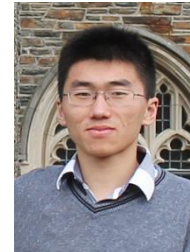
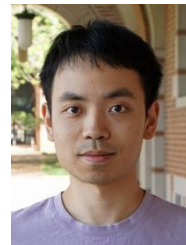
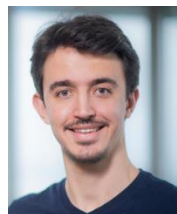


Normal Hedge, **Second Order** **Quantile** Regret, and LLM Acceleration

Yu-Xiang Wang
UCSD

Based on works with



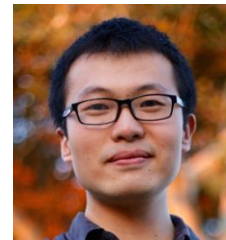
Learning from Expert Advice



Stock
Price

Day 1 Up
Day 2 Down
Day 3 Down
Day 4 Up
... ...
Day T Up

Stock “Experts”



Up 0	Down 1	Up 0
Down 0	Down 0	Down 0
Up 1	Down 1	Down 0
Down 1	Down 0	Up 0
...	...	...
Up 0	Down 1	Up 0

Cumulative Loss

17

55

0

Learner



M 0
W 0
M 1
M 1
...
R 0
11

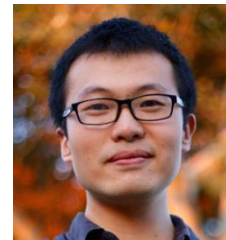
Loss = 0 if predict correctly, and **1** otherwise.

Learning from Expert Advice

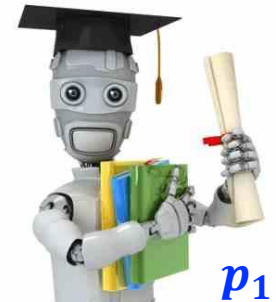


Stock
Price

Stock “Experts”



Learner



Day 1	+ 2%	0.2 $\ell_{1,1}$	0 $\ell_{1,2}$	1.0 $\ell_{1,3}$	p_1 [1/3, 1/3, 1/3]
Day 2	- 0.5%	0.2 $\ell_{2,1}$	0 $\ell_{2,2}$	0 $\ell_{2,3}$	p_2 [1/6, 5/12, 5/12]
Day 3	- 2.2%	0.5 $\ell_{3,1}$	0 $\ell_{3,2}$	0 $\ell_{3,3}$	p_3 [1/12, 1/12, 5/6]
Day 4	+5%	0.3 $\ell_{4,1}$	0 $\ell_{4,2}$	1.0 $\ell_{4,3}$	p_3
...	...	...	...	...	...
Day T	+3%	0.5 $\ell_{T,1}$	0 $\ell_{T,2}$	1.0 $\ell_{T,3}$	p_T

Return can be numerical.

Expert advice: proportion of capital to invest

Loss: - Log-of-Multiplicative-Gain

Loss at Round j :

$$\langle \mathbf{p}_j, \mathbf{\ell}_j \rangle$$

Learning from Expert Advice

Given number of experts N and a constant $B > 0$

For each iteration $j = 1, 2, 3, \dots$

1. Player chooses the weights as a probability vector $\mathbf{p}_j \in \Delta^{N-1}$.
2. Nature reveals a loss vector $\ell_j \in \mathbb{R}^N$ satisfying that $\max_{i, i' \in [N]} |\ell_{j,i} - \ell_{j,i'}| \leq B$.
3. Player incurs a loss $\langle \mathbf{p}_j, \ell_j \rangle = \mathbb{E}_{i \sim \mathbf{p}_j} [\ell_{j,i}]$.

- The goal of the learner:

Minimize **cumulative loss**: $\sum_{j=1}^T \langle \mathbf{p}_j, \ell_j \rangle$



$$\text{Regret} = \sum_{j=1}^T \langle \mathbf{p}_j, \ell_j \rangle - \min_i \sum_{j=1}^T \ell_{j,i}$$

**Cumulative loss of
the best expert**

LEA is a classical problem with decades of associated literature.

- A lot of pioneers of machine learning worked on it: [Vovk](#), [Littlestone](#), [Warmuth](#), [Lugosi](#), [Cesa-Bianchi](#), [Schapire](#), [Freund](#), ...
- Statistical / Information-theoretical roots: [[Robbins, 1951](#); [Blackwell, 1956](#); [Cover, 1966](#)]
- Connected to **boosting**, **minimax games**, **online convex optimization**, **bandits**, **calibration**, **fairness...**
- Used as fundamental building blocks in many classical algorithms / problems
 - see survey by [Arora, Hazan, Kale](#)
 - see work by [Anupam Gupta](#)

Many Applications to Modern ML

Define “Problem”, “Experts”, define “Loss Function”

Problem	Experts	Loss Function
LLM Routing	Llama, GPT, Claude, DeepSeek, etc...	Measures quality / speed / \$ cost
Hyperparameter tuning / AutoML (e.g, Hyperband)	Depth / Width / Choice of activation function	Val Loss curve improvements
...	...	...

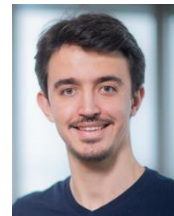
Online selection of Drafters in Speculative Decoding for LLM Acceleration. (Second part of this talk!)

Outline

1. “A second order regret bound for NormalHedge”

<https://arxiv.org/abs/2602.08151>

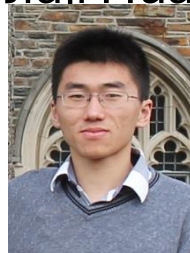
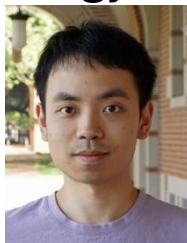
w. Yoav Freund, Nick Harvey, Victor Portella and Yabing Qi



2. “Not-A-Bandit”: An Application of NormalHedge to faster LLM inference.

<https://arxiv.org/abs/2510.20064> [ICLR’26]

w. Hongyi Liu, Jiaii Huang, Zhen Jia, Youngsuk Park



Learning from Expert Advice

Given number of experts N and a constant $B > 0$

For each iteration $j = 1, 2, 3, \dots$

1. Player chooses the weights as a probability vector $\mathbf{p}_j \in \Delta^{N-1}$.
2. Nature reveals a loss vector $\ell_j \in \mathbb{R}^N$ satisfying that $\max_{i, i' \in [N]} |\ell_{j,i} - \ell_{j,i'}| \leq B$.
3. Player incurs a loss $\langle \mathbf{p}_j, \ell_j \rangle = \mathbb{E}_{i \sim \mathbf{p}_j} [\ell_{j,i}]$.

