An Exploration of Contextual Bandits

Matthew Shumway

April 18, 2025

Abstract

This work explores both classical and modern approaches to the contextual bandit problem. We investigate the performance of Linear Upper Confidence Bound (LinUCB), Linear Thompson Sampling (LinTS), and their neural network-based counterparts, NeuralUCB and NeuralTS. The decision-making strategies of LinUCB and LinTS are shown to follow naturally from Bayesian principles, while intuition is provided for the design of their neural analogs. We evaluate all methods on both synthetic data sampled from the unit ball and on the Iris dataset. The results highlight the trade-offs between exploration and function approximation across these algorithmic families.

1 Introduction and Motivation

The contextual bandit problem is a foundational framework in reinforcement learning (RL) and sequential decision-making. In this setting, an agent repeatedly selects actions based on observable context, receiving partial feedback in return. It lies between the multi-armed bandit (which lacks context) and the Markov Decision Process (which considers full state transitions), offering a balance between expressiveness and tractability.

Contextual bandits are widely used in real-world applications such as recommendation systems, adaptive clinical trials, and personalized tutoring [4, 6, 8, 9]. The framework also serves as a common starting point for theoretical advances in RL. For example, Posterior Sampling for Reinforcement Learning (PSRL) was motivated by Thompson Sampling [7].

In this project, we investigate the theoretical foundations of two classical algorithms: Upper Confidence Bound (UCB) and Thompson Sampling (TS), along with their extensions—LinUCB, LinTS, NeuralUCB, and NeuralTS. The goal is not to introduce new methods, but to deepen understanding through both theoretical analysis and empirical evaluation.

2 Mathematical Preliminaries

This section outlines the contextual bandit framework and key notation used throughout. At each time step $t \in [T]$, an agent interacts with the environment as follows:

- **Observe contexts:** The agent receives a set of context vectors $\{x_{t,a} \in \mathbb{R}^d\}_{a=1}^K$, one for each available action $a \in [K]$.
- **Select action:** Based on the observed contexts, the agent chooses an action $a_t \in [K]$.
- **Receive reward:** The agent receives a reward r_{t,a_t}, drawn from an unknown distribution dependent on x_{t,a_t}.

- **Update model:** The agent updates its internal parameters using the tuple $(x_{t,a_t}, a_t, r_{t,a_t})$.

The agent's objective is to maximize the expected cumulative reward:

$$\mathbb{E}\left[\sum_{t=1}^{T} r_{t,a_t}\right]. \tag{1}$$

Performance is commonly evaluated via regret, defined as the difference between the reward of the optimal action and the chosen action:

$$\text{Regret}(T) = \sum_{t=1}^{T} (r_t^* - r_{t,a_t}) \tag{2}$$

where r_t^* is the reward of the optimal action at time t. A desirable property is sublinear regret, i.e., $\text{Regret}(T) \in o(T)$, implying that the average regret tends to zero as $T \to \infty$.

3 Linear Models

3.1 Upper Confidence Bound and LinUCB

The Upper Confidence Bound (UCB) algorithm selects actions by balancing exploration and exploitation, choosing actions that are optimistic under current uncertainty [2]. The generic form is:

$$a_t = \underset{a \in [K]}{\arg\max}\ \hat{\mu}_a + \alpha \hat{\sigma}_a \tag{3}$$

where $\hat{\mu}_a$ and $\hat{\sigma}_a$ are the estimated mean and uncertainty (e.g., standard deviation) of arm a, and $\alpha \in \mathbb{R}_+$ controls the exploration-exploitation trade-off.

