Nested Pooling for Online Probabilistic Forecasting with Adaptive Step Sizes

Mostafa Rezaei[1]
mrezaei@escp.eu

[1]**Department of Information and Operations Management, ESCP Business School, France**

Abstract

We study online learning for Nested Pooling (NP), a tree-structured sparse additive model with parameter sharing across leaves, in nonstationary environments. We propose an IDBD-style meta-learning algorithm that adapts node-specific step sizes online and can be used with convex, possibly nonsmooth, loss functions, including pinball and Huberized-pinball losses for probabilistic forecasting. For the corresponding fixed-step online updates, we establish a regret bound under convex Lipschitz losses, implying sublinear regret relative to any fixed comparator defined on the same tree structure. We evaluate fixed and adaptive NP models on temporal and spatiotemporal probabilistic forecasting tasks for emergency-services demand spanning the COVID-19 outbreak. In the temporal setting, adaptive step sizes improve performance across most quantile levels, whereas in the spatiotemporal setting the batch-trained NP model performs best overall.

1 Introduction

Online forecasting methods must balance stability and adaptation under changing data distributions. This is particularly important in probabilistic forecasting, where distributional shifts can affect both central tendency and tail behavior. We study online learning in Nested Pooling (NP), a tree-structured sparse additive model with parameter sharing across leaves, and consider how its node coefficients can be updated sequentially under convex, possibly nonsmooth, loss functions. NP is attractive in high-resolution forecasting because it stabilizes sparse predictions through pooling while still allowing localized adjustments.

Our approach is motivated by adaptive step-size methods and, in particular, by IDBD (Sutton, 1992), which adapts a separate step size for each parameter of a linear predictor using a stochastic meta-gradient update. This line of work builds on earlier learning-rate adaptation methods (Sutton, 1981; Jacobs, 1988) and has been extended to improve stability (Mahmood et al., 2012), to temporal-difference learning (Kearney et al., 2018), and more broadly to gradient-based learning systems (Schraudolph, 1999); see Sutton (2022) for a historical overview. In contrast to the linear setting of the original IDBD formulation, NP uses a tree-structured additive representation in which predictions are formed by summing coefficients along a root-to-leaf path.

This paper makes three contributions. First, we present an IDBD-style update, meaning a meta-learning rule that adapts a separate step size for each parameter online from experience, for NP under convex, possibly nonsmooth, loss functions. Second, we establish a regret bound for the fixed-step online updates under convex Lipschitz losses, yielding sublinear regret relative to any fixed comparator on the same tree structure. Third, we evaluate fixed and adaptive NP models on 2-day-ahead temporal and spatiotemporal probabilistic forecasting tasks for emergency services demand spanning the COVID-19 outbreak, showing that adaptive step sizes are most beneficial in the temporal setting, while the batch-trained NP model performs best in the spatiotemporal setting.

2 Methodology

Figure 1a illustrates a NP model in which each node v is associated with a coefficient γ_v.[1] Similar to a decision tree, NP partitions the feature space, although the node splits are not shown in Figure 1a. The root-to-leaf chain containing data points in leaf node LRL is highlighted in red. The prediction at that leaf is

$$\hat{y}_{LRL} = \gamma_{\varnothing} + \gamma_L + \gamma_{LR} + \gamma_{LRL},$$

that is, the sum of the coefficients along the corresponding root-to-leaf path. Unlike a standard decision tree, predictions at different leaves share parameters through their common ancestors, so higher-level nodes influence more leaves than lower-level nodes. As a result, updating an intermediate-node coefficient affects multiple leaf predictions simultaneously, which can support rapid adaptation under distribution shift.

We consider a stream of observations $\{(\mathbf{x}_t, y_t)\}_{t\geq 1}$, where $\mathbf{x}_t \in \mathbb{R}^p$ is the feature vector and $y_t \in \mathbb{R}$ is the response at time $t \in \mathbb{N}_+$.[2] Let $\mathcal{X}$ denote the feature space. We assume a fixed NP tree structure with node set $\mathcal{V}$, node regions $\mathcal{X}_v \subseteq \mathcal{X}$ for $v \in \mathcal{V}$, and time-varying coefficients $\gamma_{v,t}$.

For an input $\mathbf{x}_t$, let

$$\text{path}(\mathbf{x}_t) \;:=\; \{v \in \mathcal{V} : \mathbf{x}_t \in \mathcal{X}_v\}$$

denote the unique root-to-leaf path containing $\mathbf{x}_t$, and let $n_t = |\,\text{path}(\mathbf{x}_t)|$ denote the number of active nodes, or path length, for observation t. The prediction at time t is then

