remain unchanged. Each condition includes the full method and three standard ablations; scalar regression additionally includes Recent 60, yielding $6 \times 4 \times 20 + 20 = 500$ runs.

Table E3. All fixed-configuration ablations. Differences are variant minus full method. Clean environments use online population risk; response-change conditions use post-change population risk. Each comparison uses 20 paired seeds. Asterisks indicate $p < 0.05$ after Holm correction across 19 comparisons.

Task	Condition	Single-component change	Full method	Variant	Paired difference and 95% interval	Adjusted p
Bandit-regression-1D	Clean	Fixed radius	0.1068	0.1378	+0.0310 [+0.0071, +0.0549]	0.15
Bandit-regression-1D	Clean	No memory	0.1068	0.1253	+0.0185 [+0.0056, +0.0313]	0.0929
Bandit-regression-1D	Clean	No reset	0.1068	0.1068	+0.0000 [+0.0000, +0.0000]	1
Bandit-regression-1D	Clean	Recent 60	0.1068	0.1600	+0.0531 [+0.0338, +0.0725]	0.000237 *
Pricing-3D	Post-change	Fixed radius	-4.2934	-5.1802	-0.8868 [-3.0423, +1.2686]	1

Task	Condition	Single-component change	Full method	Variant	Paired difference and 95% interval	Adjusted p
Pricing-3D	Post-change	No memory	-4.2934	-6.3459	-2.0525 [-4.1459, +0.0408]	0.542
Pricing-3D	Post-change	No reset	-4.2934	6.0242	+10.3176 [+7.5991, +13.0361]	2.98e-06 *
Stateful-10D	Clean	Fixed radius	-70.2803	-70.2070	+0.0733 [-0.4627, +0.6093]	1
Stateful-10D	Clean	No memory	-70.2803	-32.0921	+38.1881 [+35.4574, +40.9188]	5.15e-16 *
Stateful-10D	Clean	No reset	-70.2803	-70.4542	-0.1739 [-0.5380, +0.1901]	1
Stateful-fast-3D	Clean	Fixed radius	-20.2666	-20.5160	-0.2494 [-0.4323, -0.0665]	0.122
Stateful-fast-3D	Clean	No memory	-20.2666	-19.7874	+0.4793 [+0.2305, +0.7280]	0.00997 *
Stateful-fast-3D	Clean	No reset	-20.2666	-20.2666	+0.0000 [+0.0000, +0.0000]	1
Stateful-fast-3D	Post-change	Fixed radius	-2.0957	-4.6409	-2.5453 [-5.7562, +0.6656]	0.908
Stateful-fast-3D	Post-change	No memory	-2.0957	0.7042	+2.7999 [-0.5570, +6.1568]	0.873
Stateful-fast-3D	Post-change	No reset	-2.0957	8.2530	+10.3487 [+8.4525, +12.2448]	1.01e-08 *
Stateful-slow-3D	Clean	Fixed radius	-21.5179	-21.5895	-0.0715 [-0.5138, +0.3707]	1
Stateful-slow-3D	Clean	No memory	-21.5179	-9.2975	+12.2204 [+11.6712, +12.7697]	9.03e-20 *
Stateful-slow-3D	Clean	No reset	-21.5179	-21.7789	-0.2609 [-0.6067, +0.0848]	0.915

