Two-Timescale Adaptive Memory for CTR Prediction under Temporal Distribution Shift

Abstract

Click-through rate (CTR) prediction is important in online advertising and recommendation systems. The CTR prediction can be challenging in practice, especially under temporal distribution shift. Existing methods often treat historical adaptation over model training and within-day information online updates separately, limiting their ability to exploit both heterogeneous historical data and newly available feedback. We propose Two-Timescale Adaptive Memory (TTAM), a causal framework that combines context-dependent historical weighting with adaptive-persistence online calibration. At the slower timescale, TTAM learns which historical horizons to use and how quickly their weights should change. At the faster timescale, it adapts calibration memory using delayed within-day feedback. We theoretically establish stability properties and regret guarantees for calibration aggregation. Experiments show that TTAM improves robustness across 14 controlled drift regimes and increases downstream bidding value. On the full Criteo and Avazu datasets, TTAM achieves strong performance against competing baselines under rolling-origin evaluation, supporting the value of adaptive memory at both timescales.

1 Introduction

Click-through rate (CTR) prediction is a central task in online advertising and recommendation systems. It estimates the probability that a user will click on a displayed advertisement or recommended item. These estimates help platforms select relevant content and help advertisers decide how much to *bid* for an advertising opportunity. Accurate predictions therefore support both user experience and the efficient allocation of advertising budgets. Maintaining reliable CTR predictions is consequently an important practical objective.

The CTR prediction can be challenging in practice. One major factor is *temporal distribution shift*—user interests evolve, advertising campaigns change, and the composition of traffic varies across days and hours—making the choice of training history involved. Using a long history provides more observations but may retain outdated patterns, while recent data allows faster adaptation but can discard useful information and produce less reliable estimates. Moreover, these changes need not affect all users or advertisements universally: a recent shift may make older observations less relevant for one group while leaving them useful for another. Moreover, changes can also occur between scheduled model updates (e.g., daily model retraining), creating a need to use newly observed responses without waiting for another full training cycle. These practical difficulties motivate methods that balance responsiveness with the retention of useful knowledge [17].

Existing approaches address several aspects of this problem. Adaptive rolling-window methods use past prediction errors to determine how much history is relevant, while expert-based methods combine multiple models that respond differently to changing data [10]. Another type of algorithms adjusts predictions as the latest outcomes become available by adapting online learning approaches [9]. These approaches motivate different forms of temporal adaptation, but each can fail under certain circumstances. A single historical weighting rule can overlook differences across users and advertisements and learn only an average pattern. Likewise, historical model selection alone does not use feedback arriving between retraining cycles, which could provide additional signals. On the other hand, continually adapting to new information could accelerate the forgetting of older patterns that might still be useful. These tradeoffs motivate a more integrated treatment of temporal adaptation.

In this paper, we propose Two-Timescale Adaptive Memory (TTAM), a causal framework that addresses these decisions together. At the slower timescale, TTAM performs a full training cycle and combines predictors trained on different lengths of history. Their relative weights depend on the features of each sample, allowing different users and advertisements to draw on different historical information. The method also controls how quickly these weights should change, balancing responsiveness to new conditions with stability over time. At the faster timescale, TTAM uses newly available click outcomes to adjust predictions between consecutive training cycles. It adapts how much fresh information is used, rather than committing to a single memory duration. Both adaptations use feedback only

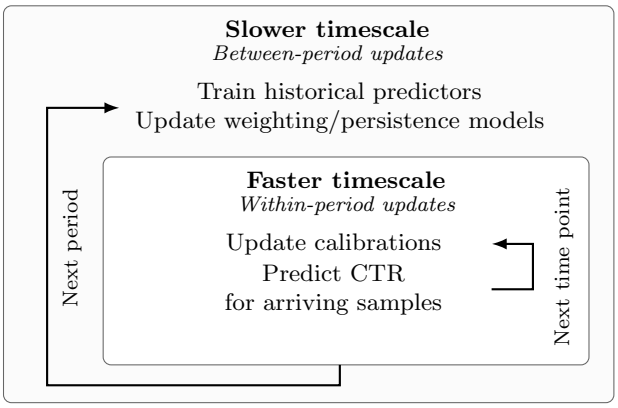

Figure 1: TTAM follows two nested update loops. Historical models and weighting mechanisms are updated between periods, while calibration and prediction proceed within each period. All updates use only available feedback.

after it becomes available. The central improvement is therefore to make temporal memory adaptive at both levels: the history supporting the prediction and the recent feedback used to refine it. Fig. 1 illustrates the pipeline of the TTAM algorithm.

Contributions. Our contributions are threefold.

1. We introduce a causal two-timescale adaptive-memory framework. Its slower-timescale adaptation uses context-dependent weights over multiple training horizons and a learned persistence parameter to maintain continuity in these weights over time. Its faster-timescale adaptation combines reset, short-memory, and long-memory Platt calibrators using online feedback. The two layers learn both *which* historical information remains useful and *how long* an adaptation state should persist.

2. We provide theoretical guarantees for the framework. We establish regret bounds for the calibration policy against a switching sequence of memory experts. These results connect TTAM to classical expert-tracking theory and extend the comparison beyond a single fixed calibration map to changing calibration memories under delayed feedback.

3. We validate the two adaptation layers through controlled experiments and evaluations on two public CTR datasets, Criteo and Avazu. Rolling-origin evaluation demonstrates consistent improvements in robustness and prediction accuracy from TTAM's adaptive-memory design. The prediction gains under recurring and local drift translate into greater bidding value under an unchanged autobidding policy.

2 Related Work

Learning under temporal distribution shift. Learning under temporal distribution shift concerns prediction when the data distribution changes over time. Such changes may affect the distribution of input features, the relationship between features and outcomes, or both. As a result, patterns learned from earlier observations may become less useful for future prediction. This problem is closely related to concept drift in data streams, for which Gama et al. [6], Lu et al. [18] review methods for detecting changes and adapting predictive models.