- Instantaneous regret (vector):

$$\Delta \mathbf{x}_j = \langle \mathbf{p}_j, \ell_j \rangle \mathbf{1} - \ell_j$$

- Cumulative regret (vector) at T_{th} iteration

$$\text{Regret}_i(T) = \mathbf{x}_{T,i} = \sum_{j=1}^T \Delta \mathbf{x}_{j,i}$$

Modern adaptive online learning focuses on *adapting to* "easy instances"

- Worst case regret bound

$$\text{Regret}_i(T) = \Theta(\sqrt{T \log N})$$

- **(Replace T)** First and Second order regret bound

$$L_T(i) = \sum_{j=1}^T |\Delta x_{j,i}|$$

$$V_T = \sum_{j=1}^T \mathbb{E}_{i \sim \mathbf{p}_j} [\Delta x_{j,i}^2]$$

First order or "Small loss" regret

Second order or "Small Variance" regret

- **(Replace log N)** Quantile regret

$$\text{Regret}_\epsilon(T) = O(\sqrt{T \log(1/\epsilon)})$$

NormalHedge ([Chaudhuri, Freund, Hsu 2009](#))

Also, ([Chernov and Vovk, 2010](#)) ([Luo and Schapire, 2015](#))

“Variance across action” vs “Variance across time”

(Cesa-Bianchi, Mansour, Stoltz, 2007)

- Variance across actions Example: AdaHedge: (de Rooij et al. 2013)

$$V_T = \sum_{j=1}^T \text{Var}_{i \sim p_j} [\ell_{j,i}] = \sum_{j=1}^T \mathbb{E}_{i \sim p_j} [\Delta x_{j,i}^2]$$

- Variance across time (per-expert)

$$V_T(i) = \sum_{j=1}^T \Delta x_{j,i}^2$$

Stronger but algorithm-dependent

Weaker but more interpretable

Example: Adapt-ML-Prod: (Gaillard et al, 2014)

Open problem from COLT'16



- Are there algorithms that *simultaneously* achieve **quantile regrets** and **second-order regret**

$$\text{Regret}_\epsilon(T) = O(\sqrt{V_T \log(1/\epsilon)})$$

where

$$V_T = \sum_{j=1}^T \text{Var}_{i \sim p_j} [\ell_{j,i}]$$



“My conjecture is that **NormalHedge** has a regret bound of this form”

“I am offering **a prize of \$500** to anyone who gives an algorithm with a proven bound of this form, or, alternatively, a proof that such a bound is impossible.”

This is a reasonable bet because NormalHedge (Chaudhuri, Freund and Hsu, 2009) is known to perform exceptionally well in practice.

- “AdaHedge” co-author: Peter D. Grünwald

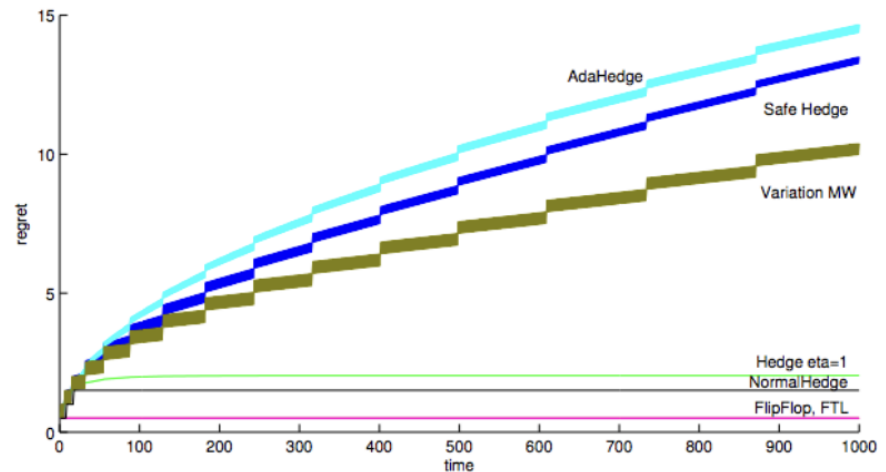
A.M. van Eden

Breaking NormalHedge

Bachelor Thesis

Thesis Supervisor: Prof. Dr. P. D. Grünwald

Date of Bachelorexamination: 14th August 2014



5.4 Conclusion

Sadly the goal of this paper has not been achieved. We did not find clear weaknesses in the NormalHedge algorithm in comparison to AdaHedge and FlipFlop. We were not able to break it. There are two reasons why this could be the case.

Notable existing results on adaptive quantile bounds

- AdaNormalHedge (Luo and Schapire, 2015)

$$\sqrt{\mathbb{E}_{i \sim \text{Top}_\epsilon} [L_T(i)] \log(1/\epsilon)}$$

(First order!)

- Squint (Koolen and van Erven, 2015)

$$\sqrt{\mathbb{E}_{i \sim \text{Top}_\epsilon} [V_T(i)] \log(1/\epsilon)}$$

(Variance across time,
rather than across action!)

- FTRL with sqrt-entropy (Negrea et al, 2021)

$$\log(1/\epsilon) \sqrt{V_T^{\text{avg}}}$$

(Quantile term not in sqrt)

2021: Yoav's conjectured bound is impossible...



[Submitted on 25 Oct 2021]

The Pareto Frontier of model selection for general Contextual Bandits

Teodor V. Marinov, Julian Zimmert



$$O(\sqrt{V_T \log(1/\epsilon)})$$

- Adaptive bounds like this is not possible in general.
- If an Expert algorithm is *adaptive* to both V_T and ϵ , then it implies model-selection problem for contextual bandits, which they prove to be impossible.

End of story?

2026: Eureka!

[Submitted on 8 Feb 2026]

A second order regret bound for NormalHedge

Yoav Freund, Nicholas J. A. Harvey, Victor S. Portella, Yabing Qi, Yu-Xiang Wang

Theorem 6 (NormalHedge.BH) For $t_0 = C B^2 \log N$

$$\text{Regret}_\varepsilon(T) \leq \sqrt{(t_0 + 2V_T) (\log(t_0 + 2V_T) + 2 \log(1/\varepsilon))}$$

$$\text{where } V_T = \sum_{j=1}^T \mathbb{E}_{i \sim \mathbf{q}_j} [\Delta x_{j,i}^2]$$

- $O(\log N)$ additive “Burn-In” term.
- A slightly different second-moment. Algorithm output $(\mathbf{p}_j, \mathbf{q}_j)$
- Matching lower bound $\Omega(\sqrt{V_T^q \log(1/\epsilon)})$ (up to a $\log(V_T)$ factor)

(*Haipeng Luo posted a variant of “squint” recently that got similar bound.)

What's the algorithm?

Constant Potential is all you need

Define a total potential function

$$\Phi(\mathbf{x}, t) = \sum_{i=1}^N \phi(x_i, t).$$

CP algorithm(ϕ): Initialization: $\mathbf{x} = 0, t = t_0, V = 0$

1. Learner plays $p_i \propto \phi'(x_i, t)$ --- First derivative of potential.
2. Nature reveals losses ℓ and compute $\Delta x = \ell - \langle p, \ell \rangle$
3. Player decide how much to increase Δt such that
 $\Phi(x + \Delta x, t + \Delta t) = \Phi(x, t)$ --- Total regret remains constant.
4. $V \leftarrow V + \mathbb{E}_{i \sim q} [\Delta x_i^2]$ where $q_i \propto \phi''(x_i, t)$ --- second derivative
5. Update $x \leftarrow x + \Delta x, t \leftarrow t + \Delta t$

*Algorithm simplified for presentation: projection to domain $\mathcal{D}$ omitted.

Many questions to address

- Which potential function to use?
- How different is q from p ? In what sense this second-moment w.r.t. q is natural?
- How constant potential helps bounding regret?

A “good” potential function satisfies 7 properties.

Definition 1. Let $\mathcal{D} = \mathbb{R}$ or $\mathcal{D} = [y_0, \infty)$. A function $\phi : \mathbb{R} \times [0, \infty) \rightarrow \mathbb{R}$ is a **good potential function** on $\mathcal{D}$ if it is (1) jointly strictly convex, (2) three-times differentiable, and satisfies: (3) $\partial_y \phi \geq 0$; (4) $\partial_t \phi \leq 0$; (5) $\lim_{t \rightarrow \infty} \phi(y, t) = \inf_{y, t} \phi(y, t) \geq 0$, and (6) $\phi(\Pi_{\mathcal{D}}(\mathbf{x}), t) \leq \phi(\mathbf{x}, t)$ for all $\mathbf{x} \in \mathbb{R}$. Furthermore, we assume (7) the backwards heat equation $\partial_t \phi = -\frac{1}{2} \partial_{yy}^2 \phi$.