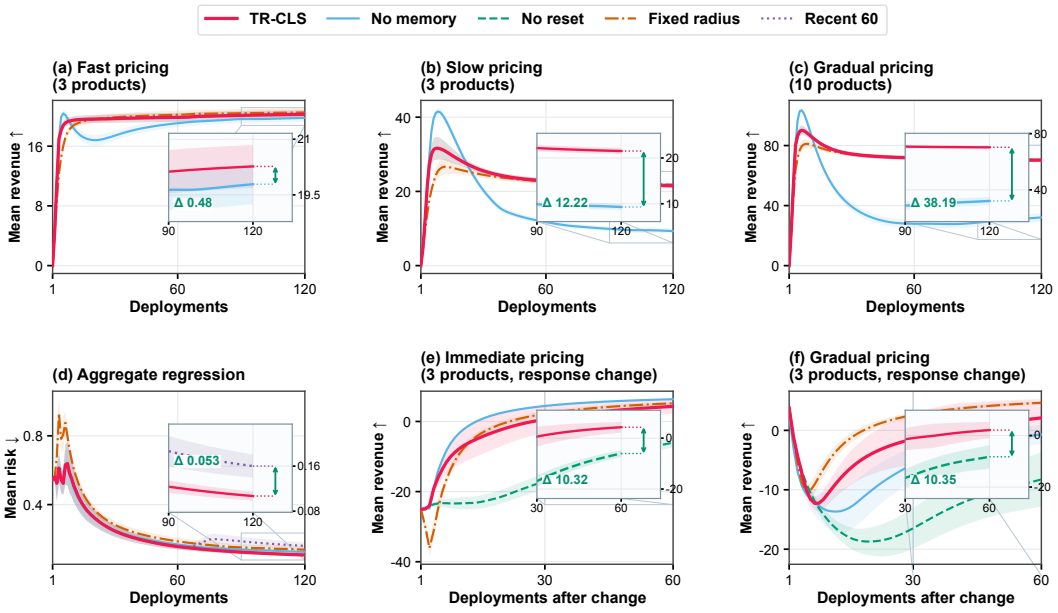


Figure E1. Fixed-configuration ablations across all six conditions. Curves show prefix-averaged population risk, negated for pricing to report mean revenue. Panels (a–d) average from deployment 1; (e,f) average from the response change after deployment 60 and show subsequent deployment counts. Bands are pointwise 95% Student t intervals over 20 seeds. Insets enlarge the last 31 deployments. Arrows give absolute mean risk increases (variant minus full TR-CLS): No memory in (a–c), Recent 60 in (d), and No reset in (e,f), in original loss units. These six contrasts pass Holm correction; Table E3 reports all 19 tests.

E.4 Transient Demand Shocks

Under transient demand shocks with probability 0.2 and strength 0.6, the shifted-distribution reference risks (equation (E1)), averaged over 120 deployments and ten seeds, in Pricing-3D are -22.5890, -21.4605 and -20.7464 for TR-CLS, SPSA and Two-point ZO, respectively. All three update from public perturbed feedback.

E.5 Evaluation Stages across Algorithm Families

Each of the 5,940 learning trajectories is paired with its 40-deployment frozen evaluation, using 20 seeds per task–method combination. Online and frozen performance are ranked separately within each task by mean population risk.

Table E4 compares public-feedback baselines outside the surrogate family, excluding Scalar surrogate, all Cross-loss and TR-CLS variants, and the additional-score method. This leaves nine methods per scalar task and 16 or 17 per sample-feedback task. The table reports the lowest test-mean method at each stage and Spearman rank correlations across these methods.

Table E4. Cross-method rankings at two evaluation stages. All configurations are selected on development data. "Best" denotes the lowest test mean. Entries report descriptive mean rankings.

Task	Meth-ods	Best online mean	Best frozen mean	Rank correla-tion
Bandit-pricing-10D	9	GP-LCB	SPSA	0.933
Bandit-pricing-3D	9	SPSA	GP-LCB	0.883
Bandit-regression-1D	9	Two-point ZO	CMA-ES	0.933
Classification-2D	16	Anderson RRM	Momentum RGD	0.876
Gaussian-1D	16	Guided ZO	GP-LCB	0.853
Mixture-1D	16	RGD	RGD	0.806
Nonlinear-1D	16	Guided ZO	GP-LCB	0.939
Pricing-10D	17	GP-LCB	Guided ZO	0.887
Pricing-3D	17	SPSA	GP-LCB	0.936
Regression-1D	16	Anderson RRM	Lazy deploy	0.938
Stateful-10D	17	UCB	Two-point ZO	0.728
Stateful-fast-3D	17	SPSA	Two-point ZO	0.782
Stateful-slow-3D	17	UCB	History PerfGD	0.838