Existing methods differ in how they incorporate new observations while retaining useful historical information. Detection-based methods monitor prediction errors and trigger model updates or replacement when there is evidence of a change [5]. Window-based methods restrict learning to recent observations and adjust the window length to balance responsiveness with the amount of available data [3]. Instead of discarding older observations, forgetting methods gradually reduce their influence through temporal weights. For example, Bennett and Clarkson [2] learn a weighting mechanism from predictive performance, allowing the effective training history to adapt over time. Historical information can also guide the choice among candidate predictors. Han et al. [10] adapt the amount of recent history used to estimate current prediction risk and select a model. Ensemble methods instead combine several predictors and adjust their influence according to observed performance. They may also introduce new predictors or remove poorly performing ones as the stream evolves [16].

Mixture-of-experts and adaptive aggregation. Mixture-of-experts (MoE) models combine several predictors through a learned gating model. The gate assigns weights according to the input, allowing different samples to use different combinations of experts. Jacobs et al. [13] develop this approach by jointly learning the experts and their gating weights, encouraging specialization across different parts of the data. Hierarchical architectures extend this idea through multiple levels of gating, allowing the input space to be divided into progressively finer regions [14].

In a changing environment, the usefulness of an expert may also vary over time. For CTR prediction, Liu et al. [17] propose AdaMoE, which uses an adaptive weighting policy to balance responsiveness to new patterns with retention of previously learned information. A related literature on online expert aggregation studies how to update combination weights from sequential prediction errors. Exponential weighting gives greater influence to experts with smaller cumulative losses. To

accommodate changes in which expert performs best, Herbster and Warmuth [12] introduce Fixed Share, which redistributes part of the weight among experts and provides guarantees against switching sequences of experts. The sharing rate controls how readily the learner can shift between experts. Rather than choosing this rate in advance, Learn-α combines Fixed Share learners with different switching rates and updates their relative weights using observed losses [19].

Online calibration. Probability calibration concerns agreement between predicted probabilities and observed event frequencies among samples receiving similar predictions. Post-hoc calibration methods seek this agreement by adjusting the outputs of a trained model. Platt scaling fits a sigmoid transformation of model scores using labeled calibration data [20]. A simpler alternative is temperature scaling, which rescales logits using a single parameter. Guo et al. [7] demonstrate its effectiveness for calibrating neural network predictions.

In a changing environment, a correction fitted to historical data may become outdated, motivating calibration methods that update as new outcomes are observed. For adjusting the predictions of a trained model, Gupta and Ramdas [9] introduce OPS, which updates the slope and intercept through online logistic regression using log loss. They also combine OPS with an additional calibration procedure to obtain guarantees for adversarial outcome sequences. Online optimization provides tools for these sequential updates. In particular, Online Newton Step achieves logarithmic regret relative to the best fixed parameters for exp-concave losses under standard boundedness assumptions [11]. When outcomes are delayed, the learner must make further predictions before receiving feedback on earlier ones. Joulani et al. [15] provide general reductions for adapting online learning algorithms to this setting and characterize how delays affect regret.

3 Preliminaries

Chronological CTR stream. The time horizon is divided into deployment periods $d = 1, \dots, D$. In CTR prediction, a period is usually a day or a fixed temporal block. The prediction models are trained between consecutive periods using only the data available at the time of training. Within each period, observations occur at a finite set of equally spaced time points $t = 1, \dots, T$, which may represent hours within a day.

The samples in period d are denoted by

$$\left\{ \left\{ (\boldsymbol{x}^{d,t,i}, y^{d,t,i}) \right\}_{i=1}^{n_{d,t}} \right\}_{t=1}^{T}, \tag{1}$$

where $\boldsymbol{x}^{d,t,i}$ contains the context of a sample, such as the advertiser and advertisement category. The click label satisfies $y^{d,t,i} \in \{0, 1\}$, with $y^{d,t,i} = 1$ if the advertisement is clicked and 0 otherwise. At time point t of period d, the contexts of all $n_{d,t}$ arriving samples $\{\boldsymbol{x}^{d,t,i}\}_{i=1}^{n_{d,t}}$ are observed before predictions are made.

Each label becomes available after a uniform delay $\delta \geq 0$, measured in time steps. At time t of period d, the learner has access to labels from earlier periods whose feedback delays have elapsed, together with labels from time points

$$1 \leq t' \leq \min\{t - 1, t - \delta\} \tag{2}$$

in the current period. The restriction $t' < t$ excludes outcomes from the current time point even when $\delta = 0$.

Mixture of experts. A mixture-of-experts (MoE) model combines the outputs of K predictive models, referred to as experts. For an input $\boldsymbol{x}$, let $\boldsymbol{p}(\boldsymbol{x}) = (p_1(\boldsymbol{x}), \dots, p_K(\boldsymbol{x}))^\top$ denote the vector of expert predictions. A gating model W_ϕ assigns a weight vector

$$\mathbf{w}(\boldsymbol{x}) = \mathrm{softmax}(W_\phi(\boldsymbol{x})) \in \Delta_K, \tag{3}$$

where Δ_K is the probability simplex in $\mathbb{R}^K$. The mixture prediction is

$$q(\boldsymbol{x}) = \sum_{k=1}^{K} w_k(\boldsymbol{x}) p_k(\boldsymbol{x}) = \mathbf{w}(\boldsymbol{x})^\top \boldsymbol{p}(\boldsymbol{x}). \tag{4}$$

The gating model is trained to assign larger weights to experts that perform well for the current input. Because the weights depend on the input, different samples may use different combinations of experts.

