Scalable Generalized Linear Bandits: Online Computation and Hashing

Kwang-Sung Jun
UW-Madison
kjun@discovery.wisc.edu

Aniruddha Bhargava
UW-Madison
aniruddha@wisc.edu

Robert Nowak
UW-Madison
rdnowak@wisc.edu

Rebecca Willett
UW-Madison
willett@discovery.wisc.edu

Abstract

Generalized Linear Bandits (GLBs), a natural extension of the stochastic linear bandits, has been popular and successful in recent years. However, existing GLBs scale poorly with the number of rounds and the number of arms, limiting their utility in practice. This paper proposes new, scalable solutions to the GLB problem in two respects. First, unlike existing GLBs, whose per-time-step space and time complexity grow at least linearly with time t, we propose a new algorithm that performs online computations to enjoy a constant space and time complexity. At its heart is a novel Generalized Linear extension of the Online-to-confidence-set Conversion (GLOC method) that takes *any* online learning algorithm and turns it into a GLB algorithm. As a special case, we apply GLOC to the online Newton step algorithm, which results in a low-regret GLB algorithm with much lower time and memory complexity than prior work. Second, for the case where the number N of arms is very large, we propose new algorithms in which each next arm is selected via an inner product search. Such methods can be implemented via hashing algorithms (i.e., "hash-amenable") and result in a time complexity sublinear in N. While a Thompson sampling extension of GLOC is hash-amenable, its regret bound for d-dimensional arm sets scales with $d^{3/2}$, whereas GLOC's regret bound scales with d. Towards closing this gap, we propose a new hash-amenable algorithm whose regret bound scales with $d^{5/4}$. Finally, we propose a fast approximate hash-key computation (inner product) with a better accuracy than the state-of-the-art, which can be of independent interest. We conclude the paper with preliminary experimental results confirming the merits of our methods.

1 Introduction

This paper considers the problem of making generalized linear bandits (GLBs) scalable. In the stochastic GLB problem, a learner makes successive decisions to maximize her cumulative rewards. Specifically, at time t the learner observes a set of arms $\mathcal{X}_t \subseteq \mathbb{R}^d$. The learner then chooses an arm $\mathbf{x}_t \in \mathcal{X}_t$ and receives a stochastic reward y_t that is a noisy function of $\mathbf{x}_t$: $y_t = \mu(\mathbf{x}_t^\top \boldsymbol{\theta}^*) + \eta_t$, where $\boldsymbol{\theta}^* \in \mathbb{R}^d$ is unknown, $\mu : \mathbb{R} \to \mathbb{R}$ is a known nonlinear mapping, and $\eta_t \in \mathbb{R}$ is some zero-mean noise. This reward structure encompasses generalized linear models [29]; e.g., Bernoulli, Poisson, etc.

The key aspect of the bandit problem is that the learner does not know how much reward she would have received, had she chosen another arm. The estimation on $\boldsymbol{\theta}^*$ is thus biased by the history of the selected arms, and one needs to mix in exploratory arm selections to avoid ruling out the optimal arm. This is well-known as the exploration-exploitation dilemma. The performance of a learner is evaluated by its *regret* that measures how much cumulative reward she would have gained additionally if she had known the true $\boldsymbol{\theta}^*$. We provide backgrounds and formal definitions in Section 2.

A linear case of the problem above ($\mu(z) = z$) is called the (stochastic) linear bandit problem. Since the first formulation of the linear bandits [7], there has been a flurry of studies on the problem [11,

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

34, 1, 9, 5]. In an effort to generalize the restrictive linear rewards, Filippi et al. [15] propose the GLB problem and provide a low-regret algorithm, whose Thompson sampling version appears later in Abeille & Lazaric [3]. Li et al. [27] evaluates GLBs via extensive experiments where GLBs exhibit lower regrets than linear bandits for 0/1 rewards. Li et al. [28] achieves a smaller regret bound when the arm set $\mathcal{X}_t$ is finite, though with an impractical algorithm.

However, we claim that all existing GLB algorithms [15, 28] suffer from two scalability issues that limit their practical use: (i) under a large time horizon and (ii) under a large number N of arms.

First, existing GLBs require storing all the arms and rewards appeared so far, $\{(\mathbf{x}_s, y_s)\}_{s=1}^t$, so the space complexity grows linearly with t. Furthermore, they have to solve a batch optimization problem for the maximum likelihood estimation (MLE) at each time step t whose per-time-step time complexity grows at least linearly with t. While Zhang et al. [41] provide a solution whose space and time complexity do not grow over time, they consider a specific 0/1 reward with the logistic link function, and a generic solution for GLBs is not provided.

Second, existing GLBs have linear time complexities in N. This is impractical when N is very large, which is not uncommon in applications of GLBs such as online advertisements, recommendation systems, and interactive retrieval of images or documents [26, 27, 40, 21, 25] where arms are items in a very large database. Furthermore, the interactive nature of these systems requires prompt responses as users do not want to wait. This implies that the typical linear time in N is not tenable. Towards a *sublinear* time in N, locality sensitive hashings [18] or its extensions [35, 36, 30] are good candidates as they have been successful in fast similarity search and other machine learning problems like active learning [22], where the search time scales with N^ρ for some $\rho < 1$ (ρ is usually optimized and often ranges from 0.4 to 0.8 depending on the target search accuracy). Leveraging hashing in GLBs, however, relies critically on the objective function used for arm selections. The function must take a form that is readily optimized using *existing* hashing algorithms.[1] For example, algorithms whose objective function (a function of each arm $\mathbf{x} \in \mathcal{X}_t$) can be written as a distance or inner product between $\mathbf{x}$ and a query $\mathbf{q}$ are hash-amenable as there *exist* hashing methods for such functions.

To be scalable to a large time horizon, we propose a new algorithmic framework called Generalized Linear Online-to-confidence-set Conversion (GLOC) that takes in an online learning (OL) algorithm with a low 'OL' regret bound and turns it into a GLB algorithm with a low 'GLB' regret bound. The key tool is a novel generalization of the online-to-confidence-set conversion technique used in [2] (also similar to [14, 10, 16, 41]). This allows us to construct a confidence set for $\boldsymbol{\theta}^*$, which is then used to choose an arm $\mathbf{x}_t$ according to the well-known optimism in the face of uncertainty principle. By relying on an online learner, GLOC inherently performs online computations and is thus free from the scalability issues in large time steps. While any online learner equipped with a low OL regret bound can be used, we choose the online Newton step (ONS) algorithm and prove a tight OL regret bound, which results in a practical GLB algorithm with almost the same regret bound as existing inefficient GLB algorithms. We present our proposed algorithms and their regret bounds in Section 3.

For large number N of arms, our proposed algorithm GLOC is not hash-amenable, to our knowledge, due to its nonlinear criterion for arm selection. As the first attempt, we derive a Thompson sampling [5, 3] extension of GLOC (GLOC-TS), which is hash-amenable due to its linear criterion. However, its regret bound scales with $d^{3/2}$ for d-dimensional arm sets, which is far from d of GLOC. Towards closing this gap, we propose a new algorithm Quadratic GLOC (QGLOC) with a regret bound that scales with $d^{5/4}$. We summarize the comparison of our proposed GLB algorithms in Table 1. In Section 4, we present GLOC-TS, QGLOC, and their regret bound.