$$\hat{y}_t := \sum_{v \in \text{path}(\mathbf{x}_t)} \gamma_{v,t}.$$

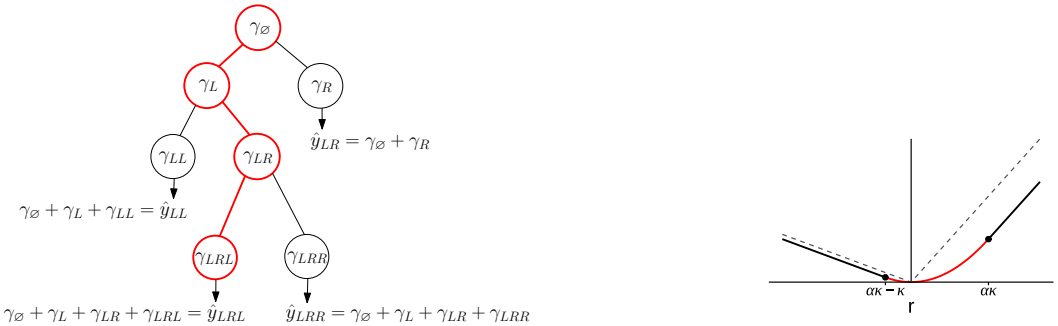

Figure 1: (a) A NP model with 7 nodes. (b) Solid line: Huberized pinball loss function with its quadratic region colored in red. Dashed line: pinball loss function.

We consider online updates under a convex, possibly nonsmooth, loss function $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$, where $\ell(y_t, \hat{y}_t)$ denotes the loss incurred at t. Of particular interest are the pinball and Huberized-pinball losses used for quantile forecasting. For residual $r = y - \hat{y}$ and quantile level $\alpha \in (0, 1)$, the pinball loss $\rho_\alpha(r)$ and Huberized-pinball loss $\rho_{\alpha,\kappa}(r)$ are defined by

$$\rho_\alpha(r) = \begin{cases} \alpha r, & r \geq 0, \\ (\alpha - 1)r, & r < 0, \end{cases} \qquad \rho_{\alpha,\kappa}(r) = \begin{cases} \alpha r - \frac{\alpha^2\kappa}{2}, & r > \alpha\kappa, \\ \frac{r^2}{2\kappa}, & \alpha\kappa - \kappa \leq r \leq \alpha\kappa, \\ (\alpha - 1)r - \frac{(\alpha-1)^2\kappa}{2}, & r < \alpha\kappa - \kappa, \end{cases}$$

[1] The splits need not be binary.

[2] When observations arrive in batches, such as demand volumes from multiple regions at the same time period, we assume a fixed ordering within each batch.

where $\kappa > 0$ is a smoothing parameter that determines the width of the quadratic region in the Huberized-pinball loss (Figure 1b, with $\alpha = 0.75$ and $\kappa = 2$). We use the following subgradient selections with respect to the forecast for the pinball and Huberized-pinball losses:

$$g_\alpha(r) = \begin{cases} -\alpha, & r > 0, \\ 0, & r = 0, \\ 1-\alpha, & r < 0, \end{cases} \qquad g_{\alpha,\kappa}(r) = \begin{cases} -\alpha, & r > \alpha\kappa, \\ -\frac{r}{\kappa}, & \alpha\kappa - \kappa \le r \le \alpha\kappa, \\ 1-\alpha, & r < \alpha\kappa - \kappa. \end{cases} \tag{1}$$

2.1 Fixed-step online updates

At time t, we update the active node coefficients using the error-correcting signal

$$\delta_t \in -\partial_{\hat{y}_t} \ell(y_t, \hat{y}_t).$$

For pinball and Huberized-pinball losses, we use the subgradient selections in (1).

Given a fixed step size $\eta > 0$, the online update is

$$\gamma_{v,t+1} = \begin{cases} \gamma_{v,t} + \eta\, \delta_t, & v \in \mathrm{path}(\mathbf{x}_t), \\ \gamma_{v,t}, & v \notin \mathrm{path}(\mathbf{x}_t). \end{cases}$$

The following result shows that, under convex Lipschitz losses, this fixed-step online procedure achieves sublinear regret relative to any fixed comparator on the same tree structure.

Theorem 1. *Let*

$$y_t^\star = \sum_{v \in \mathrm{path}(\mathbf{x}_t)} \gamma_v^\star$$

for fixed comparator coefficients $\gamma_v^\star$, $v \in \mathcal{V}$. Suppose that, for each y, the loss $\ell(y, \hat{y})$ is convex and L-Lipschitz in $\hat{y}$, and that $\|\gamma_{v,1} - \gamma_v^\star\|_2 \le R$ for all $v \in \mathcal{V}$. Then

$$\mathrm{Regret}_T(y^\star) := \sum_{t=1}^{T} \big[\ell(y_t, \hat{y}_t) - \ell(y_t, y_t^\star)\big] \le |\mathcal{V}| \left(\frac{R^2}{2\eta} + \frac{\eta T L^2}{2} \right).$$

Choosing $\eta = R/(L\sqrt{T})$ gives

$$\mathrm{Regret}_T(y^\star) \le |\mathcal{V}| R L \sqrt{T}.$$

Proof: See Appendix A.

2.2 Online Learning with Dynamic Step Sizes

We extend IDBD-style adaptation to NP by assigning each node coefficient $\gamma_{v,t}$ its own time-varying step size $\eta_{v,t} = e^{\beta_{v,t}}$, for $v \in \mathcal{V}$ and $t \ge 1$, where, $\beta_{v,t}$ denotes the log-step-size parameter. As in IDBD, we define a time-varying memory parameter $z_{v,t}$ for each node v, which tracks recent updates to the node's coefficient. Algorithm 1 summarizes the resulting update rule.