Online Platt scaling. Online Platt scaling (OPS) adjusts predicted logits using feedback received during deployment [8]. For a base prediction probability $q \in (0, 1)$, the calibrated prediction is

$$\widehat{p} = \mathrm{sigmoid}(a \, \mathrm{logit}(q) + b), \tag{5}$$

where $\mathrm{logit}(q) = \log(\frac{q}{1-q})$ and $\mathrm{sigmoid}(z) = (1 + \exp(-z))^{-1}$. The slope a adjusts the scale of the prediction logits, while the intercept b shifts them. The choice $(a, b) = (1, 0)$ leaves the base prediction unchanged.

OPS adjusts the slope and intercept through online learning updates based on newly available labels. For example, a projected gradient update takes the form

$$\boldsymbol{\theta}_{r+1} = \Pi_\Theta(\boldsymbol{\theta}_r - \eta_r \nabla L_r(\boldsymbol{\theta}_r)), \tag{6}$$

where $\boldsymbol{\theta}_r = (a_r, b_r)^\top$, η_r is the learning rate, and Π_Θ denotes projection onto the parameter domain Θ. Here, L_r is the average log loss on the data available for update r. Other online learning methods, such as Online Newton Step, can also be used.

4 Two-Timescale Adaptive Memory

Adaptive historical memory. Our first algorithmic block adapts the mixture-of-experts framework to temporal distribution shift by treating models trained on different historical horizons as experts. A gating model assigns sample-specific weights to these experts. Learned temporal persistence blends these weights with a memory of previous expert allocations, balancing responsiveness and stability.

TTAM maintains $K = 5$ experts with historical horizons $\mathcal{H} = \{1, 3, 7, 14, \text{expanding}\}$. Before each period d, expert k is trained using available data from its corresponding historical horizon. For sample (d, t, i), the expert predictions are collected in $\boldsymbol{p}^{d,t,i} = (p_1^{d,t,i}, \ldots, p_K^{d,t,i})^\top$, where $p_k^{d,t,i} = f_k^d(\boldsymbol{x}^{d,t,i})$.

Before period $d \geq 2$, we update the gating parameters ϕ and persistence parameters ψ using available feedback from the preceding period. These updates are based on the regularized log loss of the historical mixture. We also update the memory using the weights deployed during period $d-1$:

$$\overline{\boldsymbol{w}}^{d-1} = \frac{\sum_{t=1}^{T} \sum_{i=1}^{n_{d-1,t}} \boldsymbol{w}^{d-1,t,i}}{\sum_{t=1}^{T} n_{d-1,t}}, \tag{7}$$

$$\boldsymbol{m}^{d-1} = (1-\rho)\boldsymbol{m}^{d-2} + \rho\, \overline{\boldsymbol{w}}^{d-1}, \tag{8}$$

where $\rho \in [0, 1]$ controls the rate of memory updates. The first period uses initialized model parameters and $\boldsymbol{m}^0 = \mathbf{1}/K$.

Let $\boldsymbol{s}^{d-1}$ summarize the information available before period d, including recent expert losses, disagreement among experts, empirical CTR, changes in expert weights, and indicators of distribution shift. The persistence model determines

$$\beta_d = \text{sigmoid}\big(R_\psi(\boldsymbol{s}^{d-1})\big) \in [0, 1]. \tag{9}$$

The model parameters, memory, and persistence parameter remain fixed throughout period d.

As each sample arrives, the gating model produces

$$\widetilde{\boldsymbol{w}}^{d,t,i} = \text{softmax}\big(W_\phi\left(\boldsymbol{x}^{d,t,i}, \boldsymbol{p}^{d,t,i}, \boldsymbol{s}^{d-1}\right)\big) \in \Delta_K. \tag{10}$$

The deployed weights and combined prediction are

$$\boldsymbol{w}^{d,t,i} = (1-\beta_d)\widetilde{\boldsymbol{w}}^{d,t,i} + \beta_d \boldsymbol{m}^{d-1}, \tag{11}$$

$$q^{d,t,i} = \left(\boldsymbol{w}^{d,t,i}\right)^\top \boldsymbol{p}^{d,t,i}. \tag{12}$$

A larger β_d preserves the recent allocation of expert weights. A smaller β_d gives greater influence to the current sample-specific gate.

Adaptive online calibration. Our second algorithmic block adapts OPS into the long-term training process to further leverage the local information. It combines three calibrators: a reset calibrator that reproduces the classic OPS calibrator, and short-memory and long-memory calibrators that retain their states across periods.

For the historical-memory prediction $q^{d,t,i}$, let $z^{d,t,i}$ be its logit after clipping to $[\epsilon, 1-\epsilon]$, where $0 < \epsilon < 1/2$. Calibrator $k \in \{1, 2, 3\}$ produces

$$c_k^{d,t,i} = \text{sigmoid}\Big(a_k^{d,t} z^{d,t,i} + b_k^{d,t}\Big). \tag{13}$$

The reset calibrator initializes $(a_1^{d,1}, b_1^{d,1}) = (1, 0)$ for any period d and uses the classic OPS updates. The persistent calibrators use discounted Online Newton Step updates with short and long memory scales. All updates use only feedback whose delay has elapsed.

The final prediction combines the calibrators:

$$\widehat{p}^{d,t,i} = \sum_{k=1}^{3} \pi_k^{d,t} c_k^{d,t,i}, \qquad \boldsymbol{\pi}^{d,t} \in \Delta_3. \tag{14}$$

The weights are initialized at $\boldsymbol{u} = (1-\lambda, \lambda/2, \lambda/2)^\top$, where $\lambda \in [0, 1]$. For feedback batch r, let L_k^r denote the average log loss of calibrator k on its previously issued predictions. The weights are updated by

$$\widetilde{\pi}_k^{r+1} \propto \pi_k^r \exp(-\eta_m L_k^r), \tag{15}$$

$$\boldsymbol{\pi}^{r+1} = (1-\alpha)\widetilde{\boldsymbol{\pi}}^{r+1} + \alpha \boldsymbol{u}, \tag{16}$$

where $\boldsymbol{\pi}^r$ denotes the weights before update r, $\eta_m > 0$ is the learning rate, and $\alpha \in [0, 1]$ is the average parameter. The averaging toward $\boldsymbol{u}$ maintains weight on alternative memory policies, allowing them to regain influence when conditions change.