Algorithm	Regret	Hash-amenable
GLOC	$\tilde{O}(d\sqrt{T})$	✗
GLOC-TS	$\tilde{O}(d^{3/2}\sqrt{T})$	✓
QGLOC	$\tilde{O}(d^{5/4}\sqrt{T})$	✓

Table 1: Comparison of GLBs algorithms for d-dimensional arm sets T is the time horizon. QGLOC achieves the smallest regret among hash-amenable algorithms.

Note that, while hashing achieves a time complexity sublinear in N, there is a nontrivial overhead of computing the projections to determine the hash keys. As an extra contribution, we reduce this overhead by proposing a new sampling-based approximate inner product method. Our proposed sampling method has smaller variance than the state-of-the-art sampling method proposed by [22, 24] when the vectors are normally distributed, which fits our setting where projection vectors are indeed normally distributed. Moreover, our method results in thinner tails in the distribution of estimation

[1] Without this designation, no *currently known* bandit algorithm achieves a sublinear time complexity in N.

error than the existing method, which implies a better concentration. We elaborate more on reducing the computational complexity of QOFUL in Section 5.

2 Preliminaries

We review relevant backgrounds here. $\mathcal{A}$ refers to a GLB algorithm, and $\mathcal{B}$ refers to an online learning algorithm. Let $\mathcal{B}_d(S)$ be the d-dimensional Euclidean ball of radius S, which overloads the notation $\mathcal{B}$. Let $\mathbf{A}_{\cdot i}$ be the i-th column vector of a matrix $\mathbf{A}$. Define $||\mathbf{x}||_{\mathbf{A}} := \sqrt{\mathbf{x}^\top \mathbf{A}\mathbf{x}}$ and $\text{vec}(\mathbf{A}) := [\mathbf{A}_{\cdot 1}; \mathbf{A}_{\cdot 2}; \cdots; \mathbf{A}_{\cdot d}] \in \mathbb{R}^{d^2}$ Given a function $f : \mathbb{R} \to \mathbb{R}$, we denote by f' and f'' its first and second derivative, respectively. We define $[N] := \{1, 2, \ldots, N\}$.

Generalized Linear Model (GLM) Consider modeling the reward y as one-dimensional exponential family such as Bernoulli or Poisson. When the feature vector $\mathbf{x}$ is believed to correlate with y, one popular modeling assumption is the generalized linear model (GLM) that turns the *natural parameter* of an exponential family model into $\mathbf{x}^\top \boldsymbol{\theta}^*$ where $\boldsymbol{\theta}^*$ is a parameter [29]:

$$\mathbb{P}(y \mid z = \mathbf{x}^\top \boldsymbol{\theta}^*) = \exp\left(\frac{yz - m(z)}{g(\tau)} + h(y, \tau)\right), \tag{1}$$

where $\tau \in \mathbb{R}^+$ is a known scale parameter and m, g, and h are normalizers. It is known that $m'(z) = \mathbb{E}[y \mid z] =: \mu(z)$ and $m''(z) = \text{Var}(y \mid z)$. We call $\mu(z)$ the *inverse link* function. Throughout, we assume that the exponential family being used in a GLM has a *minimal representation*, which ensures that $m(z)$ is strictly convex [38, Prop. 3.1]. Then, the negative log likelihood (NLL) $\ell(z, y) := -yz + m(z)$ of a GLM is strictly convex. We refer to such GLMs as the *canonical* GLM. In the case of Bernoulli rewards $y \in \{0, 1\}$, $m(z) = \log(1 + \exp(z))$, $\mu(z) = (1 + \exp(-z))^{-1}$, and the NLL can be written as the logistic loss: $\log(1 + \exp(-y'(\mathbf{x}_t^\top \boldsymbol{\theta}^*)))$, where $y' = 2y - 1$.

Generalized Linear Bandits (GLB) Recall that $\mathbf{x}_t$ is the arm chosen at time t by an algorithm. We assume that the arm set $\mathcal{X}_t$ can be of an infinite cardinality, although we focus on finite arm sets in hashing part of the paper (Section 4). One can write down the reward model (1) in a different form:

$$y_t = \mu(\mathbf{x}_t^\top \boldsymbol{\theta}^*) + \eta_t, \tag{2}$$

where η_t is conditionally R-sub-Gaussian given $\mathbf{x}_t$ and $\{(\mathbf{x}_s, \eta_s)\}_{s=1}^{t-1}$. For example, Bernoulli reward model has η_t as $1 - \mu(\mathbf{x}_t^\top \boldsymbol{\theta}^*)$ w.p. $\mu(\mathbf{x}_t^\top \boldsymbol{\theta}^*)$ and $-\mu(\mathbf{x}_t^\top \boldsymbol{\theta}^*)$ otherwise. Assume that $||\boldsymbol{\theta}^*||_2 \le S$, where S is known. One can show that the sub-Gaussian scale R is determined by μ: $R = \sup_{z \in (-S,S)} \sqrt{\mu'(z)} \le \sqrt{L}$, where L is the Lipschitz constant of μ. Throughout, we assume that each arm has ℓ_2-norm at most 1: $||\mathbf{x}||_2 \le 1, \forall \mathbf{x} \in \mathcal{X}_t, \forall t$. Let $\mathbf{x}_{t,*} := \max_{\mathbf{x} \in \mathcal{X}_t} \mathbf{x}^\top \boldsymbol{\theta}^*$. The performance of a GLB algorithm $\mathcal{A}$ is analyzed by the expected cumulative regret (or simply *regret*): $\text{Regret}_T^{\mathcal{A}} := \sum_{t=1}^T \mu(\mathbf{x}_{t,*}^\top \boldsymbol{\theta}^*) - \mu((\mathbf{x}_t^{\mathcal{A}})^\top \boldsymbol{\theta}^*)$, where $\mathbf{x}_t^{\mathcal{A}}$ makes the dependence on $\mathcal{A}$ explicit.

We remark that our results in this paper hold true for a strictly larger family of distributions than the canonical GLM, which we call the *non-canonical* GLM and explain below. The condition is that the reward model follows (2) where the R is now independent from μ that satisfies the following:

Assumption 1. *μ is L-Lipschitz on $[-S, S]$ and continuously differentiable on $(-S, S)$. Furthermore, $\inf_{z \in (-S,S)} \mu'(z) = \kappa$ for some finite $\kappa > 0$ (thus μ is strictly increasing).*

Define $\mu'(z)$ at $\pm S$ as their limits. Under Assumption 1, m is defined to be an integral of μ. Then, one can show that m is κ-strongly convex on $\mathcal{B}_1(S)$. An example of the non-canonical GLM is the probit model for 0/1 reward where μ is the Gaussian CDF, which is popular and competitive to the Bernoulli GLM as evaluated by Li et al. [27]. Note that canonical GLMs satisfy Assumption 1.

3 Generalized Linear Bandits with Online Computation

We describe and analyze a new GLB algorithm called Generalized Linear Online-to-confidence-set Conversion (GLOC) that performs online computations, unlike existing GLB algorithms.