The log-step size is updated proportionally to the product of the memory parameter and the error-correcting signal, scaled by a meta-learning rate θ:

$$\beta_{v,t+1} = \beta_{v,t} + \theta z_{v,t}\, \delta_t / n_t.$$

Thus, if recent and current update signals are aligned, the corresponding step size is amplified geometrically; if they oppose one another, it is damped. When the signals oscillate in sign and largely cancel, the step size remains relatively stable.

The memory update uses

$$c_t := \frac{\partial \delta_t}{\partial \hat{y}_t},$$

which measures the sensitivity of the error-correcting signal to the current forecast. For squared-error loss, $c_t = -2$, so the memory update includes a decay term. For pinball loss, we skip updates when $y_t = \hat{y}_t$, where the derivative is undefined; otherwise $c_t = 0$. For the Huberized-pinball loss, the decay term reappears in the quadratic region. Following Sutton (1992), we truncate the decay term using the positive-part operator $[x]^+ := \max(x, 0)$.

IDBD-style updates can be numerically unstable (Mahmood et al., 2012). In NP, because each observation activates multiple node coefficients, the aggregate forecast correction can scale with path length n_t when no path-length normalization is used. To control this effect, we apply the IDBD-style derivation to the normalized instantaneous loss $\ell(y_t, \hat{y}_t)/n_t$, yielding updates driven by the signal δ_t/n_t. In practice, this normalization improves numerical stability. Unlike the normalization used in Autostep, which requires additional adaptive scaling of the step-size updates, the path-length normalization used here follows directly from the normalized loss and provides a simple stability mechanism for NP.

Algorithm 1: IDBD-style update rule for NP

Input: Loss function ℓ
Initialize: z_v to 0, γ_v and β_v as desired, $v \in \mathcal{V}$
for each new example $(\mathbf{x}, y)$ **do**
  $n = |\operatorname{path}(\mathbf{x})|$
  $\hat{y} = \sum_{v \in \operatorname{path}(\mathbf{x})} \gamma_v$
  $\delta \in -\partial_{\hat{y}} \ell(y, \hat{y})$
  $c = \frac{\partial \delta}{\partial \hat{y}}$
  for $v \in \operatorname{path}(\mathbf{x})$ **do**
    $\beta_v \leftarrow \beta_v + \theta\, z_v\, \delta/n$
    $\eta_v \leftarrow e^{\beta_v}$
    $\gamma_v \leftarrow \gamma_v + \eta_v\, \delta/n$
    $z_v \leftarrow z_v \big[1 + c\, \eta_v/n\big]^+ + \eta_v\, \delta/n$
  end
end

Because NP shares parameters across leaves, node-specific step-size adaptation provides a mechanism for allocating coefficient updates across the hierarchy. In principle, higher-level nodes can respond to broad shared shifts, while lower-level nodes can adapt to more localized changes.

Appendix B shows that, under the no-cross-dependency approximation, the IDBD-style updates in Algorithm 1 follow from a stochastic gradient step on the normalized loss $\ell(y_t, \hat{y})/n_t$ with respect to $\beta_{v,t}$. The positive-part operator $[\cdot]^+$ is an additional stability modification not implied by the derivation.

3 Numerical Experiments

We evaluate NP on a publicly available dataset of emergency calls placed in Montgomery County, Pennsylvania, between January 2016 and August 2020 (Chirico, 2017). Each record includes the call time, caller location, and receiving department. We focus on EMS calls and compare temporal and spatiotemporal probabilistic forecasts from NP with benchmark models.

We use the first 70% of the data (168 weeks) for training and hyperparameter tuning and the final 30% (73 weeks) for testing. The test period includes the onset of the COVID-19 pandemic, which

induced a marked change in EMS demand, including a sharp drop in call volume, and therefore provides a useful setting for assessing adaptation under nonstationarity.

3.1 Forecasting Tasks, Baselines, and Evaluation

We consider two 2-day-ahead forecasting scenarios. In Scenario 1, we generate citywide forecasts at 1-hour resolution. In Scenario 2, we generate forecasts at 2-hour resolution for 14 spatial regions obtained by rasterizing the county with a 4×4 grid and retaining 14 populated cells. At each forecast origin, all target intervals in the second day ahead are predicted simultaneously; thus, in Scenario 2, forecasts are generated jointly for all 2-hour intervals in the second day across all 14 regions. These scenarios contain 40,488 and 283,416 observations, respectively.

We compare *NP-batch* and *NP-online* with three benchmark models: ETS(A,A,A), an exponential-smoothing model; XGBoost, a gradient-boosted tree ensemble; and Quantile Regression Forest (QRF), a tree-based ensemble for quantile prediction. NP-batch learns the tree structure and node coefficients offline, whereas NP-online reuses the selected NP-batch tree structure and updates only node coefficients online via Algorithm 1; see Appendix C for details.