Overall algorithm. The complete procedure is in Algorithm 1.

5 Theoretical Guarantees

We first establish properties of the slower timescale updates. We then bound the cumulative prediction loss at the faster timescale relative to a sequence that can switch among the three calibrators. The results do not require a stochastic model of temporal shift.

5.1 Properties of slower timescale updates

Lemma 1 (Historical weights and predictions)**.** *Suppose $\widetilde{\boldsymbol{w}}^{d,t,i}, \boldsymbol{m}^{d-1} \in \Delta_K$, $\beta_d, \rho \in [0, 1]$, and $p_k^{d,t,i} \in [\epsilon, 1-\epsilon]$ for every expert k. Then*

$$\boldsymbol{w}^{d,t,i}, \boldsymbol{m}^d \in \Delta_K, \qquad q^{d,t,i} \in [\epsilon, 1-\epsilon]. \tag{17}$$

Algorithm 1 Two-Timescale Adaptive Memory (TTAM)

Require: Historical horizons $\mathcal{H}$; initial training data; parameters $\rho, \lambda, \alpha, \eta_m, \epsilon, \delta$; OPS and discounted ONS configurations.

Initialize ϕ, ψ, $\boldsymbol{m}^0 = \mathbf{1}/K$, $\boldsymbol{\pi} = \boldsymbol{u} = (1-\lambda, \lambda/2, \lambda/2)^\top$, the three calibrators at $(a_k, b_k) = (1, 0)$, their optimizer states, and empty feedback queues.
for $d = 1, \ldots, D$ **do**
  if $d > 1$ **then**
    Update ϕ, ψ using available feedback and the regularized historical-mixture loss.
    Update $\boldsymbol{m}^{d-1}$ using Eq. (8).
  end if
  Train each expert f_k^d on its historical horizon using available data from periods before d.
  Construct $\boldsymbol{s}^{d-1}$ and compute β_d using Eq. (9). $\boldsymbol{\pi}$, and their pending feedback queues.
  for $t = 1, \ldots, T$ **do**
    Update the calibrator parameters $(a_k^{d,t}, b_k^{d,t})$ and weights $\boldsymbol{\pi}^{d,t}$
    Observe the arriving contexts $\{\boldsymbol{x}^{d,t,i}\}_{i=1}^{n_{d,t}}$.
    for $i = 1, \ldots, n_{d,t}$ **do**
      Compute expert predictions $p_k^{d,t,i} = f_k^d(\boldsymbol{x}^{d,t,i})$.
      Compute $\widetilde{\boldsymbol{w}}^{d,t,i}$ and $\boldsymbol{w}^{d,t,i}$ using Eqs. (10) and (11).
      Compute $q^{d,t,i} = (\boldsymbol{w}^{d,t,i})^\top \boldsymbol{p}^{d,t,i}$ and its clipped logit $z^{d,t,i}$.
      Compute $c_k^{d,t,i}$ using the current calibrator parameters.
      Output $\widehat{p}^{d,t,i} = \sum_{k=1}^3 \pi_k^{d,t} c_k^{d,t,i}$.
    end for
  end for
end for

Moreover, for either click outcome $y \in \{0, 1\}$,

$$\ell(y, q^{d,t,i}) \le \sum_{k=1}^{K} w_k^{d,t,i} \ell(y, p_k^{d,t,i}). \tag{18}$$

Proof. The deployed weights and the updated memory are convex combinations of probability vectors. The combined prediction is therefore a convex combination of the expert predictions. The loss bound follows from convexity of log loss in the predicted probability. □

Thus, the historical weights remain nonnegative and sum to one. The combined prediction has log loss no larger than the weighted average of the historical experts' losses.

Lemma 2 (Effect of persistence)**.** *For every sample (d, t, i),*

$$\left\|\boldsymbol{w}^{d,t,i} - \boldsymbol{m}^{d-1}\right\|_1 = (1-\beta_d) \left\|\widetilde{\boldsymbol{w}}^{d,t,i} - \boldsymbol{m}^{d-1}\right\|_1. \tag{19}$$

The memory update satisfies

$$\left\|\boldsymbol{m}^d - \boldsymbol{m}^{d-1}\right\|_1 \le 2\rho. \tag{20}$$

Proof. Subtracting $\boldsymbol{m}^{d-1}$ from the deployed weight vector gives the first identity. For the second bound, the memory update gives

$$\boldsymbol{m}^d - \boldsymbol{m}^{d-1} = \rho\left(\overline{\boldsymbol{w}}^d - \boldsymbol{m}^{d-1}\right). \tag{21}$$

Both vectors on the right belong to Δ_K, so their ℓ_1 distance is at most two. □

A larger β_d keeps the sample-specific weights closer to the stored memory. The parameter ρ separately limits how much that memory changes between consecutive periods.