3.1.1 LinUCB: Linear UCB with Context

LinUCB [6] extends UCB to contextual bandits under the assumption that expected reward is linear in the context:

$$r_{t,a} = x_{t,a}^\top \theta_a^* + \varepsilon_t$$

where $x_{t,a} \in \mathbb{R}^d$ is the context vector for arm a at time t, $\theta_a^* \in \mathbb{R}^d$ is an unknown parameter vector, and $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$ is noise. At each timestep, the agent selects the arm

$$a_t = \underset{a \in [K]}{\arg\max}\ x_{t,a}^\top \hat{\theta}_{t,a} + \alpha \sqrt{x_{t,a}^\top A_{t,a}^{-1} x_{t,a}} \tag{4}$$

where

$$A_{t,a} = \lambda I + \sum_{\tau=1}^{t-1} x_{\tau,a_\tau} x_{\tau,a_\tau}^\top \cdot \mathbb{I}[a_\tau = a] \tag{5}$$

$$b_{t,a} = \sum_{\tau=1}^{t-1} r_\tau x_{\tau,a_\tau} \cdot \mathbb{I}[a_\tau = a] \tag{6}$$

$$\hat{\theta}_{t,a} = A_{t,a}^{-1} b_{t,a} \tag{7}$$

3.1.2 Bayesian Interpretation

LinUCB can be viewed through the lens of Bayesian linear regression, where $\theta_a \sim \mathcal{N}(0, \lambda^{-1} I)$ is a Gaussian prior and rewards are observed with Gaussian noise. Under this model, the posterior over θ_a is

$$\theta_a \mid D_{t,a} \sim \mathcal{N}(\hat{\theta}_{t,a}, A_{t,a}^{-1}) \tag{8}$$

and the posterior predictive distribution over reward is

$$\hat{r}_{t,a} = x_{t,a}^\top \theta_a \sim \mathcal{N}(x_{t,a}^\top \hat{\theta}_{t,a}, x_{t,a}^\top A_{t,a}^{-1} x_{t,a}) \tag{9}$$

Due to page constraints, the full derivation of this is given in Appendix A. LinUCB selects the arm with the highest upper confidence bound on expected reward under this posterior.

3.2 Thompson Sampling and LinTS

Thompson Sampling (TS) maintains a posterior distribution over the model parameters and selects actions by sampling from this posterior. Given a reward model $r(x_{t,a}; \theta_a)$, TS samples $\tilde{\theta}_a \sim P(\theta_a \mid \mathcal{D})$ at each round and chooses

$$a_t = \arg\max_{a \in [K]} \; r(x_{t,a}; \tilde{\theta}_a), \tag{10}$$

where $\mathcal{D}$ denotes the current dataset. This stochastic selection naturally balances exploration and exploitation.

3.2.1 Assumptions and Action Selection

LinTS [1] applies TS in the linear reward setting $r_{t,a} = x_{t,a}^\top \theta_a^*$. At each round, it samples

$$\tilde{\theta}_a \sim \mathcal{N}(A_{t,a}^{-1} b_{t,a}, A_{t,a}^{-1}), \tag{11}$$

and selects

$$a_t = \arg\max_{a \in [K]} \; x_{t,a}^\top \tilde{\theta}_a,$$

where $A_{t,a}$ and $b_{t,a}$ are defined in equations 5 and 6.

3.2.2 Bayesian Derivation

These updates follow directly from the Bayesian linear regression derivation used in LinUCB (Appendix A). Given the prior $\theta_a \sim \mathcal{N}(0, \lambda^{-1} I)$, the posterior is

$$\theta_a \mid \mathcal{D} \sim \mathcal{N}(\hat{\theta}_{t,a}, A_{t,a}^{-1}), \tag{12}$$

which leads to the LinTS sampling step in equation 11.

3.3 Differences in Exploration Between LinUCB and LinTS

While LinUCB and LinTS rely on the same Bayesian posterior over arm parameters (see Eq. 8), they differ in how uncertainty guides action selection. LinUCB uses an upper confidence bound (UCB) to optimistically favor uncertain arms, whereas LinTS samples from the posterior, introducing stochasticity into decisions. This difference results in distinct exploration dynamics: LinUCB deterministically favors arms with high uncertainty, while LinTS may naturally explore both high-mean and high-uncertainty arms. In practice, LinTS can outperform LinUCB without tuning, while LinUCB requires calibration of its exploration parameter α.

See Appendix B for a synthetic example comparing the selection behavior of LinUCB and LinTS in a fixed context setting.

3.4 Regret Analysis of LinUCB and LinTS

While detailed regret proofs are beyond the scope of this work, both LinUCB and LinTS achieve regret bounds of order $\widetilde{O}(\sqrt{T})$ under linear reward assumptions, where $\widetilde{O}$ hides logarithmic factors [1, 3].