NP-batch, XGBoost, and QRF are trained once and not updated during the test period. ETS(A,A,A) and NP-online are updated sequentially at the end of each day using newly observed data. For NP-batch, XGBoost, and QRF, we use temporal features (year, month, weekday, hour of day, and hour of week) and spatial features (grid ID, latitude, and longitude, when applicable). For XGBoost and QRF, we additionally include the 10 most recent lags available at prediction time and five moving averages. For ETS(A,A,A), we use a separate seasonal index for each time interval of the week and, in Scenario 2, train one independent model per region. Negative forecasts are truncated to zero.

Hyperparameters are selected using a temporally ordered training/validation split. For NP-batch, this includes the tree-growth and regularization parameters, as well as the Huberized-pinball smoothing parameter κ when that loss is used. For NP-online, we tune the meta-learning rate and, when applicable, the Huberized-pinball smoothing parameter κ. We also compare a warm-start initialization based on the NP-batch coefficients with zero initialization of all node coefficients; zero initialization is selected in all cases except for Scenario 2 at $\alpha = 0.05$. Corresponding model-specific hyperparameters are tuned for the benchmark methods: the level, trend, and seasonality smoothing parameters for ETS(A,A,A); the learning-rate, maximum depth, and regularization parameters for XGBoost; and the feature-sampling ratio and ensemble size for QRF. For online models, validation and testing are carried out sequentially so that updates use only past observations.

For each test observation $i = 1, \ldots, N_{\text{test}}$ and model m, let $\hat{y}_{i,m}^{(\alpha)}$ denote the predicted α-quantile and $\hat{y}_{i,m}$ the predicted mean. We evaluate forecasts using the mean pinball loss at quantile level α, denoted by $\text{PL}_\alpha(m)$, and the mean squared error, denoted by $\text{MSE}(m)$:

$$\text{PL}_\alpha(m) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \rho_\alpha\left(y_i - \hat{y}_{i,m}^{(\alpha)}\right), \qquad \text{MSE}(m) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \left(y_i - \hat{y}_{i,m}\right)^2.$$

The code used to train the NP models in our numerical experiments is available at https://github.com/mrez746/Nested-Pooling.git.

3.2 Results

Table 1 reports mean pinball loss and mean squared error for both scenarios. In Scenario 1 (citywide, 1-hour resolution), ETS(A,A,A) performs best for the symmetric targets, namely the median and mean, but performs worst at all asymmetric quantile levels except $\alpha = 0.75$. NP-online ranks second for the median and mean and achieves the best pinball loss at all non-median quantile levels except $\alpha = 0.05$, where it ranks second behind NP-batch. These results indicate that online step-size adaptation is particularly helpful in the temporal setting under asymmetric losses.

In Scenario 2 (14-region, 2-hour resolution), ETS(A,A,A) ranks first for the mean, while NP-online ranks second. NP-batch achieves the best quantile forecasts at five of the seven reported quantile levels. NP-online remains competitive but does not improve on NP-batch in this higher-resolution spatiotemporal setting.

A plausible explanation for the stronger performance of NP-online in the temporal task is that online step-size adaptation improves forecast discrimination, allowing the model to respond faster to changing aggregate demand conditions. Our decomposition of mean pinball loss suggests that NP-online attains better discrimination but worse calibration than NP-batch. In Scenario 1, this trade-off appears favorable because citywide call volumes vary over a wider range. In Scenario 2, region-level volumes are smaller and more tightly bounded, so stable and well-calibrated forecasts appear more valuable than additional discrimination. Additional evidence based on the miscalibration and discrimination components of mean pinball loss is reported in Appendix D.

Table 1: Mean pinball loss for quantile targets and mean squared error for the mean target across two 2-day-ahead forecasting scenarios. Bold values indicate the best-performing model.

	Target	ETS(A,A,A)	NP-batch	NP-online	XGBoost	QRF
Scenario 1: Citywide/1-hour	0.05 q	0.2825	**0.2686**	0.2691	0.2697	0.2711
	0.10 q	0.4890	0.4765	**0.4721**	0.4756	0.4742
	0.25 q	0.8979	0.8927	**0.8889**	0.8956	0.8966
	0.50 q	**1.1613**	1.1779	1.1683	1.1734	1.1771
	0.75 q	0.9771	0.9789	**0.9734**	0.9770	0.9820
	0.90 q	0.5739	0.5652	**0.5639**	0.5663	0.5698
	0.95 q	0.3488	0.3440	**0.3400**	0.3425	0.3452
	Mean	**8.9574**	9.2587	9.0169	9.1605	9.3409
Scenario 2: 14-regions/2-hour	0.05 q	0.0601	0.0563	0.0564	0.0578	**0.0560**
	0.10 q	0.1140	**0.1068**	0.1072	0.1092	0.1072
	0.25 q	0.2396	**0.2295**	0.2299	0.2433	0.2304
	0.50 q	0.3432	**0.3318**	0.3326	0.3402	0.3331
	0.75 q	0.3130	**0.3060**	0.3065	0.3108	0.3062
	0.90 q	0.1934	0.1886	0.1885	0.1905	**0.1879**
	0.95 q	0.1207	**0.1170**	0.1174	0.1176	0.1174
	Mean	**1.2428**	1.2604	1.2463	1.2507	1.2603

Bonferroni-adjusted bootstrap confidence intervals based on 1,000 resamples, controlling the familywise confidence level at 95%, show that NP-online significantly outperforms ETS(A,A,A) at all quantile levels except the median in Scenario 1, with no significant difference for the mean in either scenario. Relative to NP-batch, NP-online significantly improves mean performance in both scenarios and median performance in Scenario 1; otherwise the intervals typically include zero. NP-batch and QRF do not differ significantly for any reported quantile target or for the mean in Scenario 2, while in Scenario 1 NP-batch is significantly better only for the mean; see Appendix E for details.