5.2 Regret bound for faster timescale updates

Recall that we use $c_k^{d,t,i}$ to denote the k-th type of calibrated logit for the sample (d, t, i), and use $\widehat{p}^{d,t,i}$ to denote the combined calibrated logit. For each time point (d, t), define the average log losses

$$L_k^{d,t} = \frac{1}{n_{d,t}} \sum_{i=1}^{n_{d,t}} \ell\Big(y^{d,t,i}, c_k^{d,t,i}\Big), \tag{22}$$

$$\widehat{L}^{d,t} = \frac{1}{n_{d,t}} \sum_{i=1}^{n_{d,t}} \ell\big(y^{d,t,i}, \widehat{p}^{d,t,i}\big). \tag{23}$$

Let $k^{d,t} \in \{1, 2, 3\}$ specify a calibrator for the time point (d, t). The comparison may switch between the reset, short-memory, and long-memory calibrators. The regret against the strategy that uses a single calibrator per time point is defined as

$$\mathcal{R} = \sum_{d=1}^{D} \sum_{t=1}^{T} \left(\widehat{L}^{d,t} - L_{k^{d,t}}^{d,t}\right). \tag{24}$$

Theorem 1 (Regret of adaptive online calibration)**.** *Let $\lambda \in (0, 1)$, $\alpha \in (0, 1)$, and $\eta_m > 0$. Initialize the calibration weights at $\boldsymbol{u} = (1-\lambda, \lambda/2, \lambda/2)^\top$, and define $u_{\min} = \min\{1-\lambda, \frac{\lambda}{2}\}$. Assume $0 \le L_k^{d,t} \le L_{\max}$. Each time point's loss vector is used once, in chronological order, after its feedback becomes available. At any prediction time, at most $\lceil \delta \rceil$ earlier loss vectors remain unavailable.*

For any comparison sequence that switches calibrators at most S times, where $0 \le S \le DT - 1$, set

$$C_S = (S+1)\log\frac{1}{u_{\min}} + S\log\frac{1}{\alpha} + (DT - S)\log\frac{1}{1-\alpha}.$$

Then

$$\mathcal{R} \le \frac{C_S}{\eta_m} + \frac{\eta_m L_{\max}^2 N(1 + 2\lceil\delta\rceil)}{8} + \alpha L_{\max} N \lceil\delta\rceil. \tag{25}$$

Proof. First consider the weights obtained by processing each loss vector before the next prediction. The fixed-share update assigns probability at least

$$u_{\min}^{S+1}\alpha^S(1-\alpha)^{N-1-S} \tag{26}$$

to every comparison sequence with at most S switches. The exponential-weights argument and Hoeffding's lemma therefore bound its regret by $C_S/\eta_m + \eta_m L_{\max}^2 N/8$.

For delayed feedback, compare the deployed weights with those obtained after processing all earlier loss vectors. One exponential update changes the weights by at most $\eta_m L_{\max}/2$ in ℓ_1 distance. This follows from Hoeffding's lemma and Pinsker's inequality. The subsequent sharing step adds at most 2α to this change.

The two weight vectors differ by at most $\lceil\delta\rceil$ updates. Since losses lie in $[0, L_{\max}]$, the resulting increase in weighted loss at each time point is at most

$$\lceil\delta\rceil\left(\frac{\eta_m L_{\max}^2}{4} + \alpha L_{\max}\right). \tag{27}$$

Summing over the N time points gives the stated bound. Finally, convexity of log loss gives $\widehat{L}^{d,t} \le \sum_{k=1}^{3}\pi_k^{d,t}L_k^{d,t}$, so the same bound applies to the issued predictions. □

The theorem compares TTAM with a calibrator sequence chosen in hindsight. It therefore allows different calibration memories to be useful at different times. The comparison uses the same historical predictions and the same three calibrators as TTAM.

For a fixed feedback delay and fixed $u_{\min} > 0$, choose $\alpha = (S+1)/(N+1)$ and

$$\eta_m = \sqrt{\frac{8C_S}{L_{\max}^2 N(1+2\lceil\delta\rceil)}}. \tag{28}$$

If $S = o(N)$, the regret bound is $o(N)$. Thus, the average excess loss vanishes when the number of calibrator changes grows more slowly than the number of prediction time points.

6 Numerical experiments

In this section, we evaluate the numerical performance of TTAM on chronological CTR prediction, isolate the contribution of each timescale, and examine the resulting bidding performance. The main comparison uses a common evaluation protocol: nested rolling-origin for all methods.

6.1 Experimental setup

Datasets. We use the full Criteo Attribution dataset [4], containing $16,468,027$ observations across 31 days, and the full Avazu dataset [1], containing $40,428,967$ observations across 10 days. Observations are ordered chronologically, without random splitting across days. In our experiments, we did not explicitly include identifiers and timestamps as parts of the prediction features.

Model selections. For each historical window, we train an ℓ_2-regularized logistic regression model, with high-cardinality categorical variables represented using feature hashing. For all historical predictors, the logistic-regression regularization parameter is fixed at 10^{-4}. All methods share the same features, model class, and historical predictors whenever the corresponding histories are used. The set of training horizons is $\mathcal{H} = \{1, 3, 7, 14, \text{expanding}\}$ days. Each finite window is truncated to the available past when fewer days are available. Some truncated windows therefore coincide, particularly on Avazu; coincident histories share the same fitted predictor across methods.

For the slower-timescale adaptation, the context-dependent gating model is a linear softmax model over the five historical experts, and the persistence model is a linear map from the period-level state to a scalar logit followed by a sigmoid. Both model architectures are fixed across datasets and evaluation origins, while their parameters are updated causally. We fix the learning rate, regularization, and network architecture in advance, and select the memory update rate $\rho \in \{0.2, 0.3, 0.5\}$ separately at each rolling origin using the inner-validation log loss.

Chronological training, validation, and evaluation. Using zero-based day indices, we evaluate Criteo on days 16–30 and Avazu on days 5–9, giving 15 and 5 rolling origins, respectively. For an evaluation day d, up to three of the most recent eligible earlier days form the inner validation window. The first Avazu origin has two eligible validation days. Historical expert predictions for each validation day are generated using only the information available before that day, and online states are replayed with the prescribed feedback delay. Earlier days provide the training and initialization history. All data-tuned settings are selected separately at each origin using only labels available before day d; settings inherited from tuning on overlapping evaluation days are not reused.