- Backwards Heat equation

$$\partial_t \phi(x, t) = -\frac{1}{2} \partial_{xx}^2 \phi(x, t)$$

- Our choices of the potential

Hedge / Multiplicative Weights

$$\phi_{\text{exp}}(y, t) = \exp(\sqrt{2\eta} y - \eta^2 t),$$

$$\mathcal{D} = \mathbb{R};$$

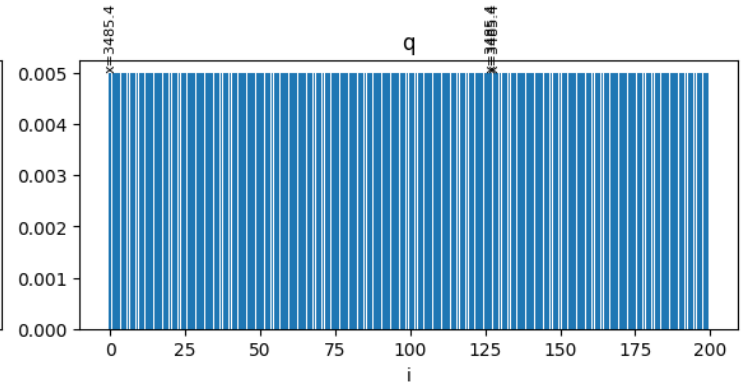
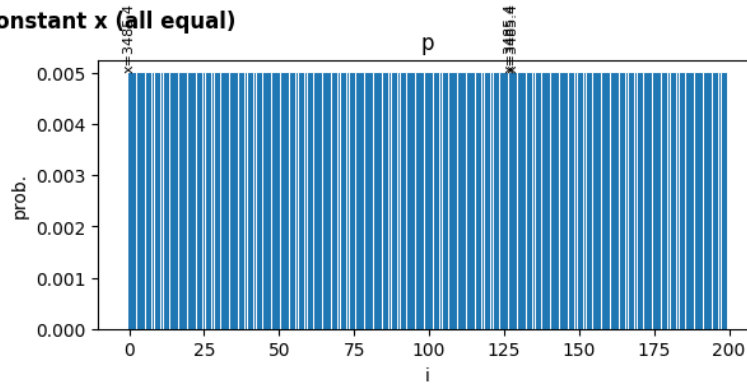
$$\phi_{\text{NH}}(y, t) = \boxed{t^{-1/2}} \exp(y^2 / 2t),$$

$$\mathcal{D} = [0, \infty).$$

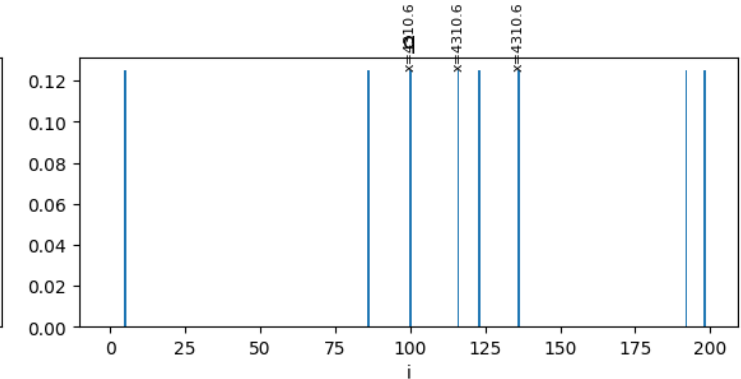
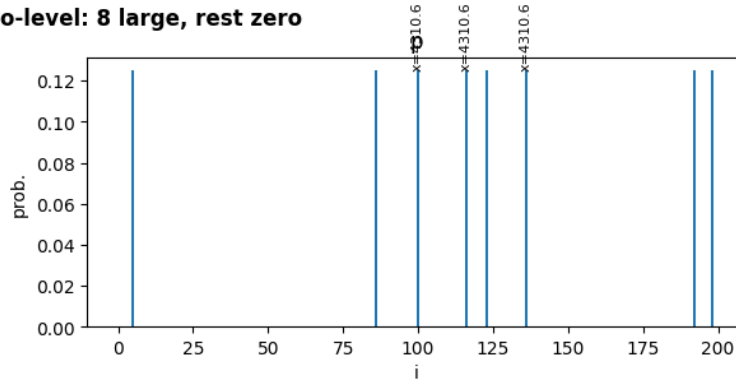
NormalHedge.BH

q vs p in NormalHedge by examples.

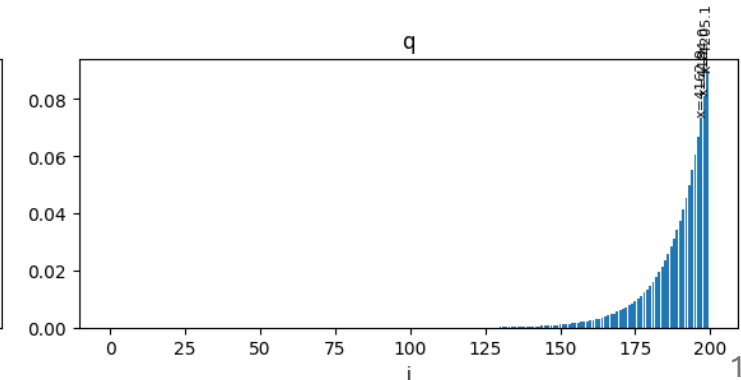
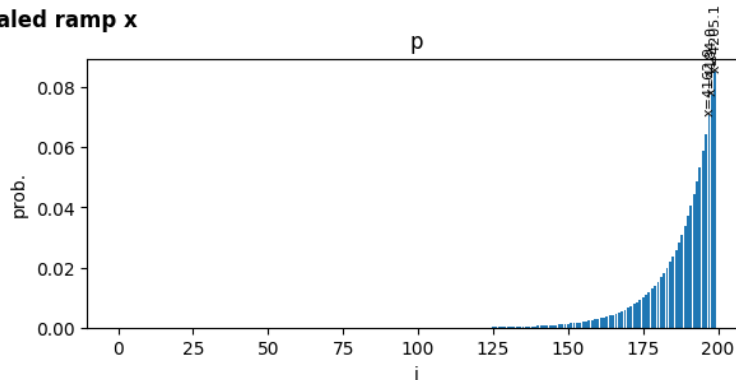
Constant x (all equal)



Two-level: 8 large, rest zero



Scaled ramp x



Constant potential provides *iteration-free* regret bound – one that depends on the **continuous time t**

- Since the total potential is constant...