3.5 Empirical Comparison of LinUCB and LinTS

We empirically compare LinUCB and LinTS under both linear and nonlinear reward functions. Following the setup in [11], we consider $d = 20$-dimensional contexts and $K = 4$ arms. Contexts are sampled uniformly from the unit ball in $\mathbb{R}^d$ [5], and each arm a has an unknown parameter $\theta_a \sim \mathcal{N}(0, 1)$.

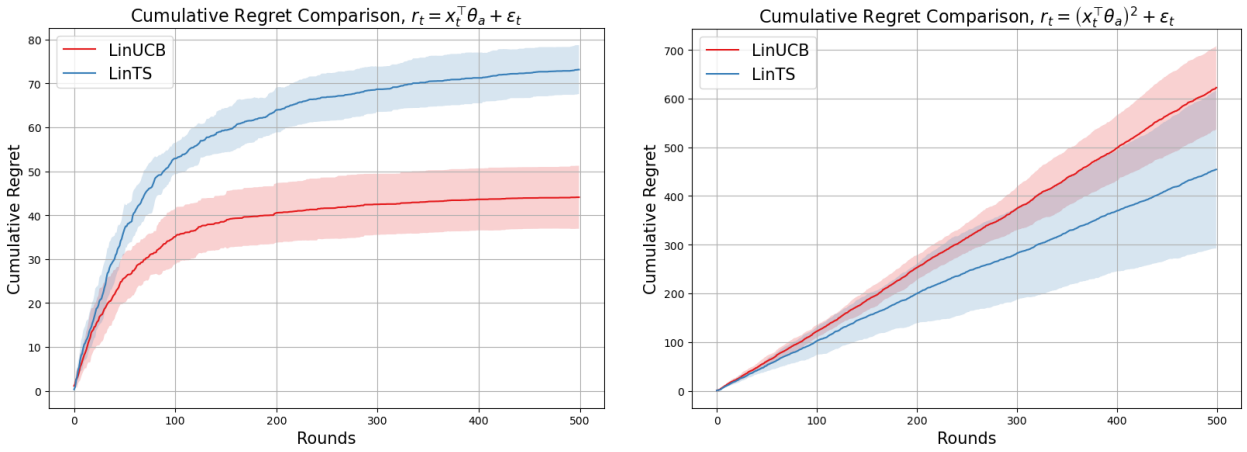
We test both a linear reward function $h_1(x) = x^\top \theta_a + \varepsilon_t$ and a quadratic reward function $h_2(x) = (x^\top \theta_a)^2 + \varepsilon_t$, where $\varepsilon_t \sim \mathcal{N}(0, 0.1^2)$. Results are shown in Figure 1, see Appendix C for full experimental details. Aligned with their theoretical guarantees, regret for both LinUCB and LinTS plateaus under the linear reward function. Under the quadratic reward function, regret appears to be linear. This highlights a need for bandit algorithms to learn in nonlinear settings.

4 Neural Models

To address the limitations of linear methods in modeling complex reward structures, we now consider neural contextual bandits. Specifically, we study NeuralUCB and NeuralTS, which leverage neural networks for flexible reward estimation.

4.1 NeuralUCB

NeuralUCB [11] extends LinUCB by modeling expected rewards with a neural network $f(x; \theta)$. Exploration is guided by an upper confidence bound, where the mean is taken as



(a) Linear rewards

(b) Quadratic rewards

Figure 1: Comparison of LinUCB and LinTS on linear vs. nonlinear rewards. Both methods exhibit sublinear regret under linear rewards but fail under nonlinear structure. Shaded regions show confidence intervals over five runs.

the output of the $f(x;\theta)$ and the uncertainty is estimated with the gradients of the network. The idea is that large gradients imply high sensitivity to the model parameters and greater uncertainty in that region of the input space. The exact equations defining the uncertainty estimate are provided in Appendix D.

4.2 NeuralTS

NeuralTS [10] models the reward directly using a neural network and samples from its predictive distribution. Unlike standard Thompson Sampling, it samples scalar outputs rather than model parameters:

$$r_{t,a} \sim \mathcal{N}(f(x_{t,a};\theta), \nu^2 \sigma_{t,a}^2),$$

where $\sigma_{t,a}$ is the same gradient-based estimate as NeuralUCB and ν^2 is a chosen variance hyperparameter. Full details appear in Appendix E.