Selection is staged rather than jointly optimized over all settings. We first select historical-module settings using uncalibrated validation log loss. On each selected historical prediction stream, we next select the daily-reset OPS settings, including the reset anchor used by TTAM's fast adaptation. We then select the remaining fast-adaptation settings, including its meta learning rate, while holding that anchor fixed. Each selection

uses impression-weighted validation log loss averaged across seeds 0, 1, and 2, yielding one configuration shared across the three seeds. The selected configuration is replayed over the preceding stream to initialize its state, and its hyperparameters remain fixed during the evaluation day. The same three seeds are used for all methods.

Click labels become available after a 30-minute delay. This availability restriction applies to every label-dependent operation, including historical predictor fitting, validation objectives, TTAM's slow-adaptation state summaries, ARW and AdaMoE loss histories, and calibration updates. In particular, labels that have not arrived by a daily fitting or selection cutoff are excluded until they become available. Calibration updates occur every 15 minutes on Criteo and every hour on Avazu, using only labels available at the update time. Within-day adaptation may therefore affect subsequent predictions, but never an already issued prediction. Persistent states are carried across day boundaries according to the method being evaluated.

Evaluation measures and uncertainty. The primary measure is log loss, first averaged over impressions within each evaluation day, then over seeds within that day, and finally equally over evaluation days. Let $L_{d,s}^{M}$ denote the impression-averaged log loss of method M on day d with seed s, and let $\mathcal{D}$ be the set of evaluation days. We report

$$L_d^M = \frac{1}{3}\sum_{s=0}^{2} L_{d,s}^M, \qquad \overline{L}^M = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} L_d^M.$$

For a comparator M, the paired gain of TTAM is

$$\Delta_d^M = L_d^M - L_d^{\mathrm{TTAM}}, \qquad \overline{\Delta}^M = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} \Delta_d^M.$$

Positive gains indicate lower log loss for TTAM. We form 95% confidence intervals for mean paired gains using 10,000 paired day-bootstrap resamples, after averaging seeds within each day. An origin is won when $\Delta_d^M > 0$. Neither impressions nor seed-by-day observations are treated as independent temporal replicates. The appendix reports seed variability, a moving-block bootstrap with block length two, impression-weighted aggregate loss, Brier score, and calibration error with a fixed binning rule. Because Avazu has only five evaluation origins, its intervals are descriptive and are interpreted together with the individual daily gains, rather than as evidence of statistical significance.

Compared methods. We compare six methods: Expanding, which trains on all available history; Best Fixed Window, which selects a horizon from $\mathcal{H}$ by past validation and holds that choice fixed during the evaluation day; ARW, which causally selects an adaptive rolling window; AdaMoE, which adaptively combines historical predictors; OPS, which applies daily-reset online Platt scaling to a historical mixture selected on past validation; and full TTAM, which combines its slow historical adaptation with its fast calibration adaptation. OPS learns both the slope a and intercept b online. Its input mixture uses constant simplex weights fitted to past validation predictions, with the objective averaged across the three seeds. These mixture weights are frozen within each evaluation day. This fitted mixture is specific to the OPS comparator; the ablation study instead uses Expanding as its control without TTAM's slow adaptation. All methods use the same chronological splits, feedback delay, and available training history. Candidate configurations, fixed constants, staged selection rules, and deterministic tie-breaking are documented in the appendix.

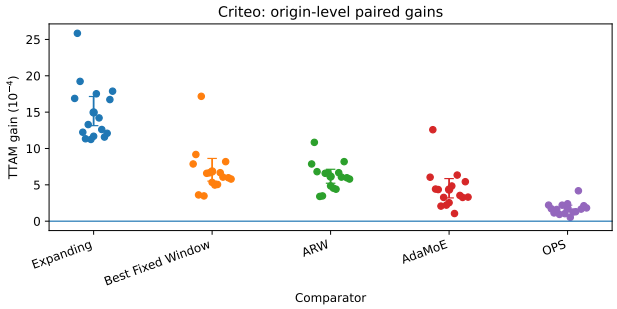

Figure 2: Origin-level paired log-loss gains of TTAM over each comparator on Criteo. Gains are reported in units of 10^{-4}. Each point corresponds to one of the 15 evaluation origins. The larger marker and error bar show the mean gain and its 95Positive values favor TTAM.

6.2 Prediction performance

Table 1 summarizes mean log loss, paired gains, confidence intervals, and origins won. Figure 2 and Figure 3 show the origin-level paired gains of TTAM over each comparator on Criteo and Avazu, respectively. Each point represents one evaluation origin, while the marker and error bar show the mean gain and its 95Positive values indicate lower log loss for TTAM.

TTAM achieves the lowest mean log loss among the six methods on both datasets. On Criteo, its mean loss is 0.608029, compared with 0.608203 for OPS, the strongest baseline. The paired improvement over OPS is 1.74×10^{-4}, with a 95% interval of $[1.36, 2.20]\times10^{-4}$. TTAM improves over every comparator on all 15 Criteo origins. The gains over AdaMoE and Expanding are larger, at 4.35×10^{-4} and 1.496×10^{-3}, respectively. All five Criteo intervals remain positive under the block-length-two sensitivity analysis reported in the

Dataset	Method	Mean log loss	TTAM gain	95% CI for gain	Origins won
Criteo	Expanding	0.609525	+14.96	[13.14, 17.15]	15/15
	Best Fixed Window	0.608713	+6.84	[5.51, 8.63]	15/15
	ARW	0.608643	+6.14	[5.22, 7.14]	15/15
	AdaMoE	0.608464	+4.35	[3.20, 5.86]	15/15
	OPS	0.608203	+1.74	[1.36, 2.20]	15/15
	TTAM	**0.608029**	—	—	—
Avazu	Expanding	0.402094	+11.57	[9.23, 14.00]	5/5
	Best Fixed Window	0.402213	+12.77	[2.02, 26.79]	4/5
	ARW	0.402359	+14.23	[2.84, 26.90]	4/5
	AdaMoE	0.401538	+6.01	[3.42, 8.94]	5/5
	OPS	0.401147	+2.10	[1.35, 2.79]	5/5
	TTAM	**0.400937**	—	—	—

Table 1: Prediction performance under the common rolling-origin protocol. Loss is averaged over seeds within each day and then equally over days. TTAM gain is the row method's loss minus TTAM's loss; positive values favor TTAM. Gains and confidence-interval endpoints are reported in units of 10^{-4} and are computed before rounding the mean losses. Intervals use paired resampling of days. Origins won count strict positive gains out of 15 Criteo or 5 Avazu origins. Avazu intervals are descriptive because of the limited number of origins.