GLOC employs the optimism in the face of uncertainty principle, which dates back to [7]. That is, we maintain a confidence set C_t (defined below) that traps the true parameter $\boldsymbol{\theta}^*$ with high probability (w.h.p.) and choose the arm with the largest feasible reward given C_{t-1} as a constraint:

$$(\mathbf{x}_t, \tilde{\boldsymbol{\theta}}_t) := \arg \max_{\mathbf{x} \in \mathcal{X}_t, \boldsymbol{\theta} \in C_{t-1}} \langle \mathbf{x}, \boldsymbol{\theta} \rangle \tag{3}$$

The main difference between GLOC and existing GLBs is in the computation of the C_t's. Prior methods involve "batch" computations that involve all past observations, and so scale poorly with

t. In contrast, GLOC takes in an *online* learner $\mathcal{B}$, and uses $\mathcal{B}$ as a co-routine instead of relying on a batch procedure to construct a confidence set. Specifically, at each time t GLOC feeds the loss function $\ell_t(\boldsymbol{\theta}) := \ell(\mathbf{x}_t^\top \boldsymbol{\theta}, y_t)$ into the learner $\mathcal{B}$ which then outputs its parameter prediction $\boldsymbol{\theta}_t$. Let $\mathbf{X}_t \in \mathbb{R}^{t\times d}$ be the design matrix consisting of $\mathbf{x}_1, \dots, \mathbf{x}_t$. Define $\overline{\mathbf{V}}_t := \lambda \mathbf{I} + \mathbf{X}_t^\top \mathbf{X}_t$, where λ is the ridge parameter. Let $z_t := \mathbf{x}_t^\top \boldsymbol{\theta}_t$ and $\mathbf{z}_t := [z_1; \cdots; z_t]$. Let $\widehat{\boldsymbol{\theta}}_t := \overline{\mathbf{V}}_t^{-1} \mathbf{X}_t^\top \mathbf{z}_t$ be the ridge regression estimator taking $\mathbf{z}_t$ as responses. Theorem 1 below is the key result for constructing our confidence set C_t, which is a function of the parameter predictions $\{\boldsymbol{\theta}_s\}_{s=1}^t$ and the online (OL) regret bound B_t of the learner $\mathcal{B}$. All the proofs are in the supplementary material (SM).

Theorem 1. *(Generalized Linear Online-to-Confidence-Set Conversion) Suppose we feed loss functions $\{\ell_s(\boldsymbol{\theta})\}_{s=1}^t$ into online learner $\mathcal{B}$. Let $\boldsymbol{\theta}_s$ be the parameter predicted at time step s by $\mathcal{B}$. Assume that $\mathcal{B}$ has an OL regret bound B_t: $\forall \boldsymbol{\theta} \in \mathcal{B}_d(S), \forall t \geq 1$,*

$$\textstyle\sum_{s=1}^t \ell_s(\boldsymbol{\theta}_s) - \ell_s(\boldsymbol{\theta}) \leq B_t \ . \tag{4}$$

Let $\alpha(B_t) := 1 + \frac{4}{\kappa} B_t + \frac{8R^2}{\kappa^2} \log(\frac{2}{\delta}\sqrt{1 + \frac{2}{\kappa} B_t + \frac{4R^4}{\kappa^4 \delta^2}})$. Then, with probability (w.p.) at least $1-\delta$,

$$\forall t \geq 1, ||\boldsymbol{\theta}^* - \widehat{\boldsymbol{\theta}}_t||_{\overline{\mathbf{V}}_t}^2 \leq \alpha(B_t) + \lambda S^2 - \left(||\mathbf{z}_t||_2^2 - \widehat{\boldsymbol{\theta}}_t^\top \mathbf{X}_t^\top \mathbf{z}_t\right) =: \beta_t \ . \tag{5}$$

Note that the center of the ellipsoid is the ridge regression estimator on the predicted natural parameters $z_s = \mathbf{x}_s^\top \boldsymbol{\theta}_s$ rather than the rewards. Theorem 1 motivates the following confidence set:

$$C_t := \{\boldsymbol{\theta} \in \mathbb{R}^d : ||\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_t||_{\overline{\mathbf{V}}_t}^2 \leq \beta_t\} \tag{6}$$

which traps $\boldsymbol{\theta}^*$ for all $t \geq 1$, w.p. at least $1-\delta$. See Algorithm 1 for pseudocode. One way to solve the optimization problem (3) is to define the function $\boldsymbol{\theta}(\mathbf{x}) := \max_{\boldsymbol{\theta} \in C_{t-1}} \mathbf{x}^\top \boldsymbol{\theta}$, and then use the Lagrangian method to write:

$$\mathbf{x}_t^{\text{GLOC}} := \arg\max_{\mathbf{x} \in \mathcal{X}_t} \mathbf{x}^\top \widehat{\boldsymbol{\theta}}_{t-1} + \sqrt{\beta_{t-1}} ||\mathbf{x}||_{\overline{\mathbf{V}}_{t-1}^{-1}} \ . \tag{7}$$

We prove the regret bound of GLOC in the following theorem.

Theorem 2. *Let $\{\overline{\beta}_t\}$ be a nondecreasing sequence such that $\overline{\beta}_t \geq \beta_t$. Then, w.p. at least $1-\delta$,*

$$\text{Regret}_T^{\text{GLOC}} = O\left(L\sqrt{\overline{\beta}_T dT \log T}\right)$$

Although any low-regret online learner can be combined with GLOC, one would like to ensure that $\overline{\beta}_T$ is $O(\text{polylog}(T))$ in which case the total regret can be bounded by $\tilde{O}(\sqrt{T})$. This means that we must use online learners whose OL regret grows logarithmically in T such as [20, 31]. In this work, we consider the online Newton step (ONS) algorithm [20].

Online Newton Step (ONS) for Generalized Linear Models Note that ONS requires the loss functions to be α-exp-concave. One can show that $\ell_t(\boldsymbol{\theta})$ is α-exp-concave [20, Sec. 2.2]. Then, GLOC can use ONS and its OL regret bound to solve the GLB problem. However, motivated by the fact that the OL regret bound B_t appears in the radius $\sqrt{\beta_t}$ of the confidence set while a tighter confidence set tends to reduce the bandit regret in practice, we derive a tight data-dependent OL regret bound tailored to GLMs.

We present our version of ONS for GLMs (ONS-GLM) in Algorithm 2. $\ell'(z, y)$ is the first derivative w.r.t. z and the parameter ϵ is for inverting matrices conveniently (usually $\epsilon = 1$ or 0.1). The only difference from the original ONS [20] is that we rely on the strong convexity of $m(z)$ instead of the α-exp-concavity of the loss thanks to the GLM structure.[2] Theorem 3 states that we achieve the desired polylogarithmic regret in T.