APPENDIX F COMPLETE CONSTRUCTION OF TR-CLS

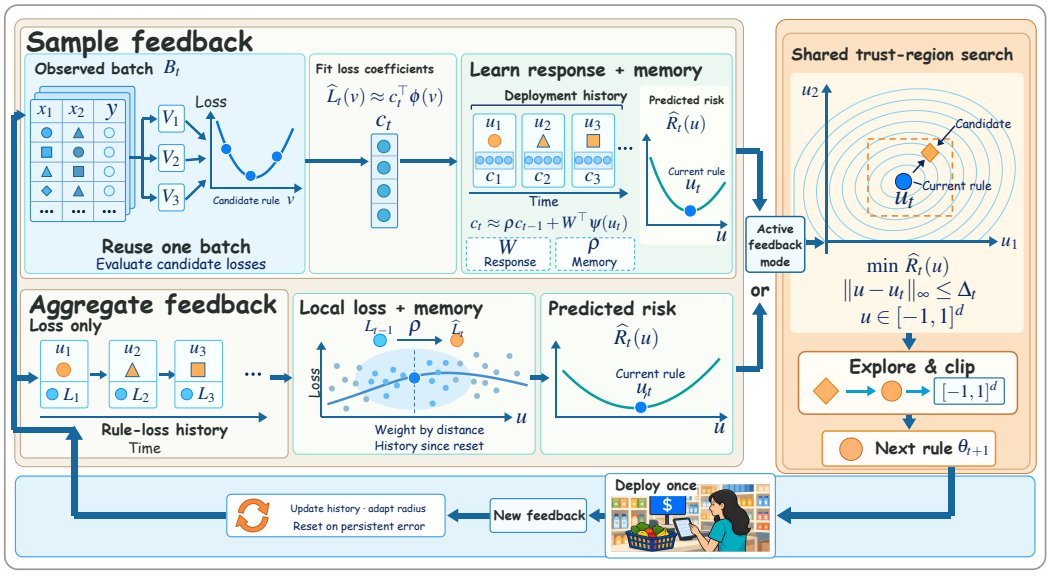


Figure F1. TR-CLS across two feedback modes. The sample path reuses an observed batch to fit loss coefficients, then models their dependence on rules and history. The aggregate path fits loss directly from rule–loss history. Both feed a shared trust-region search; exploration and clipping produce the next rule. Each deployment returns feedback to update the history, radius and change detector. In this schematic, B_t, L_t denote observed feedback and W, ρ denote the current fit W_t, ρ_t. Curves, contours and data glyphs are schematic.

F.1 COORDINATES, BASIS FUNCTIONS AND BATCH REUSE

For action bounds $[a_{\min}, a_{\max}]$ shared across coordinates, define $u = 2(\theta - a_{\min})/(a_{\max} - a_{\min}) - 1$. The inverse map is $\vartheta(u) = a_{\min} + (a_{\max} - a_{\min})(u + 1)/2$, applied coordinatewise. Candidate v is evaluated as $\ell(\vartheta(v), z)$. In up to three dimensions, the quadratic basis contains a constant, linear

terms and all quadratic monomials; the cubic basis additionally includes all cubic monomials. Above three dimensions, a diagonal quadratic basis contains a constant, linear terms and coordinatewise squares.

For q basis functions, $\phi(v), c_t \in \mathbb{R}^q$ and $W_t \in \mathbb{R}^{(d+1)\times q}$. The probe set contains the center, positive and negative coordinate directions, and $\max(12, 3q)$ uniform random points. Local probing scales the set by 0.3 around the current action and clips it to the global range. For probes v_j, the design matrix has rows $\Phi_{j,:} = \phi(v_j)^\top$; least squares fits c_t to their losses on B_t^{obs}. This is the visible batch returned by the observation map, as used in equation (4); fitting consumes no additional deployments.

F.2 Response and Memory Fitting

The sample-feedback path retains the latest 60 action–coefficient records. Candidate memory coefficients are 0, 0.3, 0.5, 0.7, 0.8, 0.9 and 0.95. For each candidate ρ, ridge regression (Hoerl & Kennard, 2000) fits $c_i - \rho c_{i-1}$ against $(1, u_i)$. The ridge coefficient is 0.01, with 10^{-8} applied to the intercept. Selection minimizes the residual sum of squares multiplied by $1 + 0.02\rho$. At fitting time t, the selected ρ_t and coefficients W_t define the equilibrium surrogate in equation (5). Both paths use $\rho_t = 0$ with fewer than four records; the no-memory variant keeps it fixed at zero.

The fitted ρ_t estimates temporal dependence from observed history. The model records normalized submitted rules; under execution error, these can differ from the executed rules returned in observations. Configuration C5 uses the latest batch coefficients as an empirical-loss objective until $d + 2$ deployments are available, then fits the response model.