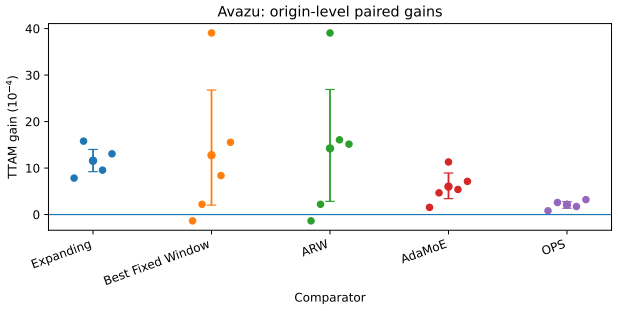

Figure 3: Origin-level paired log-loss gains of TTAM over each comparator on Avazu. Gains are reported in units of 10^{-4}. Each point corresponds to one of the five evaluation origins. The larger marker and error bar show the mean gain and its 95Positive values favor TTAM. Because only five origins are available, the confidence intervals are descriptive.

appendix.

On Avazu, TTAM attains a mean loss of 0.400937, compared with 0.401147 for OPS. Its mean improvement over OPS is 2.10×10^{-4}, with a descriptive interval of $[1.35, 2.79]\times10^{-4}$. TTAM improves over Expanding, AdaMoE, and OPS on all five origins, and over Best Fixed Window and ARW on four of five origins each. Thus, TTAM has a consistent but small advantage over the strongest baseline, rather than uniformly large improvements over all alternatives. The five-origin Avazu evaluation provides supporting directional evidence but substantially less temporal coverage than Criteo.

These results are retrospective comparisons on benchmark days that were also inspected during method development. The nested replay restricts fitting, selection, and updates to information available at each origin, but it does not provide prospective confirmation on a new period untouched during method development.

6.3 Ablation study

We isolate the two timescales through a 2×2 design: historical prediction is produced either by Expanding or by TTAM's slow adaptation, while calibration either omits or includes TTAM's fast adaptation. The four variants are Expanding without calibration, the slow adaptation without calibration, Expanding with the fast adaptation, and full TTAM. The Expanding control uses a single expanding-history predictor, not the validation-fitted multi-horizon mixture used by OPS. All variants share the same features, backbone, available histories, seeds, and evaluation origins. Expanding and Expanding with the fast adaptation receive identical historical predictions. Likewise, the slow-only variant and full TTAM share the same historical adaptation, selected using uncalibrated validation loss. Fast-adaptation settings are selected separately on each historical prediction stream using the same staged tuning rule. Table 2 reports mean loss and the paired gain of full TTAM over each variant.

Both timescales contribute relative to the Expanding control. Without calibration, replacing Expanding with the slow adaptation reduces mean loss by 1.07×10^{-3} on Criteo and 5.80×10^{-4} on Avazu. Keeping Expanding as the historical predictor, adding the fast adaptation reduces loss by 6.92×10^{-4} and 3.52×10^{-4}, respectively. With the fast adaptation already present, adding the slow adaptation yields a further reduction of 8.04×10^{-4} on Criteo and 8.05×10^{-4} on Avazu. With the slow adaptation already present, adding the fast adaptation reduces loss by 4.28×10^{-4} and 5.77×10^{-4}, respectively. Each of these four conditional improvements is positive

			Criteo		Avazu	
Variant	Slow	Fast	Mean loss	Gain [95% CI]	Mean loss	Gain [95% CI]
Expanding	No	No	0.609525	14.96 [13.14, 17.15]	0.402094	11.57 [9.23, 14.00]
Slow only	Yes	No	0.608457	4.28 [3.13, 5.77]	0.401514	5.77 [3.03, 8.94]
Expanding + fast	No	Yes	0.608833	8.04 [7.38, 8.79]	0.401742	8.05 [5.31, 11.39]
Full TTAM	Yes	Yes	**0.608029**	—	**0.400937**	—

Table 2: Contribution of each timescale relative to the Expanding control. Slow and Fast denote TTAM's slower- and faster-timescale adaptations, respectively. Full TTAM gain is the row variant's mean log loss minus that of full TTAM. Gains and paired day-bootstrap confidence-interval endpoints are in units of 10^{-4}. Historical predictions are identical within each pair with and without the fast adaptation. Both modules improve performance conditional on the presence of the other, on both datasets; Avazu intervals remain descriptive.

on all 15 Criteo origins and all five Avazu origins. Their paired intervals exclude zero, subject to the limited-sample interpretation for Avazu.