Algorithm 1 GLOC

Input: $R > 0$, $\delta \in (0,1)$, $S > 0$, $\lambda > 0$, $\kappa > 0$, an online learner $\mathcal{B}$ with known regret bounds $\{B_t\}_{t \geq 1}$.
Set $\overline{\mathbf{V}}_0 = \lambda \mathbf{I}$.
for $t = 1, 2, \dots$ **do**
  Compute $\mathbf{x}_t$ by solving (3).
  Pull $\mathbf{x}_t$ and then observe y_t.
  Receive $\boldsymbol{\theta}_t$ from $\mathcal{B}$.
  Feed into $\mathcal{B}$ the loss $\ell_t(\boldsymbol{\theta}) = \ell(\mathbf{x}_t^\top \boldsymbol{\theta}, y_t)$.
  Update $\overline{\mathbf{V}}_t = \overline{\mathbf{V}}_{t-1} + \mathbf{x}_t \mathbf{x}_t^\top$ and $z_t = \mathbf{x}_t^\top \boldsymbol{\theta}_t$
  Compute $\widehat{\boldsymbol{\theta}}_t = \overline{\mathbf{V}}_t^{-1} \mathbf{X}_t^\top \mathbf{z}_t$ and β_t as in (5).
  Define C_t as in (6).
end for

Algorithm 2 ONS-GLM

Input: $\kappa > 0$, $\epsilon > 0$, $S > 0$.
$\mathbf{A}_0 = \epsilon \mathbf{I}$.
Set $\boldsymbol{\theta}_1 \in \mathcal{B}_d(S)$ arbitrarily.
for $t = 1, 2, 3, \dots$ **do**
  Output $\boldsymbol{\theta}_t$.
  Observe $\mathbf{x}_t$ and y_t.
  Incur loss $\ell(\mathbf{x}_t^\top \boldsymbol{\theta}_t, y_t)$.
  $\mathbf{A}_t = \mathbf{A}_{t-1} + \mathbf{x}_t \mathbf{x}_t^\top$
  $\boldsymbol{\theta}'_{t+1} = \boldsymbol{\theta}_t - \frac{\ell'(\mathbf{x}_t^\top \boldsymbol{\theta}_t, y_t)}{\kappa} \mathbf{A}_t^{-1} \mathbf{x}_t$
  $\boldsymbol{\theta}_{t+1} = \arg\min_{\boldsymbol{\theta} \in \mathcal{B}_d(S)} ||\boldsymbol{\theta} - \boldsymbol{\theta}'_{t+1}||_{\mathbf{A}_t}^2$
end for

[2] A similar change to ONS has been applied in [16, 41].

Theorem 3. *Define $g_s := \ell'(\mathbf{x}_s^\top \boldsymbol{\theta}_s, y_s)$. The regret of ONS-GLM satisfies, for any $\epsilon > 0$ and $t \geq 1$,*

$$\sum_{s=1}^t \ell_s(\boldsymbol{\theta}_s) - \ell_s(\boldsymbol{\theta}^*) \leq \frac{1}{2\kappa} \sum_{s=1}^t g_s^2 ||\mathbf{x}_s||^2_{\mathbf{A}_s^{-1}} + 2\kappa S^2 \epsilon =: B_t^{\text{ONS}} \ ,$$

where $B_t^{\text{ONS}} = O(\frac{L^2+R^2\log(t)}{\kappa} d \log t), \forall t \geq 1$ w.h.p. If $\max_{s\geq 1} |\eta_s|$ is bounded by $\bar{R}$ w.p. 1, $B_t^{\text{ONS}} = O(\frac{L^2+\bar{R}^2}{\kappa} d \log t)$.

We emphasize that the OL regret bound is data-dependent. A confidence set constructed by combining Theorem 1 and Theorem 3 directly implies the following regret bound of GLOC with ONS-GLM.

Corollary 1. *Define β_t^{ONS} by replacing B_t with B_t^{ONS} in (5). With probability at least $1-2\delta$,*

$$\forall t \geq 1, \boldsymbol{\theta}^* \in C_t^{\text{ONS}} := \left\{ \boldsymbol{\theta} \in \mathbb{R}^d : ||\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_t||^2_{\bar{\mathbf{V}}_t} \leq \beta_t^{\text{ONS}} \right\} \ . \tag{8}$$

Corollary 2. *Run GLOC with C_t^{ONS}. Then, w.p. at least $1-2\delta$, $\forall T \geq 1$, $\text{Regret}_T^{\text{GLOC}} = \hat{O}\left(\frac{L(L+R)}{\kappa} d\sqrt{T}\log^{3/2}(T)\right)$ where $\hat{O}$ ignores $\log\log(t)$. If $|\eta_t|$ is bounded by $\bar{R}$, $\text{Regret}_T^{\text{GLOC}} = \hat{O}\left(\frac{L(L+\bar{R})}{\kappa} d\sqrt{T}\log(T)\right)$.*

We make regret bound comparisons ignoring $\log\log T$ factors. For generic arm sets, our dependence on d is optimal for linear rewards [34]. For the Bernoulli GLM, our regret has the same order as Zhang et al. [41]. One can show that the regret of Filippi et al. [15] has the same order as ours if we use their assumption that the reward y_t is bounded by $R_{\max}$. For unbounded noise, Li et al. [28] have regret $O((LR/\kappa)d\sqrt{T}\log T)$, which is $\sqrt{\log T}$ factor smaller than ours and has LR in place of $L(L+R)$. While $L(L+R)$ could be an artifact of our analysis, the gap is not too large for canonical GLMs. Let L be the smallest Lipschitz constant of μ. Then, $R = \sqrt{L}$. If $L \leq 1$, R satisfies $R > L$, and so $L(L+R) = O(LR)$. If $L > 1$, then $L(L+R) = O(L^2)$, which is larger than $LR = O(L^{3/2})$. For the Gaussian GLM with known variance σ^2, $L = R = 1$.[3] For finite arm sets, SupCB-GLM of Li et al. [28] achieves regret of $\tilde{O}(\sqrt{dT\log N})$ that has a better scaling with d but is not a practical algorithm as it wastes a large number of arm pulls. Finally, we remark that none of the existing GLB algorithms are scalable to large T. Zhang et al. [41] is scalable to large T, but is restricted to the Bernoulli GLM; e.g., theirs does not allow the probit model (non-canonical GLM) that is popular and shown to be competitive to the Bernoulli GLM [27].

Discussion The trick of obtaining a confidence set from an online learner appeared first in [13, 14] for the linear model, and then was used in [10, 16, 41]. GLOC is slightly different from these studies and rather close to Abbasi-Yadkori et al. [2] in that the confidence set is a function of a known regret bound. This generality frees us from re-deriving a confidence set for every online learner. Our result is essentially a nontrivial extension of Abbasi-Yadkori et al. [2] to GLMs.

One might have notice that C_t does not use $\boldsymbol{\theta}_{t+1}$ that is available before pulling $\mathbf{x}_{t+1}$ and has the most up-to-date information. This is inherent to GLOC as it relies on the OL regret bound directly. One can modify the proof of ONS-GLM to have a tighter confidence set C_t that uses $\boldsymbol{\theta}_{t+1}$ as we show in SM Section E. However, this is now specific to ONS-GLM, which looses generality.