F.3 Scalar-Feedback Path

At fitting time t, the scalar path fits $L_i^{\text{obs}} \approx \rho_t L_{i-1}^{\text{obs}} + \beta_t^\top \phi(u_i)$ on retained records, using locally weighted ridge regression. Here $\beta_t \in \mathbb{R}^q$ contains the polynomial coefficients. Search uses the equilibrium prediction $\widehat{R}_t(u) = \beta_t^\top \phi(u)/(1 - \rho_t)$. For current rule u_t, let $\nu_{i,t} = \|u_i - u_t\|_\infty$. The weights and bandwidth are

$$w_{i,t} = \exp\left(-\left(\nu_{i,t}/b_t^{\text{loc}}\right)^2\right), \qquad b_t^{\text{loc}} = \max\{2\Delta_t, 0.25, \nu_{(q+2),t}\}. \tag{F1}$$

Here, q is the number of scalar-surrogate basis functions, and $\nu_{(q+2),t}$ is the $(q + 2)$th smallest distance to u_t among retained rules. Local weighting begins with $2q + 2$ observations; earlier fits use equal weights. The scalar path keeps the two records retained at reset and all subsequent history; Recent 60 additionally limits this to the latest 60 deployments.

F.4 Trust Region, Exploration and Change Detection

Given the predicted objective, we generate 32 random candidates and the center within $[-1, 1]^d \cap [u_t - \Delta_t, u_t + \Delta_t]$. The best candidate initializes L-BFGS-B (Byrd et al., 1995), which runs for at most 30 internal iterations. Gaussian exploration with standard deviation $\min(0.25\Delta_t, \text{perturbation cap})$ is then added. After clipping, the normalized rule u_{t+1} gives the submitted rule $\theta_{t+1} = \vartheta(u_{t+1})$. The trust region constrains the optimization candidate; the action after exploration always lies within the global range.

For configuration C1, the model is underdetermined with fewer than $d + 2$ sample-feedback records or $q + 1$ scalar-feedback records. When underdetermined and its predicted improvement is below $10^{-3}(|L_t^{\text{obs}}| + 10^{-3})$, it moves one radius in a random direction before adding Gaussian exploration. This underdetermined exploration step skips radius adaptation. After deployment $t + 1$, actual improvement is $L_t^{\text{obs}} - L_{t+1}^{\text{obs}}$; predicted improvement is $\widehat{R}_t(u_t) - \widehat{R}_t(u_{t+1}^{\text{cand}})$, computed before exploration. When predicted improvement reaches $10^{-3}(|L_{t+1}^{\text{obs}}| + 10^{-3})$, the radius doubles if the actual-to-predicted ratio exceeds 0.5, halves if it falls below 0.1, and stays unchanged otherwise. Below this prediction threshold, the radius halves unless active exploration is enabled and the model is underdetermined. Updates are clipped to $[\Delta_{\min}, \Delta_{\max}]$, with $\Delta_{\min} = 0.02$ and $\Delta_{\max} = 0.5$; the fixed-radius ablation sets both bounds to the selected initial radius.

Each deployment records observed loss minus the preceding surrogate prediction at the action after exploration. After more than eight residuals, the reference scale is their standard deviation over at

most the latest 30 entries, excluding the newest two. Three consecutive absolute residuals exceeding four times this scale trigger a reset: retain the latest two history records and restore the initial radius. The no-reset variant disables this operation.

Adaptive forgetting variant. Hard resets are disabled. For each $\lambda \in \{1, 0.97, 0.9, 0.8, 0.6\}$, response fitting weights retained record i by λ^{t-i} at fitting time t. The scalar path multiplies this by the locality weight in equation (F1). Each candidate uses the same ridge fit and memory-coefficient selection as the full method, with weighted residual sums of squares.

Let $E_{\lambda,t}$ be the candidate's prediction-error score after deployment t, initialized at zero. Its one-step prediction uses the model fitted before that deployment: $\rho_{\lambda,t-1}c_{t-1} + W_{\lambda,t-1}^{\top}\psi(u_t)$ for sample feedback, or $\rho_{\lambda,t-1}L_{t-1}^{\text{obs}} + \beta_{\lambda,t-1}^{\top}\phi(u_t)$ for scalar feedback. The error $\varepsilon_{\lambda,t}$ is the sum of squared coefficient errors against a fresh probe fit on the new batch, or the squared scalar-loss error, respectively. Scores update when candidate models are available:

$$E_{\lambda,t} = 0.9E_{\lambda,t-1} + \varepsilon_{\lambda,t}, \qquad \lambda_t \in \arg\min_{\lambda\in\{1,0.97,0.9,0.8,0.6\}} E_{\lambda,t}\,[1 + 4(1-\lambda)]. \tag{F2}$$

Ties favor the larger λ. The selected model supplies the equilibrium search objective. After more than four deployments, switching to a smaller λ raises the radius to at least its initial value. The sample path retains its 60-record window; the scalar path retains its full history.