$$\Phi(\mathbf{x}, t) = \Phi(\mathbf{x}_0, t_0) = \frac{N}{\sqrt{t_0}}$$

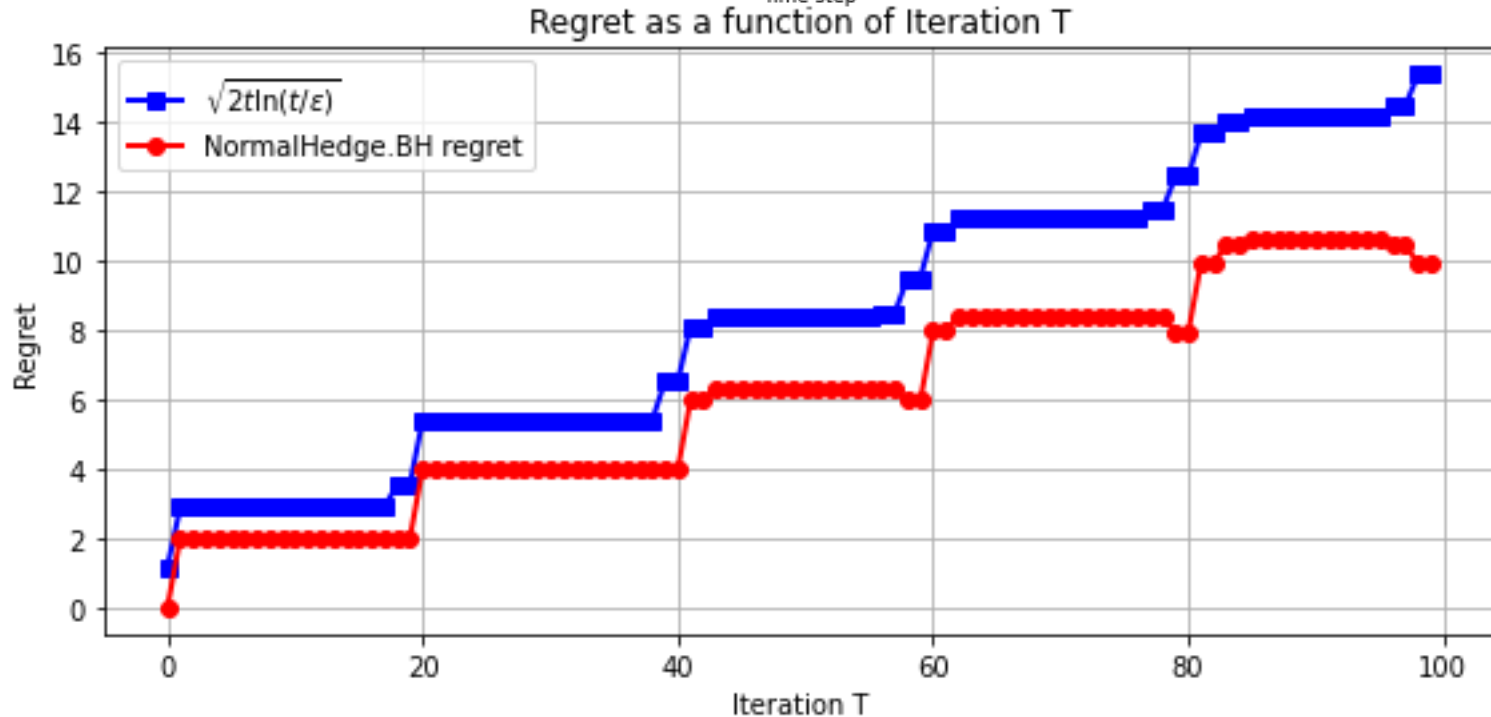
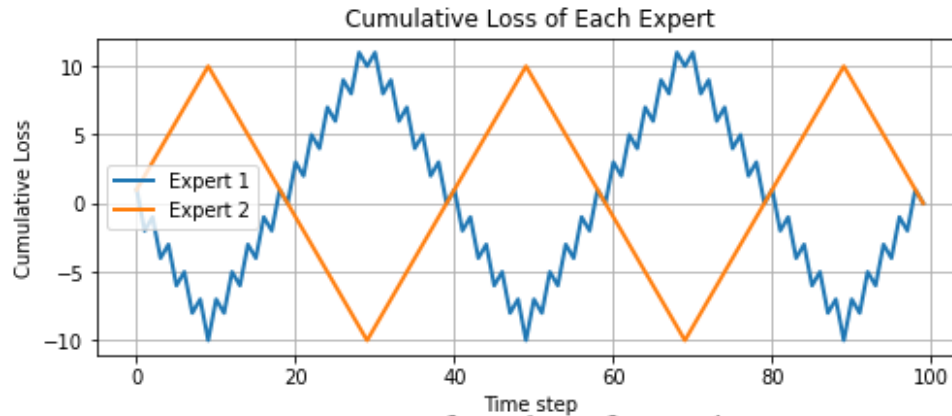
$$N\epsilon \cdot \frac{1}{\sqrt{t}} \exp\left(\frac{x_{(N\epsilon)}^2}{2t}\right) = N\epsilon \cdot \phi(x_{(N\epsilon)}, t) \leq \sum_{i=1}^N \phi(x_{(i)}, t) = \Phi(\mathbf{x}, t) = \Phi(\mathbf{x}_0, t_0) = \frac{N}{\sqrt{t_0}}$$

Order statistics (decreasing)

- Taking log on both sides

$$\text{Regret}_\epsilon = x_{(N\epsilon)} \leq \sqrt{t(\log(t/t_0) + 2\log(1/\epsilon))}.$$

Parameter t only increases near “leader switches”.



Regret vs continuous time t always looks like $\sqrt{t}$



Adaptive non-uniform discretization of time t
Only “paying” when the algorithm incurs regret.

It remains to bound t with V_T

$$\text{Regret}_\epsilon = x_{(N\epsilon)} \leq \sqrt{t(\log(t/t_0) + 2\log(1/\epsilon))}.$$

Theorem 6 (NormalHedge.BH) For $t_0 = C B^2 \log N$

$$\text{Regret}_\epsilon(T) \leq \sqrt{(t_0 + 2V_T) (\log(t_0 + 2V_T) + 2\log(1/\epsilon))}$$

where
$$V_T = \sum_{j=1}^T \mathbb{E}_{i \sim \mathbf{q}_j} [\Delta x_{j,i}^2]$$

Continuous-Time Analysis for Intuition

- Discrete Ito's formula

$$\underbrace{\Phi(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t) - \Phi(\mathbf{x}, t)}_{= 0} \approx \underbrace{\langle \nabla_{\mathbf{x}} \Phi(\mathbf{x}, t), \Delta\mathbf{x} \rangle}_{(*)} + \underbrace{\partial_t \Phi(\mathbf{x}, t) \Delta t}_{(**)} + \underbrace{\frac{1}{2} \sum_i \partial_{yy}^2 \phi(x_i, t) \Delta x_i^2}_{\frac{1}{2} (\Delta\mathbf{x})^T \nabla_{\mathbf{x}}^2 \Phi(\mathbf{x}, t) \Delta\mathbf{x}}.$$

(Constant Potential)

$$(*) \propto \langle \mathbf{p}, \langle \mathbf{p}, \ell \rangle \mathbf{1} - \ell \rangle = \langle \mathbf{p}, \ell \rangle - \langle \mathbf{p}, \ell \rangle = 0$$

By BHE: $(**) = \sum_i \partial_t \phi(x_i, t) = -\frac{1}{2} \sum_i \partial_{yy}^2 \phi(x_i, t) \Delta t$

$$\Delta t \approx \frac{\sum_i \partial_{yy}^2 \phi(x_i, t) \Delta x_i^2}{\sum_i \partial_{yy}^2 \phi(x_i, t)} = \mathbb{E}_{i \sim \mathbf{q}}[\Delta x_i^2]$$

Technical challenges start to emerge when we need to rigorously prove

$$\Delta t \approx \frac{\sum_i \partial_{yy}^2 \phi(x_i, t) \Delta x_i^2}{\sum_i \partial_{yy}^2 \phi(x_i, t)} = \mathbb{E}_{i \sim q}[\Delta x_i^2]$$

- Main technical innovations:
 - A local self-concordant analysis
 - More convenient to analyze $\log \Phi$ rather than Φ
 - Self-bounding tricks for breaking “circular logic”
- Clean result that is asymptotically exact

Read the paper!