4 Hash-Amenable Generalized Linear Bandits

We now turn to a setting where the arm set is finite but very large. For example, imagine an interactive retrieval scenario [33, 25, 6] where a user is shown K images (e.g., shoes) at a time and provides relevance feedback (e.g., yes/no or 5-star rating) on each image, which is repeated until the user is satisfied. In this paper, we focus on showing one image (i.e., arm) at a time.[4] Most existing algorithms require maximizing an objective function (e.g., (7)), the complexity of which scales linearly with the number N of arms. This can easily become prohibitive for large numbers of images. Furthermore, the system has to perform real-time computations to promptly choose which image to show the user in the next round. Thus, it is critical for a practical system to have a time complexity sublinear in N.

One naive approach is to select a subset of arms ahead of time, such as volumetric spanners [19]. However, this is specialized for an efficient exploration only and can rule out a large number of good arms. Another option is to use hashing methods. Locality-sensitive hashing and Maximum

[3] The reason why R is not σ here is that the sufficient statistic of the GLM is y/σ, which is equivalent to dealing with the normalized reward. Then, σ appears as a factor in the regret bound.

[4] One image at a time is a simplification of the practical setting. One can extend it to showing multiple images at a time, which is a special case of the combinatorial bandits of Qin et al. [32].

Inner Product Search (MIPS) are effective and well-understood tools but can only be used when the objective function is a distance or an inner product computation; (7) cannot be written in this form. In this section, we consider alternatives to GLOC which are compatible with hashing.

Thompson Sampling We present a Thompson sampling (TS) version of GLOC called GLOC-TS that chooses an arm $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}_t} \mathbf{x}^\top \dot{\boldsymbol{\theta}}_t$ where $\dot{\boldsymbol{\theta}}_t \sim \mathcal{N}(\widehat{\boldsymbol{\theta}}_{t-1}, \beta_{t-1}\overline{\mathbf{V}}_{t-1}^{-1})$. TS is known to perform well in practice [8] and can solve the polytope arm set case in polynomial time[5] whereas algorithms that solve an objective function like (3) (e.g., [1]) cannot since they have to solve an NP-hard problem [5]. We present the regret bound of GLOC-TS below. Due to space constraints, we present the pseudocode and the full version of the result in SM.

Theorem 4. *(Informal) If we run GLOC-TS with* $\dot{\boldsymbol{\theta}}_t \sim \mathcal{N}(\widehat{\boldsymbol{\theta}}_{t-1}, \beta_{t-1}^{\text{ONS}}\overline{\mathbf{V}}_{t-1}^{-1})$, $\text{Regret}_T^{\text{GLOC-TS}} = \hat{O}\left(\frac{L(L+R)}{\kappa} d^{3/2}\sqrt{T}\log^{3/2}(T)\right)$ *w.h.p. If* η_t *is bounded by* $\bar{R}$*, then* $\hat{O}\left(\frac{L(L+\bar{R})}{\kappa} d^{3/2}\sqrt{T}\log(T)\right)$.

Notice that the regret now scales with $d^{3/2}$ as expected from the analysis of linear TS [4], which is higher than scaling with d of GLOC. This is concerning in the interactive retrieval or product recommendation scenario since the relevance of the shown items is harmed, which makes us wonder if one can improve the regret without loosing the hash-amenability.

Quadratic GLOC We now propose a new hash-amenable algorithm called Quadratic GLOC (QGLOC). Recall that GLOC chooses the arm $\mathbf{x}^{\text{GLOC}}$ by (7). Define $r = \min_{\mathbf{x}\in\mathcal{X}} ||\mathbf{x}||_2$ and

$$\overline{m}_{t-1} := \min_{\mathbf{x}:||\mathbf{x}||_2\in[r,1]} ||\mathbf{x}||_{\overline{\mathbf{V}}_{t-1}^{-1}} , \tag{9}$$

which is r times the square root of the smallest eigenvalue of $\overline{\mathbf{V}}_{t-1}^{-1}$. It is easy to see that $\overline{m}_{t-1} \leq ||\mathbf{x}||_{\overline{\mathbf{V}}_{t-1}^{-1}}$ for all $\mathbf{x} \in \mathcal{X}$ and that $\overline{m}_{t-1} \geq r/\sqrt{t+\lambda}$ using the definition of $\overline{\mathbf{V}}_{t-1}$. There is an alternative way to define $\overline{m}_{t-1}$ without relying on r, which we present in SM.

Let $c_0 > 0$ be the exploration-exploitation tradeoff parameter (elaborated upon later). At time t, QGLOC chooses the arm

$$\mathbf{x}_t^{\text{QGLOC}} := \arg\max_{\mathbf{x}\in\mathcal{X}_t} \langle\widehat{\boldsymbol{\theta}}_{t-1}, \mathbf{x}\rangle + \frac{\beta_{t-1}^{1/4}}{4c_0\overline{m}_{t-1}} ||\mathbf{x}||^2_{\overline{\mathbf{V}}_{t-1}^{-1}} = \arg\max_{\mathbf{x}\in\mathcal{X}_t} \langle\mathbf{q}_t, \phi(\mathbf{x})\rangle \ , \tag{10}$$

where $\mathbf{q}_t = [\widehat{\boldsymbol{\theta}}_{t-1}; \text{vec}(\frac{\beta_{t-1}^{1/4}}{4c_0\overline{m}_{t-1}}\overline{\mathbf{V}}_{t-1}^{-1})] \in \mathbb{R}^{d+d^2}$ and $\phi(\mathbf{x}) := [\mathbf{x}; \text{vec}(\mathbf{x}\mathbf{x}^\top)]$. The key property of QGLOC is that the objective function is now quadratic in $\mathbf{x}$, thus the name *Quadratic* GLOC, and can be written as an inner product. Thus, QGLOC is hash-amenable. We present the regret bound of QGLOC (10) in Theorem 5. The key step of the proof is that the QGLOC objective function (10) plus $c_0\beta^{3/4}\overline{m}_{t-1}$ is a tight upper bound of the GLOC objective function (7).

Theorem 5. *Run QGLOC with* C_t^{ONS}*. Then, w.p. at least* $1-2\delta$*,* $\text{Regret}_T^{QGLOC} = O\left(\left(\frac{1}{c_0}\left(\frac{L+R}{\kappa}\right)^{1/2} + c_0\left(\frac{L+R}{\kappa}\right)^{3/2}\right) Ld^{5/4}\sqrt{T}\log^2(T)\right)$*. By setting* $c_0 = \left(\frac{L+R}{\kappa}\right)^{-1/2}$*, the regret bound is* $O(\frac{L(L+R)}{\kappa} d^{5/4}\sqrt{T}\log^2(T))$.

Note that one can have a better dependence on $\log T$ when η_t is bounded (available in the proof). The regret bound of QGLOC is a $d^{1/4}$ factor improvement over that of GLOC-TS; see Table 1. Furthermore, in (10) c_0 is a free parameter that adjusts the balance between the exploitation (the first term) and exploration (the second term). Interestingly, the regret guarantee *does not break down* when adjusting c_0 in Theorem 5. Such a characteristic is not found in existing algorithms but is attractive to practitioners, which we elaborate in SM.