4 Discussion and Conclusion

We proposed an IDBD-style adaptive step-size method for online learning in NP under convex, possibly nonsmooth, loss functions, and established a sublinear-regret guarantee for the corresponding fixed-step updates. In probabilistic forecasting experiments on emergency services demand, the adaptive method improved performance across most quantile levels in the temporal forecasting task, while the batch-trained NP model performed best in the higher-resolution spatiotemporal setting.

These results suggest that the benefits of adaptive online updates depend on the forecasting setting. In the temporal task, online coefficient and step-size adaptation appears to help the model respond to nonstationarity induced by the COVID-19 shock. In the spatiotemporal task, however, the gains

from meta-learning were smaller, and the strongest results were obtained by the batch-trained NP model.

A limitation of the present work is that the tree structure is learned offline and only the node coefficients are adapted online. An important direction for future work is therefore to study fully online versions of NP that update both the coefficients and the tree structure, for example by incorporating ideas from incremental decision-tree methods such as Hoeffding trees (Domingos & Hulten, 2000). This limitation is particularly relevant in continually changing environments, where new structural patterns may emerge that are not well represented by the initial offline tree. Another direction is to study whether the path-length normalization introduced here can improve the stability and performance of other IDBD-style methods, such as TIDBD, and to evaluate the resulting updates in additional forecasting settings.

A Theorem 1 Proof

Proof. By convexity of $\ell(y, \hat{y})$ in its second argument and since $-\delta_t \in \partial_{\hat{y}_t} \ell(y_t, \hat{y}_t)$, we have

$$\ell(y_t, \hat{y}_t) - \ell(y_t, y_t^\star) \leq -\delta_t(\hat{y}_t - y_t^\star).$$

Therefore,

$$\begin{aligned}
\text{Regret}_T(y^\star) &= \sum_{t=1}^{T} \{\ell(y_t, \hat{y}_t) - \ell(y_t, y_t^\star)\} && (2)\\
&\leq \sum_{t=1}^{T} -\delta_t(\hat{y}_t - y_t^\star) && (3)\\
&= \sum_{t=1}^{T} \sum_{v \in \text{path}(\mathbf{x}_t)} -\delta_t(\gamma_{v,t} - \gamma_v^\star) && (4)\\
&= \sum_{t=1}^{T} \sum_{v \in \text{path}(\mathbf{x}_t)} \left\{ \frac{\|\gamma_{v,t} - \gamma_v^\star\|_2^2 - \|\gamma_{v,t+1} - \gamma_v^\star\|_2^2}{2\eta} + \frac{\eta}{2}\delta_t^2 \right\} && (5)\\
&\leq \sum_{v \in \mathcal{V}} \sum_{t=1}^{T} \left\{ \frac{\|\gamma_{v,t} - \gamma_v^\star\|_2^2 - \|\gamma_{v,t+1} - \gamma_v^\star\|_2^2}{2\eta} + \frac{\eta}{2}\delta_t^2 \right\} && (6)\\
&= \sum_{v \in \mathcal{V}} \left\{ \frac{\|\gamma_{v,1} - \gamma_v^\star\|_2^2 - \|\gamma_{v,T+1} - \gamma_v^\star\|_2^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \delta_t^2 \right\} && (7)\\
&\leq \sum_{v \in \mathcal{V}} \left\{ \frac{\|\gamma_{v,1} - \gamma_v^\star\|_2^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \delta_t^2 \right\}. && (8)
\end{aligned}$$

Here, Line 5 uses

$$-\delta_t(\gamma_{v,t} - \gamma_v^\star) = \frac{\|\gamma_{v,t} - \gamma_v^\star\|_2^2 - \|\gamma_{v,t+1} - \gamma_v^\star\|_2^2}{2\eta} + \frac{\eta}{2}\delta_t^2,$$

obtained by expanding and rearranging $\|\gamma_{v,t+1} - \gamma_v^\star\|_2^2$, and Line 7 follows by telescoping the first term over t.