Outline

1. “A second order regret bound for NormalHedge”
<https://arxiv.org/abs/2602.08151>

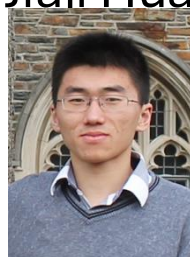
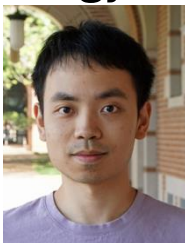
w. Yoav Freund, Nick Harvey, Victor Portella and Yabing Qi



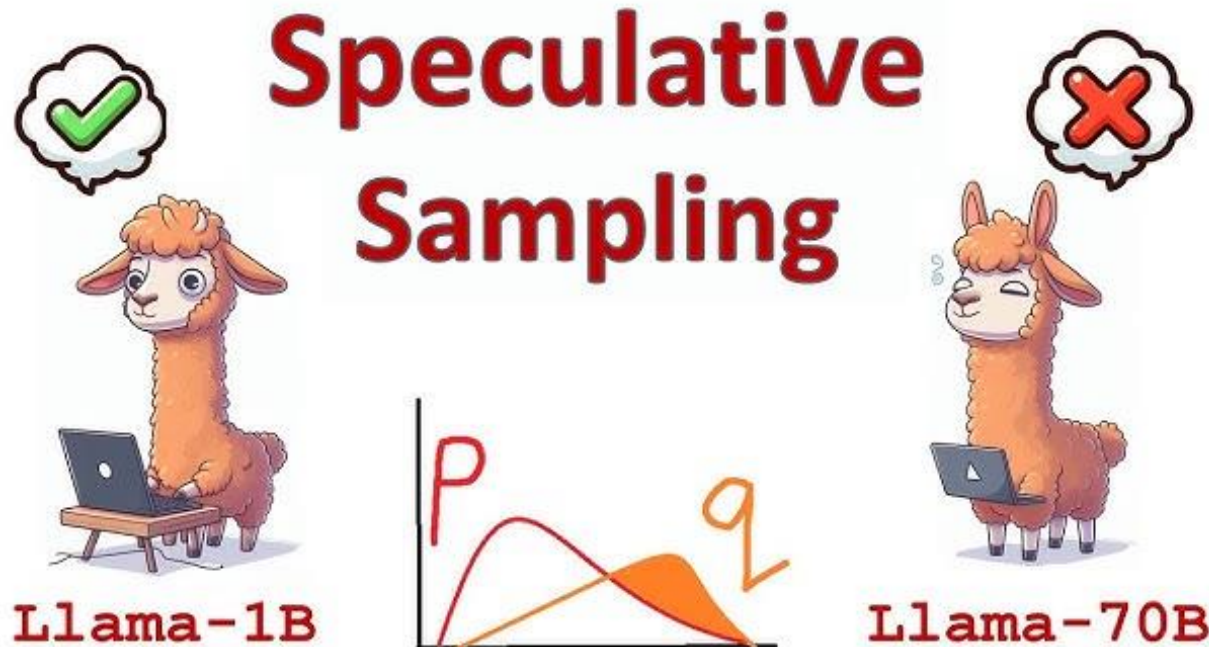
2. “Not-A-Bandit”: An Application of NormalHedge to faster LLM inference.

<https://arxiv.org/abs/2510.20064> [ICLR’26]

w. Hongyi Liu, Jiaii Huang, Zhen Jia, Youngsuk Park



Speculative Decoding uses a small LM to guess what a large LM would've generated.



- Draft with small LM
- Verify with Large LM (Rejection sampling-like) in parallel

Lower-latency. Faster Throughput.

How speculative decoding works?

- Drafting

HedgeSpec [chooses with no remorse at all]

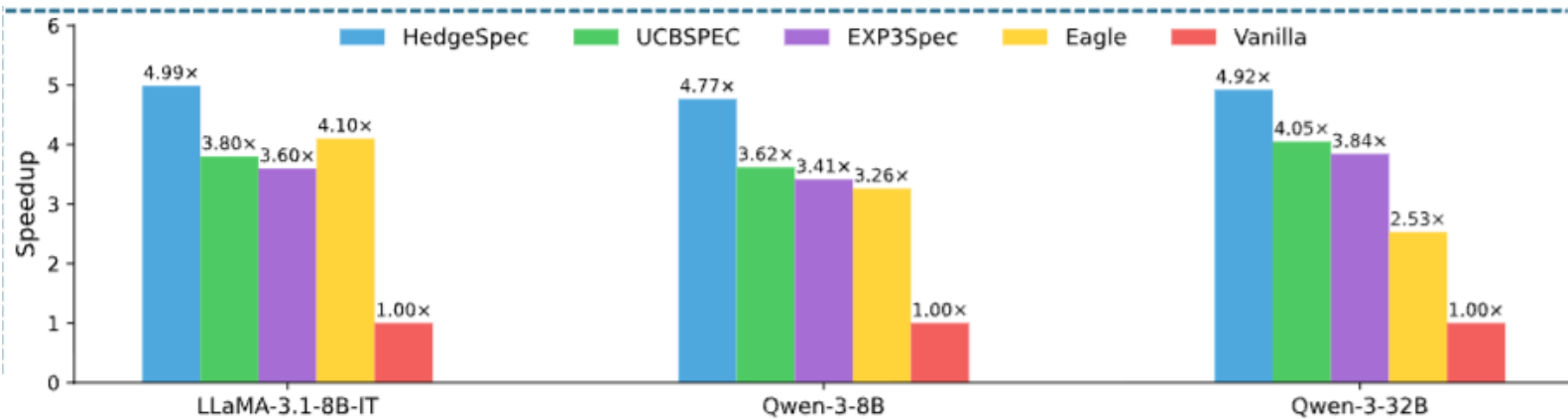
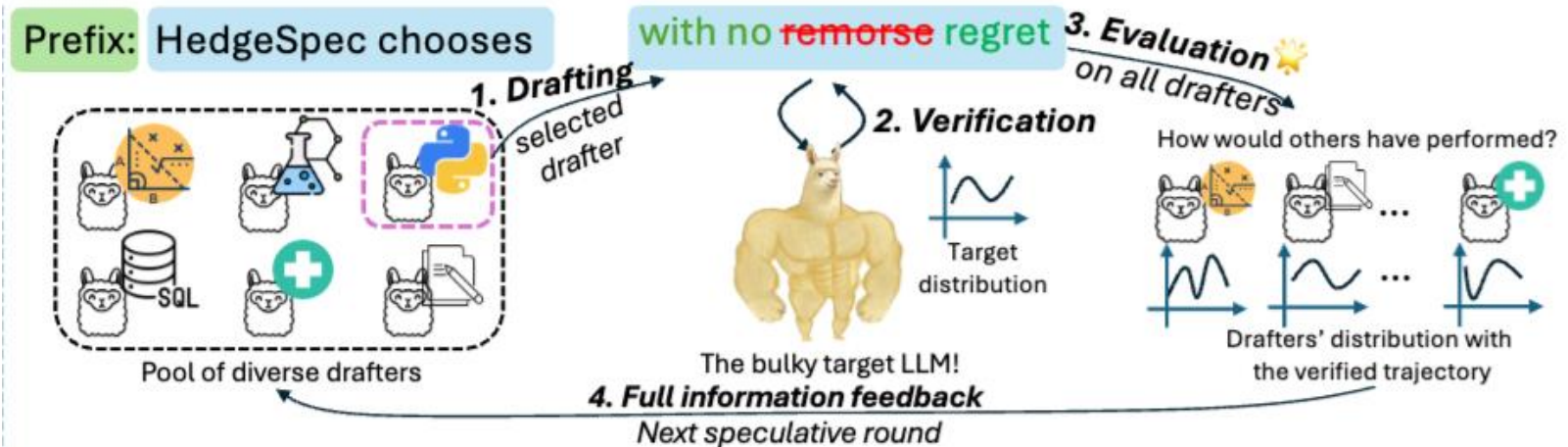
- Verify **in parallel** (rejection sampling style)

HedgeSpec ^{regrets} [chooses with no ~~remorse~~ ...]