Maximum Inner Product Search (MIPS) Hashing While MIPS hashing algorithms such as [35, 36, 30] can solve (10) in time sublinear in N, these necessarily introduce an approximation error. Ideally, one would like the following guarantee on the error with probability at least $1-\delta_{\text{H}}$:

Definition 1. *Let* $\mathcal{X} \subseteq \mathbb{R}^{d'}$ *satisfy* $|\mathcal{X}| < \infty$*. A data point* $\tilde{\mathbf{x}} \in \mathcal{X}$ *is called* c_{H}*-MIPS w.r.t. a given query* $\mathbf{q}$ *if it satisfies* $\langle\mathbf{q}, \tilde{\mathbf{x}}\rangle \geq c_{\text{H}} \cdot \max_{\mathbf{x}\in\mathcal{X}} \langle\mathbf{q}, \mathbf{x}\rangle$ *for some* $c_{\text{H}} < 1$*. An algorithm is called* c_{H}*-MIPS if, given a query* $\mathbf{q} \in \mathbb{R}^{d'}$*, it retrieves* $\mathbf{x} \in \mathcal{X}$ *that is* c_{H}*-MIPS w.r.t.* $\mathbf{q}$*.*

Unfortunately, existing MIPS algorithms do not directly offer such a guarantee, and one must build a series of hashing schemes with varying hashing parameters like Har-Peled et al. [18]. Under the fixed budget setting T, we elaborate our construction that is simpler than [18] in SM.

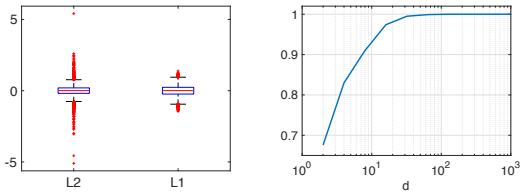
[5] ConfidenceBall$_1$ algorithm of Dani et al. [11] can solve the problem in polynomial time as well.

Time and Space Complexity Our construction involves saving Gaussian projection vectors that are used for determining hash keys and saving the buckets containing pointers to the actual arm vectors. The time complexity for retrieving a c_{H}-MIPS solution involves determining hash keys and evaluating inner products with the arms in the retrieved buckets. Let $\rho^* < 1$ be an optimized value for the hashing (see [35] for detail). The time complexity for d'-dimensional vectors is $O\left(\log\left(\frac{\log(dT)}{\log(c_{\mathrm{H}}^{-1})}\right) N^{\rho^*} \log(N) d'\right)$, and the space complexity (except the original data) is $O\left(\frac{\log(dT)}{\log(c_{\mathrm{H}}^{-1})} N^{\rho^*}(N + \log(N) d')\right)$. While the time and space complexity grows with the time horizon T, the dependence is mild; $\log\log(T)$ and $\log(T)$, respectively. QGLOC uses $d' = d + d^2$,[6] and GLOC-TS uses $d' = d'$. While both achieve a time complexity sublinear in N, the time complexity of GLOC-TS scales with d that is better than scaling with d^2 of QGLOC. However, GLOC-TS has a $d^{1/4}$-factor worse regret bound than QGLOC.

Discussion While it is reasonable to incur small errors in solving the arm selection criteria like (10) and sacrifice some regret in practice, the regret bounds of QGLOC and GLOC-TS do not hold anymore. Though not the focus of our paper, we prove a regret bound under the presence of the hashing error in the fixed budget setting for QGLOC; see SM. Although the result therein has an inefficient space complexity that is linear in T, it provides the first low regret bound with time sublinear in N, to our knowledge.

5 Approximate Inner Product Computations with L1 Sampling

While hashing allows a time complexity sublinear in N, it performs an additional computation for determining the hash keys. Consider a hashing with U tables and length-k hash keys. Given a query $\mathbf{q}$ and projection vectors $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(Uk)}$, the hashing computes $\mathbf{q}^\top \mathbf{a}^{(i)}$, $\forall i \in [Uk]$ to determine the hash key of $\mathbf{q}$. To reduce such an overhead, approximate inner product methods like [22, 24] are attractive since hash keys are determined by discretizing the inner products; small inner product errors often do not alter the hash keys.



(a) (b)

Figure 1: (a) A box plot of estimators. L1 and L2 have the same variance, but L2 has thicker tails. (b) The frequency of L1 inducing smaller variance than L2 in 1000 trials. After 100 dimensions, L1 mostly has smaller variance than L2.

In this section, we propose an improved approximate inner product method called *L1 sampling* which we claim is more accurate than the sampling proposed by Jain et al. [22], which we call *L2 sampling*. Consider an inner product $\mathbf{q}^\top \mathbf{a}$. The main idea is to construct an unbiased estimate of $\mathbf{q}^\top \mathbf{a}$. That is, let $\mathbf{p} \in \mathbb{R}^d$ be a probability vector. Let

$$i_k \overset{\text{i.i.d.}}{\sim} \text{Multinomial}(\mathbf{p}) \quad \text{and} \quad G_k := q_{i_k} a_{i_k} / p_{i_k},\ k \in [m]\ . \tag{11}$$

It is easy to see that $\mathbb{E}G_k = \mathbf{q}^\top \mathbf{a}$. By taking $\frac{1}{m}\sum_{k=1}^m G_k$ as an estimate of $\mathbf{q}^\top \mathbf{a}$, the time complexity is now $O(mUk)$ rather than $O(d'Uk)$. The key is to choose the right $\mathbf{p}$. L2 sampling uses $\mathbf{p}^{(\mathrm{L2})} := [q_i^2 / ||\mathbf{q}||_2^2]_i$. Departing from L2, we propose $\mathbf{p}^{(\mathrm{L1})}$ that we call L1 sampling and define as follows:

$$\mathbf{p}^{(\mathrm{L1})} := [|q_1|; \cdots; |q_{d'}|] / ||\mathbf{q}||_1\ . \tag{12}$$

We compare L1 with L2 in two different point of view. Due to space constraints, we summarize the key ideas and defer the details to SM.

The first is on their concentration of measure. Lemma 1 below shows an error bound of L1 whose failure probability decays exponentially in m. This is in contrast to decaying polynomially of L2 [22], which is inferior.[7]

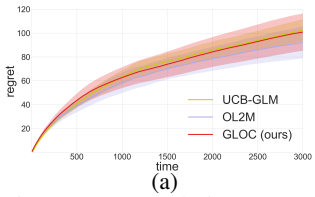
Lemma 1. *Define G_k as in* (11) *with $\mathbf{p} = \mathbf{p}^{(\mathrm{L1})}$. Then, given a target error $\epsilon > 0$,*

$$\mathbb{P}\left(\left|\tfrac{1}{m}\textstyle\sum_{k=1}^m G_k - \mathbf{q}^\top \mathbf{a}\right| \geq \epsilon\right) \leq 2\exp\left(-\frac{m\epsilon^2}{2||\mathbf{q}||_1^2 ||\mathbf{a}||_{\max}^2}\right) \tag{13}$$

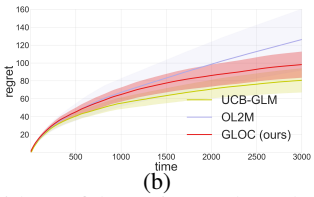
To illustrate such a difference, we fix $\mathbf{q}$ and $\mathbf{a}$ in 1000 dimension and apply L2 and L1 sampling 20K times each with $m = 5$ where we scale down the L2 distribution so its variance matches that of L1.