4.3 Theoretical Guarantees

Both methods achieve sublinear regret $\widetilde{O}(\sqrt{T})$ under mild assumptions, similar to their linear counterparts [10, 11].

4.4 Empirical Performance

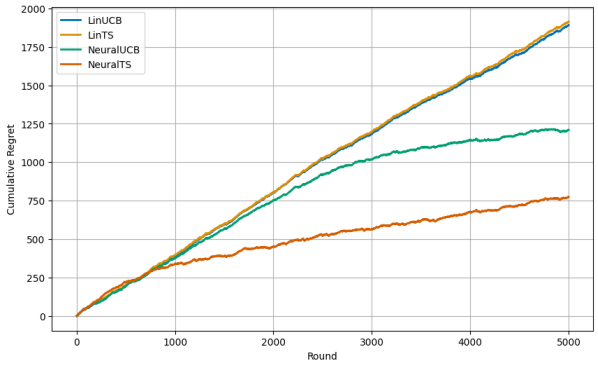
Figure 2a shows performance on the previously described synthetic data using the quadratic reward function h_2 from Section 3.5. Neural methods plateau, suggesting sublinear regret, while linear methods do not. Similar behavior is seen on the Iris dataset (Figure 2b), where NeuralUCB initially over-explores. Both experiments highlight the power of these methods to learn optimal behavior in nonlinear settings.

4.4.1 K-Class Classification as a Contextual Bandit

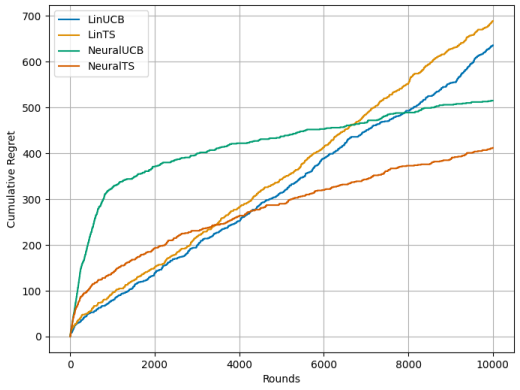
We frame K-class classification as a K-armed contextual bandit problem: each class corresponds to an arm $a \in [K]$, and rewards are defined as

$$r_t = \begin{cases} 1 & \text{if } a_t = y_t \\ 0 & \text{otherwise} \end{cases}$$

The Iris dataset has 3 classes and 4 features. Results in Figure 2b mirror synthetic trends.



(a) Synthetic dataset: Linear methods show linear regret, while neural methods plateau.



(b) Iris dataset: NeuralUCB appears to overexplore initially, but both neural methods plateau.

Figure 2: Comparison of all four algorithms (LinUCB, LinTS, NeuralUCB, NeuralTS) on synthetic and real-world datasets.

5 Discussion and Conclusion

This study highlights the advantages of neural contextual bandit algorithms, specifically NeuralUCB and NeuralTS, over traditional methods like LinUCB and LinTS. Both neural models demonstrate sublinear regret, outperforming the linear approaches on synthetic and real-world datasets. While NeuralUCB achieves good performance, it exhibits overexploration in the early rounds on the Iris dataset, which suggests a need for improved exploration-exploitation balance.

Future work could focus on refining exploration strategies in NeuralUCB, improving scalability to larger datasets, and exploring advanced network architectures for more complex reward functions. Overall, neural bandits show great promise for handling nonlinear relationships in contextual bandit problems, but further development is needed to address challenges in exploration and scalability.

References

[1] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International conference on machine learning*, pages 127–135. PMLR, 2013.

[2] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[3] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 208–214. JMLR Workshop and Conference Proceedings, 2011.

[4] Benjamin Clement, Didier Roy, Pierre-Yves Oudeyer, and Manuel Lopes. Multi-armed bandits for intelligent tutoring systems. *arXiv preprint arXiv:1310.3174*, 2013.