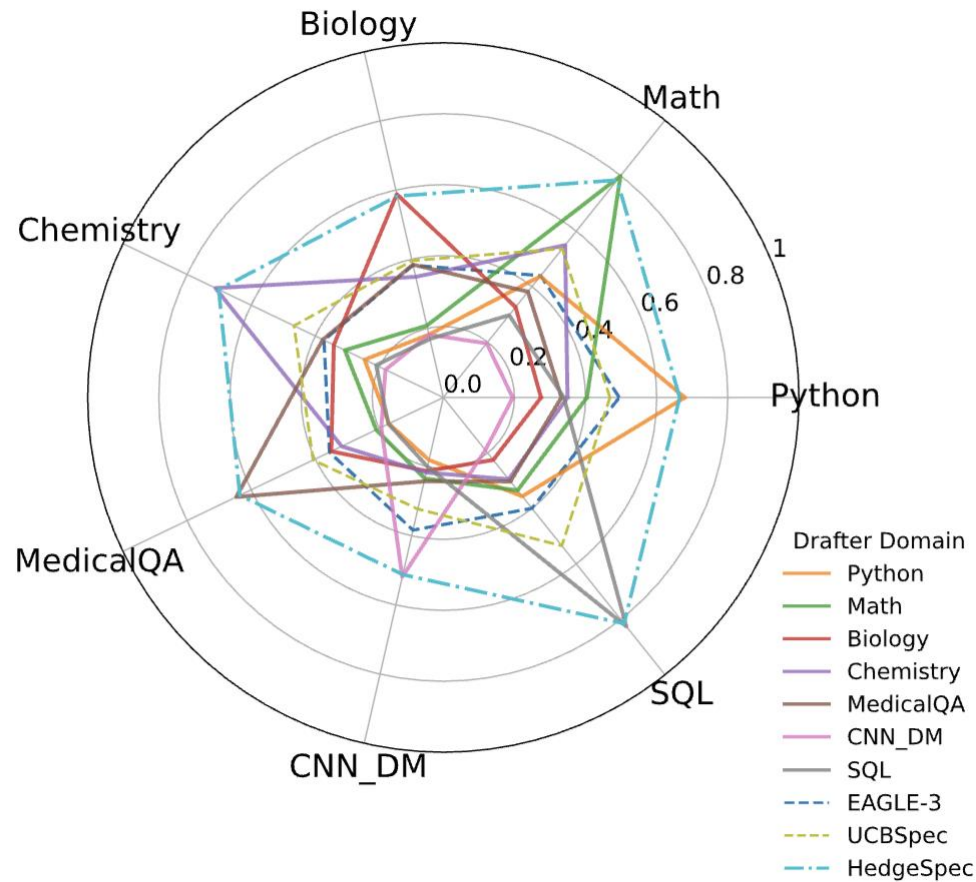
- 4x speed up.

- 4 is “accepted tokens length”.
- MAT: mean accepted tokens (per round)

Online Draft-model selection with NormalHedge

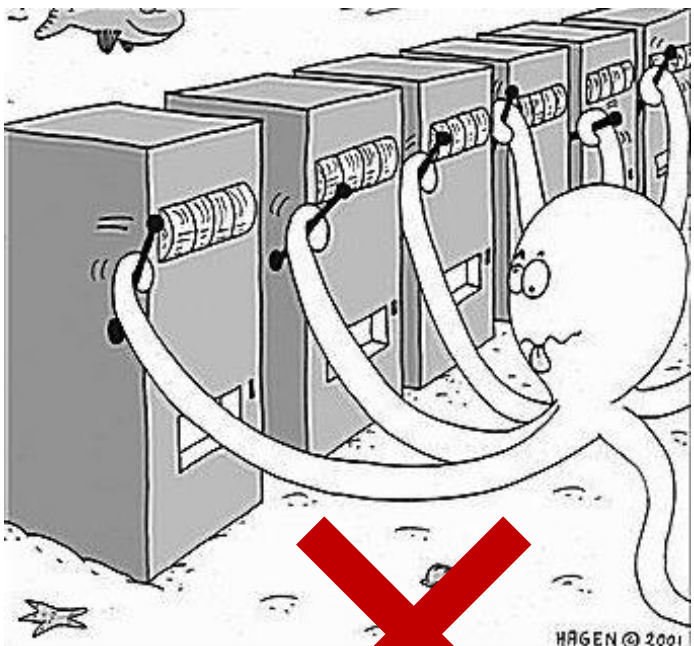


We can now complete with the best drafter for each “prompt query” separately.

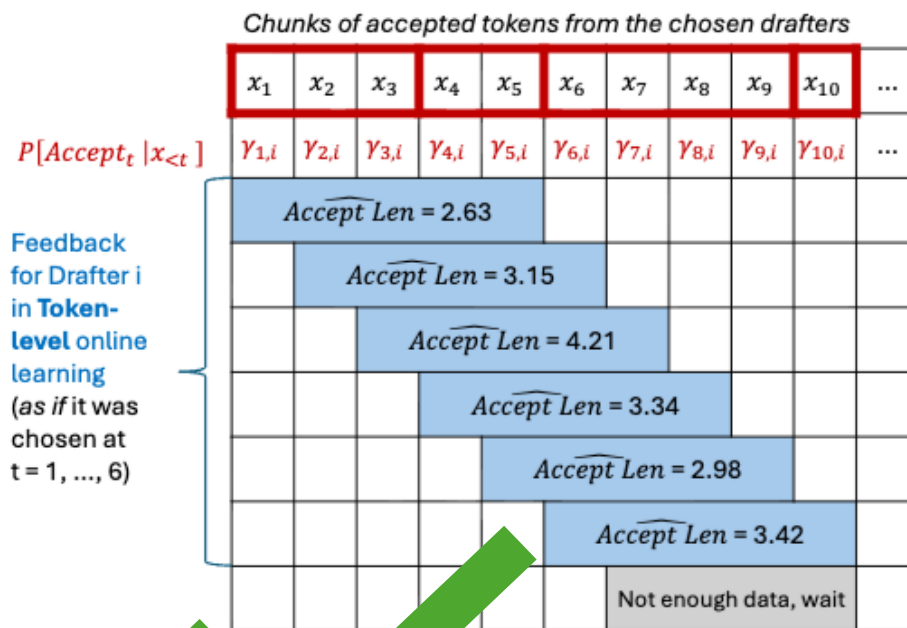


Existing work runs bandits, but we show that we can get away with “Full-information” feedback.

MetaSD (Kim et al 2024)
BanditSpec (Hou et al, 2025)