[6] Note that this does not mean we need to store $\mathrm{vec}(\mathbf{x}\mathbf{x}^\top)$ since an inner product with it is structured.

[7] In fact, one can show a bound for L2 that fails with exponentially-decaying probability. However, the bound introduces a constant that can be arbitrarily large, which makes the tails thick. We provide details on this in SM.



(a)



(b)

Algorithm	Cum. Regret
QGLOC	266.6 ($\pm$19.7)
QGLOC-Hash	285.0 ($\pm$30.3)
GLOC-TS	277.0 ($\pm$36.1)
GLOC-TS-Hash	289.1 ($\pm$28.1)

(c)

Figure 2: Cumulative regrets with confidence intervals under the (a) logit and (b) probit model. (c) Cumulative regrets with confidence intervals of hash-amenable algorithms.

Figure 1(a) shows that L2 has thicker tails than L1. Note this is not a pathological case but a typical case for Gaussian $\mathbf{q}$ and $\mathbf{a}$. This confirms our claim that L1 is safer than L2.

Another point of comparison is the variance of L2 and L1. We show that the variance of L1 may or may not be larger than L2 in SM; there is no absolute winner. However, if $\mathbf{q}$ and $\mathbf{a}$ follow a Gaussian distribution, then L1 induces smaller variances than L2 for large enough d; see Lemma 9 in SM. Figure 1(b) confirms such a result. The actual gap between the variance of L2 and L1 is also nontrivial under the Gaussian assumption. For instance, with $d = 200$, the average variance of G_k induced by L2 is 0.99 whereas that induced by L1 is 0.63 on average. Although a stochastic assumption on the vectors being inner-producted is often unrealistic, in our work we deal with projection vectors $\mathbf{a}$ that are truly normally distributed.

6 Experiments

We now show our experiment results comparing GLB algorithms and hash-amenable algorithms.

GLB Algorithms We compare GLOC with two different algorithms: UCB-GLM [28] and Online Learning for Logit Model (OL2M) [41].[8] For each trial, we draw $\boldsymbol{\theta}^* \in \mathbb{R}^d$ and N arms ($\mathcal{X}$) uniformly at random from the unit sphere. We set $d = 10$ and $\mathcal{X}_t = \mathcal{X}$, $\forall t \geq 1$. Note it is a common practice to scale the confidence set radius for bandits [8, 27]. Following Zhang et al. [41], for OL2M we set the squared radius $\gamma_t = c \log(\det(\mathbf{Z}_t)/\det(\mathbf{Z}_1))$, where c is a tuning parameter. For UCB-GLM, we set the radius as $\alpha = \sqrt{cd \log t}$. For GLOC, we replace β_t^{ONS} with $c \sum_{s=1}^t g_s^2 ||\mathbf{x}_s||^2_{\mathbf{A}_s^{-1}}$. While parameter tuning in practice is nontrivial, for the sake of comparison we tune $c \in \{10^1, 10^{0.5}, \ldots, 10^{-3}\}$ and report the best one. We perform 40 trials up to time $T = 3000$ for each method and compute confidence bounds on the regret.

We consider two GLM rewards: (i) the logit model (the Bernoulli GLM) and (ii) the probit model (non-canonical GLM) for 0/1 rewards that sets μ as the probit function. Since OL2M is for the logit model only, we expect to see the consequences of model mismatch in the probit setting. For GLOC and UCB-GLM, we specify the correct reward model. We plot the cumulative regret under the logit model in Figure 2(a). All three methods perform similarly, and we do not find any statistically significant difference based on paired t test. The result for the probit model in Figure 2(b) shows that OL2M indeed has higher regret than both GLOC and UCB-GLM due to the model mismatch in the probit setting. Specifically, we verify that at $t = 3000$ the difference between the regret of UCB-GLM and OL2M is statistically significant. Furthermore, OL2M exhibits a significantly higher variance in the regret, which is unattractive in practice. This shows the importance of being generalizable to *any* GLM reward. Note we observe a big increase in running time for UCB-GLM compared to OL2M and GLOC.

Hash-Amenable GLBs To compare hash-amenable GLBs, we use the logit model as above but now with N=100,000 and T=5000. We run QGLOC, QGLOC with hashing (QGLOC-Hash), GLOC-TS, and GLOC-TS with hashing (GLOC-TS-Hash), where we use the hashing to compute the objective function (e.g., (10)) on just 1% of the data points and save a significant amount of computation. Details on our hashing implementation is found in SM. Figure 2(c) summarizes the result. We observe that QGLOC-Hash and GLOC-TS-Hash increase regret from QGLOC and GLOC-TS, respectively, but only moderately, which shows the efficacy of hashing.

7 Future Work

In this paper, we have proposed scalable algorithms for the GLB problem: (i) for large time horizon T and (ii) for large number N of arms. There exists a number of interesting future work. First,

[8] We have chosen UCB-GLM over GLM-UCB of Filippi et al. [15] as UCB-GLM has a lower regret bound.

we would like to extend the GLM rewards to the single index models [23] so one does not need to know the function μ ahead of time under mild assumptions. Second, closing the regret bound gap between QGLOC and GLOC without loosing hash-amenability would be interesting: i.e., develop a hash-amenable GLB algorithm with $O(d\sqrt{T})$ regret. In this direction, a first attempt could be to design a hashing scheme that can directly solve (7) approximately.

Acknowledgments This work was partially supported by the NSF grant IIS-1447449 and the MURI grant 2015-05174-04. The authors thank Yasin Abbasi-Yadkori and Anshumali Shrivastava for providing constructive feedback and Xin Hunt for her contribution at the initial stage.

References

[1] Abbasi-Yadkori, Yasin, Pal, David, and Szepesvari, Csaba. Improved Algorithms for Linear Stochastic Bandits. *Advances in Neural Information Processing Systems (NIPS)*, pp. 1–19, 2011.

[2] Abbasi-Yadkori, Yasin, Pal, David, and Szepesvari, Csaba. Online-to-Confidence-Set Conversions and Application to Sparse Stochastic Bandits. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

[3] Abeille, Marc and Lazaric, Alessandro. Linear Thompson Sampling Revisited. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54, pp. 176–184, 2017.

[4] Agrawal, Shipra and Goyal, Navin. Thompson Sampling for Contextual Bandits with Linear Payoffs. *CoRR*, abs/1209.3, 2012.

[5] Agrawal, Shipra and Goyal, Navin. Thompson Sampling for Contextual Bandits with Linear Payoffs. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 127–135, 2013.

[6] Ahukorala, Kumaripaba, Medlar, Alan, Ilves, Kalle, and Glowacka, Dorota. Balancing Exploration and Exploitation: Empirical Parameterization of Exploratory Search Systems. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 1703–1706, 2015.

[7] Auer, Peter and Long, M. Using Confidence Bounds for Exploitation-Exploration Trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

[8] Chapelle, Olivier and Li, Lihong. An Empirical Evaluation of Thompson Sampling. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2249–2257, 2011.