F.5 Algorithm

Algorithm F1. One learning trajectory of TR-CLS.

```
Input: action range, feedback mode, deployment budget, selected
 ↪  configuration
Initialize: central action, history, initial trust-region
 ↪  radius, random stream
For each real deployment:
    If this is the first deployment: submit the central action
    Otherwise:
        Sample path: fit the loss-coefficient response to
         ↪  actions and history
        Scalar path: fit a locally weighted action-loss
         ↪  objective
        Optimize the current predicted objective within the
         ↪  trust region
        Apply the C1 underdetermined exploration rule when
         ↪  triggered
        Add Gaussian exploration, clip, and map back to deployed
         ↪  action coordinates
    Execute one deployment, advance the population, and return
     ↪  permitted feedback
    Sample path: evaluate candidate losses on the same batch to
     ↪  form coefficients
    Scalar path: record the action and observed loss
    Append the new record to the corresponding history
    Detect persistent prediction mismatch; reset history and
     ↪  radius if triggered
    If no reset occurred and predicted improvement is valid,
     ↪  adapt the next radius
Output: learning trajectory and final executed deployment rule
```

The quadratic representation in the sample path exactly expresses fixed-batch losses for several tasks and approximates losses for tasks such as classification.

APPENDIX G COMPONENT INTEGRATION

G.1 REGISTERING A NEW TASK INSTANCE

A task configuration can turn an existing backend into a new research instance. The example below constructs a four-dimensional immediate-demand task while retaining the standard action, observation, budget and evaluation interfaces.

Python

```python
from gymnasium.envs.registration import register
from ppbench.tasks import Task
from ppbench.env import PPGym

def make_pricing4(horizon=12, sample_size=32):
    task = Task(
        "Pricing-4D-example", "gaussian_demand", 4, 0., 5.,
        config={"base_mean": (5., 6., 7., 8.),
                "response_strength": 1.5, "noise_std": .5},
    )
    return PPGym(task, horizon=horizon, sample_size=
   sample_size)

register(id="PPGym/Pricing-4D-example-v0",
         entry_point=make_pricing4)
```

The example reuses the demand backend with a new configuration. A new response mechanism supplies initialization, deployment transitions and observation generation, together with rule and information requirements; the same adapter exposes it through the common interface.

G.2 Integrating a Sample-Gradient Learner

For pricing, the direct gradient of the loss with respect to the rule on the current batch is the negative mean demand. The following learner updates from public samples and executed actions.

Python

```python
import numpy as np

class SamplePriceGradient:
    requires = {"samples", "previous_action"}

    def __init__(self, dim, low, high, learning_rate=.05):
        self.theta = np.zeros(dim)
        self.low, self.high, self.lr = low, high, learning_rate

    def act(self, observation):
        return self.theta.copy()

    def observe(self, action, observation, reward):
        if observation["sample_mask"] != 1:
            raise ValueError("Sample feedback is required")
        demand = observation["samples"][:, :-1]
        gradient = -demand.mean(axis=0)
        self.theta = np.clip(
            observation["previous_action"] - self.lr * gradient,
            self.low, self.high,
        )
```

The `requires` declaration allows users to inspect feedback requirements; this example explicitly checks the sample mask during updates. Registered learners in the generic backend also undergo pre-run checks against the public capability table. This gradient defines an update for empirical loss on the current batch. The total derivative of performative risk additionally accounts for the distribution response.

G.3 Running Tasks and Recording Metrics

The following example runs SPSA on three-product pricing with 120 deployments and 128 samples per deployment. Replacing the task and compatible learner preserves the interaction loop. Reward is negative observed loss.