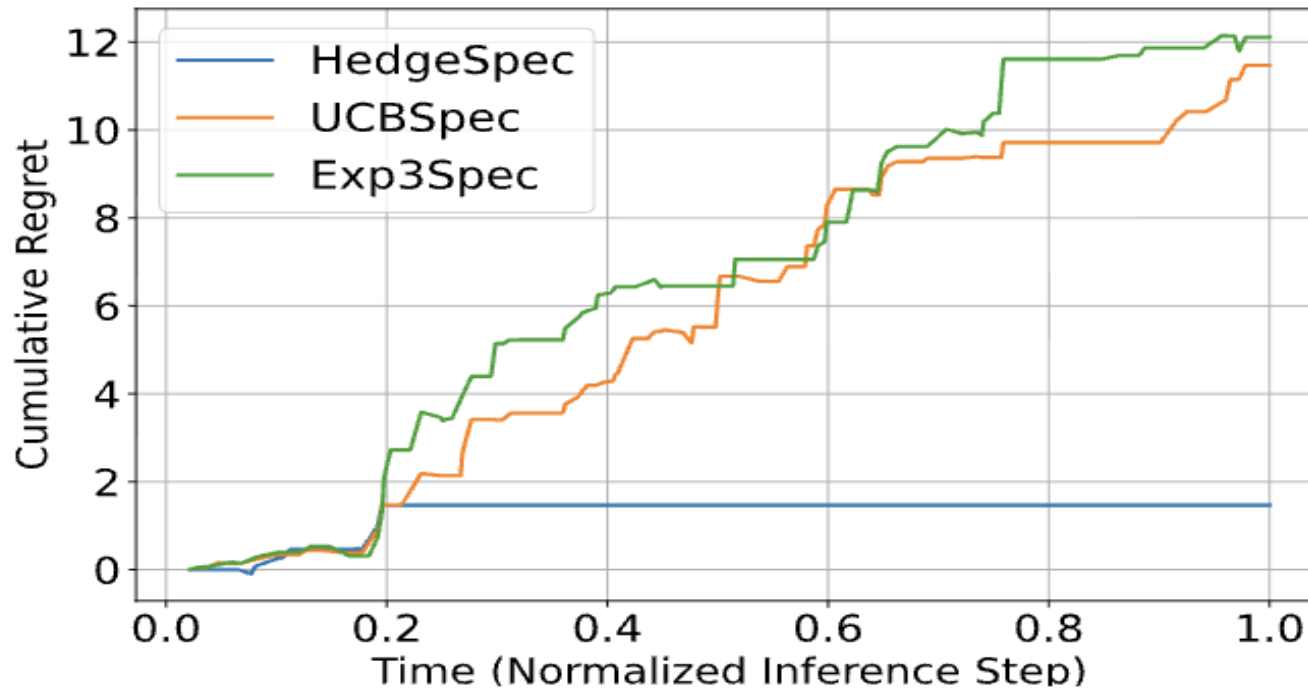


Our method:
“Experience-Replay” +
Learning with Delayed feedback.



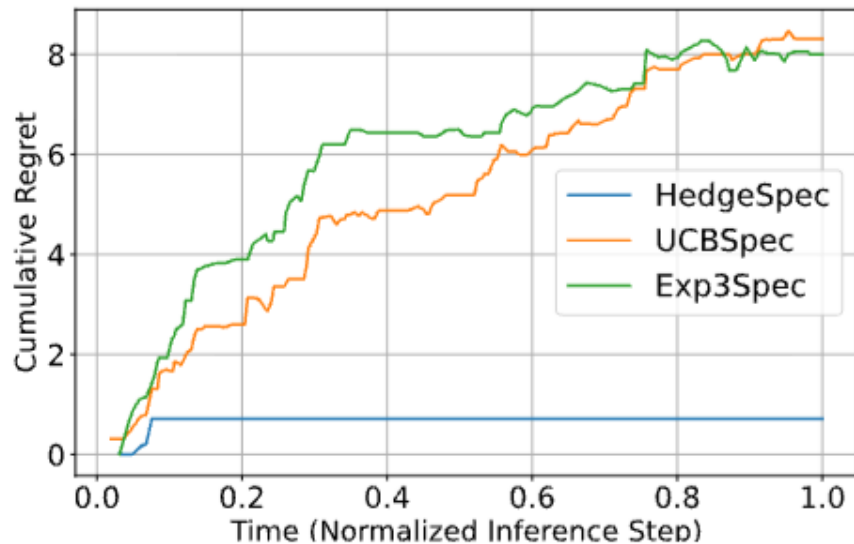
Theorem: Let MAT be “Mean Accepted Tokens”, i.e., the number of accepted tokens per speculative round. For N drafters, max-length K , response length T . HedgeSpec enjoys

$$\text{MAT}(\text{HedgeSpec}) \geq \text{MAT}(\text{Best Drafter}) - O\left(\sqrt{\frac{K \log N}{T}}\right)$$

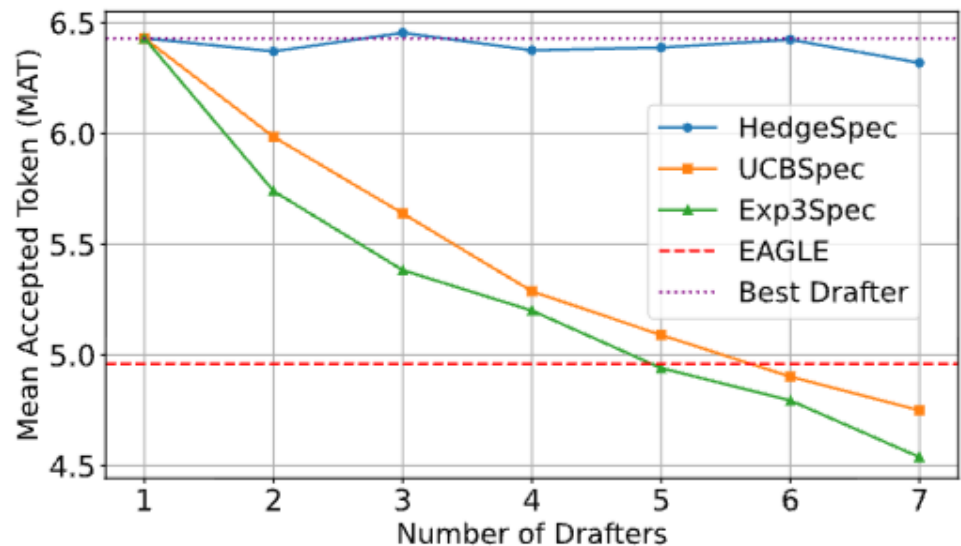


a. Cumulative Regret

NormalHedge learns so quickly that it almost has no regret at all.



a. Cumulative Regret



b. MAT with increasing number of drafters

NormalHedge performs better than other Hedge algorithms that come with stronger adaptive guarantees

Table 1: Llama-3.1-8B-IT Mean Accepted Tokens (MAT) across datasets. Bold indicates highest performance per column.

Datasets	Python	Math	Biology	Chemistry	MedQA	CNN_DM	SQL
NormalHedge	7.69	7.69	7.18	7.10	6.47	5.88	8.06
Standard Hedge	6.90	6.67	6.53	6.19	5.38	4.68	7.05
AdaNormalHedge	7.36	7.30	7.05	6.66	5.85	5.40	7.71
Adapt-ML-Prod	6.59	6.38	6.24	5.78	4.93	4.29	6.63

This partially piqued my interest in understanding NormalHedge.

Reiterate the main results

Part 1: progress on a decade-old open problem:

- NormalHedge.BH enjoys second-order quantile regret $O(\sqrt{V_T \log(V_T/\epsilon)})$ when $V_T > \log N$
- Clean / Elegant potential-based algorithm.
- New insight: Continuous-time view, BHE in potential
- New local self-concordance analysis.

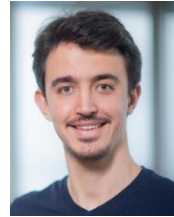
Part 2: Normal-Hedge is practical (more practical than other hedge algos) in LLM practice!

Open problems

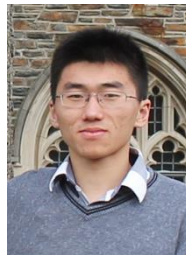
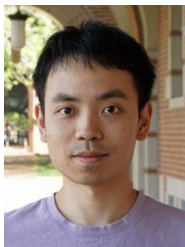
1. Is $O(B^2 \log N)$ in “burn-in” needed for adaptivity?
 - [Marinov and Zimmert](#) seems to imply it's needed for second-order bounds.
 - AdaNormalHedge ([Luo and Schapire](#)) does not depend on N (though only first-order bound)
2. Is it possible to shave $\log V_T$ from the bound while not adding dependence in T ?
3. Are there other interesting potential functions satisfying BHE to use for CP?
4. What other modern LLM problems we can apply Normal-Hedge to?