Now suppose $\|\gamma_{v,1} - \gamma_v^\star\|_2 \leq R$ for all $v \in \mathcal{V}$. Since $\ell(y, \hat{y})$ is L-Lipschitz in $\hat{y}$ and $\delta_t \in -\partial_{\hat{y}_t} \ell(y_t, \hat{y}_t)$, every such subgradient satisfies $|\delta_t| \leq L$. Hence

$$\text{Regret}_T(y^\star) \leq \sum_{v \in \mathcal{V}} \left\{ \frac{R^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} L^2 \right\} = |\mathcal{V}| \left(\frac{R^2}{2\eta} + \frac{\eta T L^2}{2} \right).$$

Finally, setting $\eta = R/(L\sqrt{T})$ gives

$$\text{Regret}_T(y^\star) \leq |\mathcal{V}| R L \sqrt{T}.$$

□

B Update Derivation

Define the normalized instantaneous loss

$$\tilde{\ell}_t = \ell(y_t, \hat{y}_t)/n_t.$$

If $\delta_t \in -\partial_{\hat{y}_t}\ell(y_t, \hat{y}_t)$, then $\delta_t/n_t \in -\partial_{\hat{y}_t}\tilde{\ell}_t$. Since

$$\hat{y}_t = \sum_{w \in \mathrm{path}(\mathbf{x}_t)} \gamma_{w,t},$$

we have

$$\partial_{\gamma_{v,t}}\tilde{\ell}_t = \partial_{\hat{y}_t}\tilde{\ell}_t, \qquad v \in \mathrm{path}(\mathbf{x}_t),$$

and hence

$$-\delta_t/n_t \in \partial_{\gamma_{v,t}}\tilde{\ell}_t, \qquad v \in \mathrm{path}(\mathbf{x}_t).$$

Taking a subgradient step on $\beta_{v,t}$ with meta-learning rate θ gives

$$\beta_{v,t+1} = \beta_{v,t} - \theta h_{v,t}, \qquad h_{v,t} \in \partial_{\beta_{v,t}}\tilde{\ell}_t.$$

By the chain rule,

$$h_{v,t} = \sum_{w \in \mathcal{V}} g_{w,t}\frac{\partial \gamma_{w,t}}{\partial \beta_{v,t}}, \qquad g_{w,t} \in \partial_{\gamma_{w,t}}\tilde{\ell}_t.$$

Define

$$z_{v,t} := \frac{\partial \gamma_{v,t}}{\partial \beta_{v,t}}.$$

Under the no-cross-dependency approximation,

$$\frac{\partial \gamma_{w,t}}{\partial \beta_{v,t}} = 0, \qquad w \neq v,$$

so choosing $g_{v,t} = -\delta_t/n_t$ yields

$$\beta_{v,t+1} = \beta_{v,t} + \theta z_{v,t}\delta_t/n_t.$$

Since $\eta_{v,t} = e^{\beta_{v,t}}$, we have

$$\frac{\partial \eta_{v,t}}{\partial \beta_{v,t}} = \eta_{v,t}.$$

Thus,

$$\begin{aligned} z_{v,t+1} = \frac{\partial \gamma_{v,t+1}}{\partial \beta_{v,t}} &= \frac{\partial}{\partial \beta_{v,t}}\left(\gamma_{v,t} + \eta_{v,t}\delta_t/n_t\right) \\ &= z_{v,t} + \frac{\eta_{v,t}}{n_t}\delta_t + \frac{\eta_{v,t}}{n_t}\frac{\partial \delta_t}{\partial \beta_{v,t}} \\ &= z_{v,t} + \frac{\eta_{v,t}}{n_t}\delta_t + \frac{\eta_{v,t}}{n_t}\frac{\partial \delta_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial \beta_{v,t}}. \end{aligned}$$

Using

$$\frac{\partial \hat{y}_t}{\partial \gamma_{v,t}} = 1, \qquad v \in \mathrm{path}(\mathbf{x}_t),$$

and again ignoring cross-dependencies, we obtain

$$z_{v,t+1} = z_{v,t} + \frac{\eta_{v,t}}{n_t}\delta_t + \frac{\eta_{v,t}}{n_t}\frac{\partial \delta_t}{\partial \hat{y}_t}z_{v,t}.$$

Defining

$$c_t := \frac{\partial \delta_t}{\partial \hat{y}_t},$$

gives

$$z_{v,t+1} = z_{v,t}\left(1 + \eta_{v,t}c_t/n_t\right) + \eta_{v,t}\delta_t/n_t.$$

References

Jochen Bröcker. Reliability, sufficiency, and the decomposition of proper scores. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 135(643):1512–1519, 2009.

Mike Chirico. Montgomery County 911 Calls Dataset, 2017. URL https://www.kaggle.com/datasets/mchirico/montcoalert. Accessed: 2026-04-13.

Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 71–80, 2000.

Tilmann Gneiting, Daniel Wolffram, Johannes Resin, Kristof Kraus, Johannes Bracher, Timo Dimitriadis, Veit Hagenmeyer, Alexander I Jordan, Sebastian Lerch, Kaleb Phipps, et al. Model diagnostics and forecast evaluation for quantiles. *Annual Review of Statistics and Its Application*, 10(1):597–621, 2023.

Robert A Jacobs. Increased rates of convergence through learning rate adaptation. *Neural networks*, 1(4):295–307, 1988.

Alex Kearney, Vivek Veeriah, Jaden B Travnik, Richard S Sutton, and Patrick M Pilarski. TIDBD: Adapting temporal-difference step-sizes through stochastic meta-descent. *ArXiv:1804.03334*, 2018.

Ashique Rupam Mahmood, Richard S Sutton, Thomas Degris, and Patrick M Pilarski. Tuning-free step-size adaptation. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 2121–2124. IEEE, 2012.