Python

```python
from ppbench.env import PPGym
from ppbench.tasks_v2 import by_name
from ppbench.agents_v2 import make_agent

task = by_name("Pricing-3D")
env = PPGym(task, horizon=120, sample_size=128)
learner = make_agent("spsa", task.public(), seed=7, cfg={"lr":0.02, "delta":0.1})
observation, _ = env.reset(seed=7)
for _ in range(env.horizon):
    action = learner.act(observation)
    observation, reward, _, _, _ = env.step(action)
    learner.observe(action, observation, reward)
env.close()
```

The custom task and learner in Sections G.1 and G.2 use the same loop; the runner additionally records evaluation metrics:

Python

```python
import gymnasium as gym

env = gym.make("PPGym/Pricing-4D-example-v0")
agent = SamplePriceGradient(4, 0., 5.)
observation, _ = env.reset(seed=0)
rows = []
for deployment in range(1, 13):
    action = agent.act(observation)
    observation, reward, terminated, truncated, _ = env.step(action)
    agent.observe(action, observation, reward)
    metrics = env.unwrapped.evaluation_metrics()
    rows.append({"deployment": deployment,
                 "observed_loss": -reward, **metrics})
env.close()
```

The runner records observed loss, population risk, stationary risk and sample counts after each deployment. Evaluation metrics remain separate from the observations passed to the learner.

Appendix H Experimental Protocol and Resource Accounting

H.1 Data Splits and Statistical Units

Development uses seeds 20 to 23; main testing uses 340 to 359; controlled sweeps use 520 to 529; fixed-configuration ablations use 600 to 619. Within each task, configurations are selected by mean online observed loss across development seeds and then used for all main-test seeds. Comparators are also selected on the development set; test results do not enter selection.

Methods share environment initialization and random-stream starting points. Their deployed actions differ, so subsequent distributions evolve along their own trajectories. Paired statistics subtract the two results obtained under the same seed; different seeds provide independent replicates. Sweeps retain configurations selected on the original clean tasks, without retuning for each perturbation level.

Table H1. Experimental scope. A run is a complete trajectory for one task, algorithm configuration and random seed. A frozen evaluation corresponds to the final rule from one main-test trajectory.

Experiment	Configuration rule	Independent replicates	Runs
Development	Candidate grids in Appendix B	Four seeds per candidate	4,544
Main test	Selected configurations for 297 compatible combinations	20 seeds per combination	5,940
Controlled sweeps	Configuration selected on the original clean task	10 seeds per condition	15,640
Frozen deployment	Fixed final action from each main-test trajectory	Corresponding 20 test seeds	5,940
Single-component ablations	Fixed full-method configuration	20 seeds per condition	500

Controlled sweeps comprise 4,000 action-noise runs, 2,760 observation-noise runs, 3,200 response-change runs, 2,080 external-shock runs, 750 response-strength runs, 700 adaptation-rate runs and 2,150 sample-size runs. Table E1 specifies their task coverage and levels.

Each compatible task–method pair contributes 20 test trajectories. Observation-noise comparisons use learners that consume the perturbed feedback. Appendix A.6 specifies the analytical risk references; losses are compared within tasks.

Table H2. Shared evaluation settings. Each deployment samples 128 observations, except in sample-size sweeps.

Evaluation	Tasks and configurations	Budget and repetitions	Evaluation target
Base comparison	13 tasks, 297 combinations	120 deployments, 20 seeds	Online loss, population risk, late-stage risk
Controlled variation	Response, adaptation, noise or sample-size sweeps	120 deployments, 10 seeds per condition	Perturbed risk, post-change risk
Continued deployment	Fix the final rule and retain population state	40 additional deployments, the original 20 seeds	Subsequent risk of the final rule

H.2 Metrics, Budgets and Computation

The main learning stage uses $T = 120$ and $n_t = 128$ at every deployment. Late-stage risk is $J_T^{\text{late}} = \frac{1}{20}\sum_{t=T-19}^{T} r_t$, the mean population risk over the final 20 deployments; frozen evaluation uses $H = 40$. Stateless main-test trajectories draw 15,360 observations. Stateful trajectories include one initialization batch, for a total of 15,488. Each query in a two-point method consumes one deployment. Evaluating candidate losses on a fixed batch adds neither deployments nor samples.

Supplementary interface checks and fixed-configuration ablations run on CPUs with Python 3.11.14, NumPy 2.4.4 and Gymnasium 1.3.0. They use four worker processes, each limited to one numerical-library thread. Tasks, configurations and seeds can run independently; statistical analyses use completed trajectories.