[5] Jeffrey Humpherys, Tyler J. Jarvis, and Emily J. Evans. *Foundations of Applied Mathematics, Volume 3: Modeling Uncertainty with Data.*

[6] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[7] Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. *Advances in Neural Information Processing Systems*, 26, 2013.

[8] Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018.

[9] Ambuj Tewari and Susan A Murphy. From ads to interventions: Contextual bandits in mobile health. *Mobile health: sensors, analytic methods, and applications*, pages 495–517, 2017.

[10] Weitong Zhang, Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural thompson sampling. *arXiv preprint arXiv:2010.00827*, 2020.

[11] Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, pages 11492–11502. PMLR, 2020.

A Bayesian Derivation of LinUCB

We derive the LinUCB update equations from Bayesian linear regression. Assume that the reward for arm a at time t follows the linear-Gaussian model:

$$r_{t,a} = x_{t,a}^\top \theta_a + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0,1)$$

with prior $\theta_a \sim \mathcal{N}(0, \lambda^{-1} I)$ for some regularization parameter $\lambda > 0$.

Let $D_{t,a} \in \mathbb{R}^{(t-1)\times d}$ denote the design matrix formed by stacking all feature vectors $x_{\tau,a}$ for which arm a was selected up to time $t-1$, and let $r_{1:t-1,a} \in \mathbb{R}^{t-1}$ denote the corresponding rewards. The likelihood is:

$$r_{1:t-1,a} \mid \theta_a, D_{t,a} \sim \mathcal{N}(D_{t,a}\theta_a, I)$$

By Bayes' rule, the posterior is:

$$\begin{aligned}
P(\theta_a \mid r_{1:t-1,a}, D_{t,a}) &\propto P(r_{1:t-1,a} \mid \theta_a, D_{t,a}) P(\theta_a \mid D_{t,a}) \\
&\propto \exp\left(-\frac{1}{2}\|r_{1:t-1,a} - D_{t,a}\theta_a\|^2\right) \cdot \exp\left(-\frac{\lambda}{2}\|\theta_a\|^2\right) \\
&= \exp\left(-\frac{1}{2}\left[\theta_a^\top \left(D_{t,a}^\top D_{t,a} + \lambda I\right)\theta_a - \left(2 r_{1:t-1,a}^\top D_{t,a}\right)\theta_a + \text{const}\right]\right) \\
&\propto \exp\left(-\frac{1}{2}\left[\theta_a^\top A_{t,a}\theta_a - 2 b_{t,a}\theta_a\right]\right) \\
&= \exp\left(-\frac{1}{2}\left[(\theta_a - A_{t,a}^{-1} b_{t,a})^\top A_{t,a} (\theta_a - A_{t,a}^{-1} b_{t,a})\right]\right) \\
&\propto \mathcal{N}\left(\hat{\theta}_a, A_{t,a}^{-1}\right)
\end{aligned}$$

where $A_{t,a}$, $b_{t,a}$, and $\hat{\theta}_{t,a}$ are defined in equations 5, 6, and 7, respectively. Then the posterior is Gaussian:

$$\theta_a \mid r_{1:t-1,a}, D_{t,a} \sim \mathcal{N}(\hat{\theta}_{t,a}, A_{t,a}^{-1}) \tag{13}$$

with posterior mean $\hat{\theta}_{t,a} = A_{t,a}^{-1} b_{t,a}$.

Finally, the predictive distribution for the reward of arm a at time t is:

$$\hat{r}_{t,a} = x_{t,a}^\top \theta_a \sim \mathcal{N}(x_{t,a}^\top \hat{\theta}_{t,a}, x_{t,a}^\top A_{t,a}^{-1} x_{t,a}) \tag{14}$$

The LinUCB algorithm selects the arm with the largest upper confidence bound:

$$a_t = \arg\max_{a \in [K]} \; x_{t,a}^\top \hat{\theta}_{t,a} + \alpha \sqrt{x_{t,a}^\top A_{t,a}^{-1} x_{t,a}}$$

This completes the Bayesian justification of the LinUCB action selection rule.