Nicol N Schraudolph. Local gain adaptation in stochastic gradient descent. In *1999 Ninth international conference on artificial neural networks ICANN 99.(Conf. Publ. No. 470)*, volume 2, pp. 569–574. IET, 1999.

Richard S Sutton. Adaptation of learning rate parameters. *Goal Seeking Components for Adaptive Intelligence: An Initial Assessment, by AG Barto and RS Sutton. Air Force Wright Aeronautical Laboratories Technical Report AFWAL-TR-81-1070. Wright-Patterson Air Force Base, Ohio*, 45433, 1981.

Richard S Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *AAAI*, volume 92, pp. 171–176. San Jose, CA, 1992.

Richard S Sutton. A history of meta-gradient: Gradient methods for meta-learning. *ArXiv preprint arXiv:2202.09701*, 2022.

Supplementary Materials

The following content was not necessarily subject to peer review.

C Batch Training of Nested Pooling

The batch-trained Nested Pooling model (NP-batch) is learned offline by a boosting-style tree-growing procedure, in which remaining residuals from the training observations are fitted iteratively. Residuals are initialized to the response values and, after each iteration, are updated using the current fitted model so that the next iteration predicts the remaining residuals. Each iteration adds structure that explains part of the remaining forecast error.

More specifically, each iteration: (i) selects the leaf with the largest current training loss, (ii) computes pseudo-residuals for the observations in that leaf, (iii) chooses the split that best estimates those pseudo-residuals, (iv) estimates coefficients for the two child nodes by minimizing loss plus an ℓ_2 penalty within each child region, and (v) updates the fitted model and residuals. In step (ii), we calculate pseudo-residuals as $\delta_i \in -\partial_{\hat{y}}\ell(y_i, \hat{y}_i)$ for the observations in the selected leaf node, where predictions are derived from the current model fit. We use the subgradient selection in (1) when the loss function is the quantile or Huberized-quantile losses.

Tree growth is terminated using validation performance. The final NP-batch model consists of the selected tree structure together with its fitted node coefficients. In the experiments, NP-online reuses this selected tree structure and updates only the node coefficients online, which are initialized either at zero or at the NP-batch values, with the choice treated as a tuning option during cross-validation. In the reported experiments, zero initialization was selected in all cases except for Scenario 2 at $\alpha = 0.05$, where the best cross-validated NP-online model used warm-start initialization from the NP-batch coefficients.

D Mean Pinball Loss Decomposition

A quantile forecast is unconditionally α-quantile calibrated if the proportion of observations below the predicted α-quantile equals α. It is *conditionally α-quantile calibrated* if this property holds within strata defined by the forecast value.

Following Gneiting et al. (2023), we decompose the empirical mean pinball loss for model m at quantile level α as

$$\text{PL}_\alpha(m) = \text{MCB}_\alpha(m) - \text{DSC}_\alpha(m) + \text{UNC}_\alpha,$$

where $\text{MCB}_\alpha(m)$, $\text{DSC}_\alpha(m)$, and UNC_α denote the miscalibration, discrimination, and uncertainty components, respectively.

The miscalibration component $\text{MCB}_\alpha(m)$ measures the difference in mean pinball loss between the model's predicted quantiles and a recalibrated version obtained post hoc using the pool-adjacent-violators algorithm (Gneiting et al., 2023). By construction, the recalibrated forecasts are conditionally α-quantile calibrated. Lower values of $\text{MCB}_\alpha(m)$ are preferred, as they indicated better calibration.

The discrimination component $\text{DSC}_\alpha(m)$ measures the improvement in mean pinball loss of the recalibrated forecasts relative to a calibrated constant forecast equal to an empirical α-quantile of the test set. Discrimination, also called resolution, reflects how informative the forecasts are (Bröcker, 2009). Higher values of $\text{DSC}_\alpha(m)$ are preferred.

The uncertainty component UNC_α is the mean pinball loss of the calibrated constant forecast. Because it depends only on the realized response values, it is the same for all models at a given quantile level. All three components are nonnegative by construction.

Figure 2 reports the miscalibration and discrimination components for Scenario 1. ETS(A,A,A) exhibits the highest discrimination at all quantile levels, indicating highly responsive forecasts, but

also the highest miscalibration. QRF shows the opposite pattern, with the lowest miscalibration but also the lowest discrimination. NP-online, NP-batch, and XGBoost lie between these extremes. Relative to NP-batch, NP-online achieves higher discrimination at all quantile levels, but at the cost of worse calibration. This pattern helps explain why NP-online performs well in Scenario 1, where higher responsiveness appear beneficial.

Table 2 reports the average miscalibration and discrimination score for Scenario 2. In the spatiotemporal setting, where regional call volumes are smaller and have narrower range, trading better discrimination for worse calibration appears less favorable, which is consistent with the stronger performance of NP-batch in that scenario.