To distinguish individual contributions from interaction between the two timescales, we compute

$$\mathcal{I} = \left(\overline{L}^{\text{TTAM}} - \overline{L}^{\text{Slow only}}\right) - \left(\overline{L}^{\text{Expanding + fast}} - \overline{L}^{\text{Expanding}}\right). \tag{29}$$

A positive value means that the fast adaptation provides a smaller loss reduction when the slow adaptation is present; a negative value means that it provides a larger reduction. On Criteo, $\mathcal{I} = 2.64 \times 10^{-4}$, with a 95% interval of $[1.37, 4.00] \times 10^{-4}$, indicating partial overlap between the benefits of the two adaptations. On Avazu, $\mathcal{I} = -2.25 \times 10^{-4}$, with a descriptive interval of $[-3.35, -1.14] \times 10^{-4}$, suggesting reinforcing effects on the five evaluated days. Thus, both adaptations add predictive value, but their interaction is dataset-dependent rather than uniformly reinforcing. These comparisons establish the contribution of the slow adaptation relative to a single expanding-history predictor; they do not by themselves isolate its advantage over a validation-fitted multi-horizon mixture. Detailed persistence, contextual-weighting, calibration-memory, and slope controls are reported separately in the appendix.

6.4 Downstream bidding performance

We evaluate the final Criteo predictions in a common offline auction replay, changing only the prediction input. The recorded display cost c_i is used as a replay price proxy, not as an observed counterfactual clearing price. For predicted click probability q_i, a bid $b_i = \kappa q_i$ wins impression i when $b_i \geq c_i$; the replay then charges c_i and credits the logged click label. The primary daily target is 25% of the reference spend, defined as the sum of recorded costs over all eligible impressions that day. Targets reset daily and are identical across paired methods. No Avazu bidding result is reported because the dataset lacks a recorded price field and no separately specified synthetic-price experiment was run.

The main comparison is an *interpolated offline replay*. For each method, origin, and seed, we evaluate a predeclared grid of bid multipliers and interpolate clicks to the common target on that method's click-versus-spend curve. Interpolation brackets are selected using spend only, never evaluation click labels. All 1,080 targets across origins, seeds, methods, and the four evaluated budget levels lie within non-degenerate interpolation brackets; none is clamped to a curve endpoint. Spend is therefore matched by interpolation, not by asserting that individual paced runs achieve identical actual spend. Actual paced clicks and spend are recorded separately as diagnostics.

Let C_d^M denote method M's interpolated click count at the target spend on day d, averaged across the three prediction seeds. For each comparator M, we report the aggregate relative click gain

$$100\left(\frac{\sum_{d\in\mathcal{D}} C_d^{\text{TTAM}}}{\sum_{d\in\mathcal{D}} C_d^M} - 1\right).$$

Confidence intervals recompute this ratio in each of 10,000 paired day-bootstrap resamples. An origin is won when TTAM has a strictly larger interpolated click count at that origin's target spend. The appendix reports absolute click and spend results, actual paced-spend diagnostics, and additional target fractions of 10%, 50%, and 75%.

The log-loss improvements do not translate into higher click value in this replay. At the 25% target, TTAM obtains 0.024% fewer clicks than OPS, with a 95% interval of $[-0.032, -0.016]$% for its relative gain. Its losses against the other comparators range from 0.051% to 0.066%, and every reported interval lies below zero. The differences are small, but they consistently favor the comparators rather than TTAM. Consequently, these experiments support improved predictive log loss on the evaluated streams, but not improved downstream bidding value under the specified

Dataset	Comparator	TTAM click gain (%)	95% CI for gain	Origins won
Criteo	Expanding	−0.062	[−0.085, −0.039]	1/15
	Best Fixed Window	−0.051	[−0.078, −0.022]	4/15
	ARW	−0.054	[−0.075, −0.032]	2/15
	AdaMoE	−0.066	[−0.089, −0.043]	1/15
	OPS	−0.024	[−0.032, −0.016]	1/15

Table 3: Criteo downstream bidding performance at the 25% reference-spend target. Click gain is the percentage change in aggregate interpolated clicks for TTAM relative to the row comparator, after averaging seeds within each origin. Confidence-interval endpoints are also percentages and use paired origin resampling. Negative values favor the comparator. These are interpolated offline replay results using recorded costs as price proxies, not outcomes from a deployed bidding policy.

replay model and budget target.

References

[1] Avazu. Click-through rate prediction. Kaggle Competition, 2014. `https://www.kaggle.com/c/avazu-ctr-prediction`.

[2] S. Bennett and J. Clarkson. Time series prediction under distribution shift using differentiable forgetting. *arXiv preprint arXiv:2207.11486*, 2022.

[3] A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

[4] E. Diemert, J. Meynet, P. Galland, and D. Lefortier. Attribution modeling increases efficiency of bidding in display advertising. In *Proceedings of the ADKDD'17*, pages 1–6. 2017.

[5] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer, 2004.

[6] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46 (4):1–37, 2014.

[7] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

[8] C. Gupta and A. Ramdas. Online platt scaling with calibeating. In *International Conference on Machine Learning*, pages 12182–12204. PMLR, 2023.

[9] C. Gupta and A. Ramdas. Online platt scaling with calibeating. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 12182–12204. PMLR, 23–29 Jul

2023. URL https://proceedings.mlr.press/v202/gupta23c.html.

[10] E. Han, C. Huang, and K. Wang. Model assessment and selection under temporal distribution shift. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 17374–17392. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/han24b.html.

[11] E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2):169–192, 2007.

[12] M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine learning*, 32(2):151–178, 1998.

[13] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[14] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

[15] P. Joulani, A. Gyorgy, and C. Szepesvári. Online learning under delayed feedback. In *International conference on machine learning*, pages 1453–1461. PMLR, 2013.

[16] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research*, 8:2755–2790, 2007.

[17] C. Liu, Y. Li, F. Teng, X. Zhao, C. Peng, Z. Lin, J. Hu, and J. Shao. On the adaptation to concept drift for ctr prediction. *arXiv preprint arXiv:2204.05101*, 2022.

[18] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12):2346–2363, 2018.

[19] C. Monteleoni and T. Jaakkola. Online learning of non-stationary sequences. *Advances in Neural Information Processing Systems*, 16, 2003.

[20] J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.