B Exploration Behavior: LinUCB vs. LinTS

To illustrate the different exploration strategies, we consider a synthetic contextual bandit scenario with a fixed context:

$$\mathbf{x} = \begin{bmatrix} 1.0 \\ 0.5 \end{bmatrix}$$

Each arm has a true reward parameter vector:

$$\boldsymbol{\theta}_0 = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix}, \quad \boldsymbol{\theta}_1 = \begin{bmatrix} 0.0 \\ 1.0 \end{bmatrix}$$

which yield expected rewards $\mu_0 = 1.0$ and $\mu_1 = 0.5$.

We simulate the agent's beliefs with synthetic Gaussian posteriors:

$$\text{Arm 0: } \mathcal{N}([0.5, 0.0],\ 0.1 \cdot I), \quad \text{Arm 1: } \mathcal{N}([0.0, 0.5],\ 1.0 \cdot I)$$

Figure 3 shows that LinUCB always selects the arm with the higher upper confidence bound (in this case, arm 1 due to high uncertainty), while LinTS samples from the posterior and occasionally selects the lower-variance arm due to its higher mean.

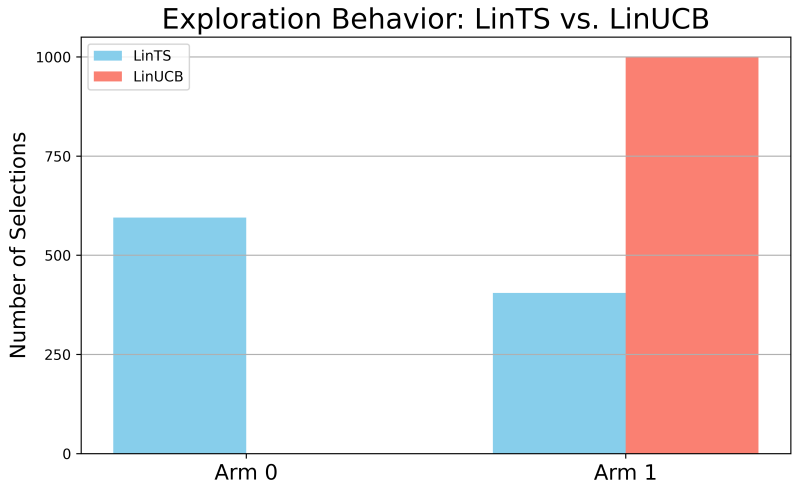


Figure 3: Empirical selection frequency over 1000 runs at a single decision point. LinTS stochastically explores, while LinUCB deterministically chooses the highest UCB arm.

C Experimental Setup for LinUCB and LinTS

For all experiments in Section 3.5, we use the following setup:

- Number of arms: $K = 4$
- Context dimension: $d = 20$
- Time horizon: $T = 500$
- Each context vector $x_{t,a}$ is drawn uniformly from the d-dimensional unit ball.
- True arm parameters $\theta_a \sim \mathcal{N}(0, I_d)$ independently for each arm.
- Additive noise: $\varepsilon_t \sim \mathcal{N}(0, 0.1^2)$

- Algorithms are averaged over 5 runs to compute regret and confidence intervals.

D NeuralUCB Derivation

NeuralUCB estimates uncertainty for arm a at time t via

$$\sqrt{\frac{\nabla f(x_{t,a};\theta_{t-1})^\top Z_{t-1}^{-1} \nabla f(x_{t,a};\theta_{t-1})}{m}},$$

where

$$Z_t = Z_{t-1} + \frac{1}{m}\nabla f(x_{t,a};\theta_{t-1})\nabla f(x_{t,a};\theta_{t-1})^\top.$$

This Mahalanobis norm captures gradient sensitivity, increasing in unexplored input regions.

E NeuralTS Derivation

NeuralTS trains the network by minimizing

$$\arg\min_\theta \frac{1}{2}\sum_{i=1}^{t}[f(x_{i,a_i};\theta) - r_{i,a_i}]^2 + \frac{m\lambda}{2}\|\theta - \theta_0\|_2^2.$$

It samples rewards as

$$r_{t,a} \sim \mathcal{N}(f(x_{t,a};\theta), \nu^2\sigma_{t,a}^2),$$

where $\sigma_{t,a}^2$ matches the uncertainty from NeuralUCB and ν^2 is a chosen variance parameter.