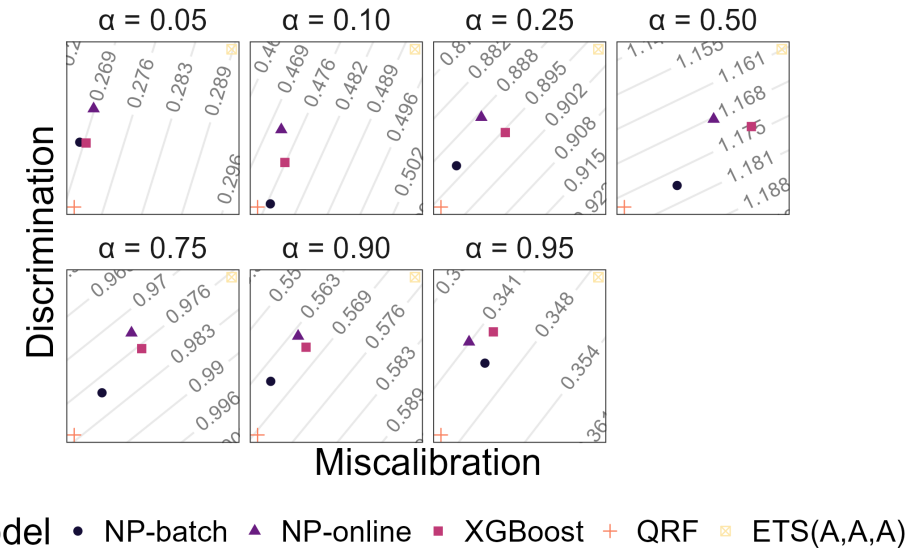

Figure 2: Miscalibration and discrimination components of the mean pinball loss for each model in Scenario 1.

Table 2: Average miscalibration and discrimination scores (MCB, DSC) across regions for Scenario 2.

Quantile	ETS(A,A,A)	NP-batch	NP-online	XGBoost	QRF
0.05	(1.66, 0.22)	(0.00, 0.14)	(0.02, 0.16)	(0.00, 0.00)	(0.01, 0.18)
0.10	(2.23, 0.70)	(0.00, 0.59)	(0.08, 0.62)	(0.00, 0.36)	(0.01, 0.56)
0.25	(2.08, 2.37)	(0.19, 2.25)	(0.31, 2.30)	(0.88, 1.56)	(0.05, 2.03)
0.50	(1.76, 4.50)	(0.12, 4.24)	(0.33, 4.34)	(0.84, 4.11)	(0.03, 4.01)
0.75	(1.29, 5.04)	(0.21, 4.68)	(0.36, 4.78)	(0.72, 4.70)	(0.02, 4.47)
0.90	(0.90, 3.02)	(0.08, 2.68)	(0.25, 2.84)	(0.45, 2.86)	(0.04, 2.71)
0.95	(0.65, 1.88)	(0.10, 1.70)	(0.21, 1.78)	(0.25, 1.79)	(0.05, 1.62)

E Bootstrap Confidence Intervals for Pairwise Performance Difference

Table 3: Bonferroni-adjusted bootstrap confidence intervals for the difference in loss between NP-online and ETS(A,A,A). Negative values favor NP-online, and positive values favor ETS(A,A,A).

	Scenario 1		Scenario 2	
Target	lower	upper	lower	upper
$\alpha = 0.05$	-0.019	-0.009	-0.004	-0.003
$\alpha = 0.10$	-0.023	-0.011	-0.008	-0.006
$\alpha = 0.25$	-0.018	-0.001	-0.011	-0.009
$\alpha = 0.50$	0.001	0.013	-0.012	-0.010
$\alpha = 0.75$	-0.011	0.004	-0.008	-0.005
$\alpha = 0.90$	-0.017	-0.002	-0.006	-0.004
$\alpha = 0.95$	-0.014	-0.003	-0.004	-0.002
Mean	-0.019	0.135	-0.001	0.008

Table 4: Bonferroni-adjusted bootstrap confidence intervals for the difference in loss between NP-online and NP-batch. Negative values favor NP-online, and positive values favor NP-batch.

	Scenario 1		Scenario 2	
Target	lower	upper	lower	upper
$\alpha = 0.05$	-0.003	0.004	0.000	0.000
$\alpha = 0.10$	-0.009	0.001	0.000	0.001
$\alpha = 0.25$	-0.011	0.003	0.000	0.001
$\alpha = 0.50$	-0.018	-0.001	0.000	0.002
$\alpha = 0.75$	-0.013	0.002	-0.001	0.002
$\alpha = 0.90$	-0.006	0.004	-0.001	0.001
$\alpha = 0.95$	-0.009	0.001	0.000	0.001
Mean	-0.363	-0.124	-0.020	-0.008

Table 5: Bonferroni-adjusted bootstrap confidence intervals for the difference in loss between NP-batch and QRF. Negative values favor NP-batch, and positive values favor QRF.

	Scenario 1		Scenario 2	
Target	lower	upper	lower	upper
$\alpha = 0.05$	-0.006	0.000	0.000	0.001
$\alpha = 0.10$	-0.004	0.008	-0.001	0.000
$\alpha = 0.25$	-0.010	0.003	-0.002	0.000
$\alpha = 0.50$	-0.007	0.010	-0.003	0.000
$\alpha = 0.75$	-0.011	0.004	-0.002	0.002
$\alpha = 0.90$	-0.011	0.001	-0.001	0.002
$\alpha = 0.95$	-0.006	0.005	-0.001	0.001
Mean	-0.161	-0.005	-0.004	0.004