Thank you for your attention!

- “A second order regret bound for NormalHedge”
<https://arxiv.org/abs/2602.08151>



- “Not-a-Bandit: Provably No-Regret Drafter Selection in Speculative Decoding for LLMs”: <https://arxiv.org/abs/2510.20064>



Supplementary slides

NormalHedge.BH assigns 0 to any expert performing worse than the algorithm.

$$p_i \propto \partial_y \phi([x_i]_+, t) = \frac{[x_i]_+}{t} \phi([x_i]_+, t)$$

$$q_i \propto \partial_{yy}^2 \phi([x_i]_+, t) = \left(\frac{1}{t} + \frac{[x_i]_+^2}{t^2} \right) \phi([x_i]_+, t)$$

A small improvement in the bound that gives **provable benefit from “sparsity”**:

$$V_T = \sum_{j=1}^T \mathbb{E}_{i \sim \mathbf{q}_j} [\Delta x_{j,i}^2 \mathbf{1}(p_{j,i} \neq 0)]$$

Constant Potential algorithm: details fully written out.

Define a total potential function

$$\Phi(\mathbf{x}, t) = \sum_{i=1}^N \phi(x_i, t).$$

CP($\phi, \mathcal{D}$) algorithm

Initialize $t = t_0$, $x_{0,i} = 0 \forall i \in [N]$, $V_0 = 0$. Let $\Pi_{\mathcal{D}}$ be the projection to $\mathcal{D}$, and $\tilde{\mathbf{x}}_0 \leftarrow \Pi_{\mathcal{D}}(\mathbf{x}_0)$.

For $j = 1, \dots, T$

1. The algorithm plays probability p_j where $p_{j,i} \propto \frac{\partial}{\partial x} \phi(\tilde{x}_{j-1,i}, t_{j-1})$ for $i \in [N]$.
2. Nature reveals loss $\ell_{j,i}$ for every expert $i \in [N]$.
3. Algorithm incurs instantaneous regret $\Delta x_{j,i} := \mathbb{E}_{k \sim p_j}[\ell_{j,k}] - \ell_{j,i}$ for $i \in [N]$.
4. Algorithm updates $\mathbf{x}_j \leftarrow \mathbf{x}_{j-1} + \Delta \mathbf{x}_j$; $\tilde{\mathbf{x}}_j \leftarrow \Pi_{\mathcal{D}}(\mathbf{x}_j)$.
5. Algorithm finds $\Delta t_j \geq 0$ s.t. $\Phi(\tilde{\mathbf{x}}_j, t_{j-1} + \Delta t_j) = \Phi(\tilde{\mathbf{x}}_{j-1}, t_{j-1})$ and sets $t_j \leftarrow t_{j-1} + \Delta t_j$.
6. Algorithm sets $V_j \leftarrow V_{j-1} + \mathbb{E}_{i \sim q_j} [\Delta x_{j,i}^2]$ where $q_j \in \Delta^{N-1}$ is s.t. $q_{j,i} \propto \partial_{xx} \phi(\tilde{x}_{j-1,i}, t_{j-1})$

Return V_T .

Figure 1: The Constant Potential Algorithm

The pursuit on more adaptive learners

Work	Cumulative variance	Regret bound	Adaptive Minimax?
Hedge [Cesa-Bianchi et al., 2007]	$V_T^{\text{avg}} := \sum_{j=1}^T \mathbb{E}_{i \sim p_j} [(\Delta x_{j,i})^2]$	$\sqrt{V_T^{\text{avg}} \log N}$	No
Hedge [Cesa-Bianchi et al., 2007]	$V_T(i) := \sum_{j=1}^T (\Delta x_{j,i})^2$	$\sqrt{V_T(i) \log N}$	No
NormalHedge [Chaudhuri et al., 2009]	T	$\sqrt{T \log(1/\epsilon)}$	Yes
Squint [Koolen and van Erven, 2015]	$V_T(\epsilon) := \mathbb{E}_{i \sim \text{Unif}(\text{Top}_\epsilon)} [V_T(i)]$	$\sqrt{V_T(\epsilon) \log(1/\epsilon)}$	Yes
Ada-Normal-Hedge [Luo and Schapire, 2015]	$S_T(\epsilon) := \mathbb{E}_{i \sim \text{Unif}(\text{Top}_\epsilon)} [\sum_{j=1}^T \Delta x_{j,i}]$	$\sqrt{S_T(\epsilon) \log(1/\epsilon)}$	Yes
AdaHedge [De Rooij et al., 2014]	$V_T^{\text{avg}} := \sum_{j=1}^T \mathbb{E}_{i \sim p_j} [(\Delta x_{j,i})^2]$	$\sqrt{V_T^{\text{avg}} \log N}$	Yes
Adapt-ML-Prod [Gaillard et al., 2014]	$V_T(i)$	$\sqrt{V_T(i) \log N}$	Yes
FTRL [Negrea et al., 2021]	$V_T^{\text{avg},q} := \sum_{j=1}^T \mathbb{E}_{i \sim q_j} [(\Delta x_{j,i})^2]$	$\log(1/\epsilon) \sqrt{V_T^{\text{avg},q}}$	Yes
Hedge (Theorem 5 this paper)	V_T^{avg}	$\sqrt{V_T^{\text{avg}} \log(1/\epsilon)}$	No
NormalHedge (Theorem 6 this paper)	$V_T^{\text{avg},q}$	$\sqrt{V_T^{\text{avg},q} \log(1/\epsilon)}$	Yes
Lower bound (Theorem 7 this paper)	$V_T^{\text{avg},q}$ for any $q^{1:T}$	$\sqrt{V_T^{\text{avg},q} \log(1/\epsilon)}$	not applicable

Table 1: Summary of instance-dependent regret bounds. For those that depends on priors, e.g., [Koolen and van Erven \[2015\]](#), we choose the prior to be uniform, so as to be comparable to quantile regret bounds. For clarity, we omit multiplicative $\log(V_T)$ factors and additive $\text{poly}(\log(N))$ factors. Also, note that $V_T^{\text{avg},q}$ in [Negrea et al. \[2021\]](#) is different from the $V_T^{\text{avg},q}$ in our Theorem 6 due to different choices of $q^{1:T}$, both are natural for the algorithm under consideration.

Comparison with: Haipeng Luo's Squint Variant: <https://arxiv.org/abs/2603.03409>

- Our regret bound for NormalHedge.BH

$$\text{Regret} \asymp \sqrt{(\log N + V_T) \left(\log V_T^q + \log \left(\frac{1}{\epsilon} \right) \right)}$$

- Haipeng's bound for Squint-variant

$$\text{Regret} \asymp \log \log(T) + \sqrt{V_T^q \log \left(\frac{1}{\epsilon} \right)}$$

q is an one-step-look-forward version.

Comparison to trained-static router in out-of-distribution case.

- Naïve baseline:
 - Use prompt features to predict the best-drafter.
- We are not using any features.
- Learn from scratch for each query separately.

	Llama-3.1-8B-IT		Qwen-3-8B	
Math	MAT	Token/s	MAT	Token/s
Eagle	5.83	78.05	4.63	54.49
Static Router	5.28	70.40	4.95	55.18
HedgeSpec	6.94	90.15	7.12	77.45
MedQA	MAT	Token/s	MAT	Token/s
Eagle	5.23	67.67	3.91	43.54
Static Router	3.03	40.83	2.49	28.18
HedgeSpec	5.56	70.38	6.04	65.99

Table 4: HedgeSpec vs. offline router under O.O.D (**Bold**=best). The router suffers and mis-routes requests to sub-optimal drafters. HedgeSpec adapts through runtime feedback, and remains robust.