[9] Chu, Wei, Li, Lihong, Reyzin, Lev, and Schapire, Robert E. Contextual Bandits with Linear Payoff Functions. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15, pp. 208–214, 2011.

[10] Crammer, Koby and Gentile, Claudio. Multiclass Classification with Bandit Feedback Using Adaptive Regularization. *Mach. Learn.*, 90(3):347–383, 2013.

[11] Dani, Varsha, Hayes, Thomas P, and Kakade, Sham M. Stochastic Linear Optimization under Bandit Feedback. In *Proceedings of the Conference on Learning Theory (COLT)*, pp. 355–366, 2008.

[12] Datar, Mayur, Immorlica, Nicole, Indyk, Piotr, and Mirrokni, Vahab S. Locality-sensitive Hashing Scheme Based on P-stable Distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, pp. 253–262, 2004.

[13] Dekel, Ofer, Gentile, Claudio, and Sridharan, Karthik. Robust selective sampling from single and multiple teachers. In *In Proceedings of the Conference on Learning Theory (COLT)*, 2010.

[14] Dekel, Ofer, Gentile, Claudio, and Sridharan, Karthik. Selective sampling and active learning from single and multiple teachers. *Journal of Machine Learning Research*, 13:2655–2697, 2012.

[15] Filippi, Sarah, Cappe, Olivier, Garivier, Aurélien, and Szepesvári, Csaba. Parametric Bandits: The Generalized Linear Case. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 586–594. 2010.

[16] Gentile, Claudio and Orabona, Francesco. On Multilabel Classification and Ranking with Bandit Feedback. *Journal of Machine Learning Research*, 15:2451–2487, 2014.

[17] Guo, Ruiqi, Kumar, Sanjiv, Choromanski, Krzysztof, and Simcha, David. Quantization based Fast Inner Product Search. *Journal of Machine Learning Research*, 41:482–490, 2016.

[18] Har-Peled, Sariel, Indyk, Piotr, and Motwani, Rajeev. Approximate nearest neighbor: towards removing the curse of dimensionality. *Theory of Computing*, 8:321–350, 2012.

[19] Hazan, Elad and Karnin, Zohar. Volumetric Spanners: An Efficient Exploration Basis for Learning. *Journal of Machine Learning Research*, 17(119):1–34, 2016.

[20] Hazan, Elad, Agarwal, Amit, and Kale, Satyen. Logarithmic Regret Algorithms for Online Convex Optimization. *Mach. Learn.*, 69(2-3):169–192, 2007.

[21] Hofmann, Katja, Whiteson, Shimon, and de Rijke, Maarten. Contextual Bandits for Information Retrieval. In *NIPS Workshop on Bayesian Optimization, Experimental Design and Bandits: Theory and Applications*, 2011.

[22] Jain, Prateek, Vijayanarasimhan, Sudheendra, and Grauman, Kristen. Hashing Hyperplane Queries to Near Points with Applications to Large-Scale Active Learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 928–936, 2010.

[23] Kalai, Adam Tauman and Sastry, Ravi. The Isotron Algorithm: High-Dimensional Isotonic Regression. In *Proceedings of the Conference on Learning Theory (COLT)*, 2009.

[24] Kannan, Ravindran, Vempala, Santosh, and Others. Spectral algorithms. *Foundations and Trends in Theoretical Computer Science*, 4(3–4):157–288, 2009.

[25] Konyushkova, Ksenia and Glowacka, Dorota. Content-based image retrieval with hierarchical Gaussian Process bandits with self-organizing maps. In *21st European Symposium on Artificial Neural Networks*, 2013.

[26] Li, Lihong, Chu, Wei, Langford, John, and Schapire, Robert E. A Contextual-Bandit Approach to Personalized News Article Recommendation. *Proceedings of the International Conference on World Wide Web (WWW)*, pp. 661–670, 2010.

[27] Li, Lihong, Chu, Wei, Langford, John, Moon, Taesup, and Wang, Xuanhui. An Unbiased Offline Evaluation of Contextual Bandit Algorithms with Generalized Linear Models. In *Proceedings of the Workshop on On-line Trading of Exploration and Exploitation 2*, volume 26, pp. 19–36, 2012.

[28] Li, Lihong, Lu, Yu, and Zhou, Dengyong. Provable Optimal Algorithms for Generalized Linear Contextual Bandits. *CoRR*, abs/1703.0, 2017.

[29] McCullagh, P and Nelder, J A. *Generalized Linear Models*. London, 1989.

[30] Neyshabur, Behnam and Srebro, Nathan. On Symmetric and Asymmetric LSHs for Inner Product Search. *Proceedings of the International Conference on Machine Learning (ICML)*, 37: 1926–1934, 2015.

[31] Orabona, Francesco, Cesa-Bianchi, Nicolo, and Gentile, Claudio. Beyond Logarithmic Bounds in Online Learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 22, pp. 823–831, 2012.

[32] Qin, Lijing, Chen, Shouyuan, and Zhu, Xiaoyan. Contextual Combinatorial Bandit and its Application on Diversified Online Recommendation. In *SDM*, pp. 461–469, 2014.

[33] Rui, Yong, Huang, T S, Ortega, M, and Mehrotra, S. Relevance feedback: a power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):644–655, 1998.

[34] Rusmevichientong, Paat and Tsitsiklis, John N. Linearly Parameterized Bandits. *Math. Oper. Res.*, 35(2):395–411, 2010.

[35] Shrivastava, Anshumali and Li, Ping. Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS). *Advances in Neural Information Processing Systems 27*, pp. 2321–2329, 2014.

[36] Shrivastava, Anshumali and Li, Ping. Improved Asymmetric Locality Sensitive Hashing (ALSH) for Maximum Inner Product Search (MIPS). In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 812–821, 2015.

[37] Slaney, Malcolm, Lifshits, Yury, and He, Junfeng. Optimal parameters for locality-sensitive hashing. *Proceedings of the IEEE*, 100(9):2604–2623, 2012.

[38] Wainwright, Martin J and Jordan, Michael I. Graphical Models, Exponential Families, and Variational Inference. *Found. Trends Mach. Learn.*, 1(1-2):1–305, 2008.

[39] Wang, Jingdong, Shen, Heng Tao, Song, Jingkuan, and Ji, Jianqiu. Hashing for Similarity Search: A Survey. *CoRR*, abs/1408.2, 2014.

[40] Yue, Yisong, Hong, Sue Ann Sa, and Guestrin, Carlos. Hierarchical exploration for accelerating contextual bandits. *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1895–1902, 2012.

[41] Zhang, Lijun, Yang, Tianbao, Jin, Rong, Xiao, Yichi, and Zhou, Zhi-hua. Online Stochastic Linear Optimization under One-bit Feedback. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 48, pp. 392–401, 2016.

Improved Strongly Adaptive Online Learning using Coin Betting

Kwang-Sung Jun UW-Madison
Francesco Orabona Stony Brook University
Stephen Wright UW-Madison
Rebecca Willett UW-Madison

Abstract

This paper describes a new parameter-free online learning algorithm for changing environments. In comparing against algorithms with the same time complexity as ours, we obtain a strongly adaptive regret bound that is a factor of at least $\sqrt{\log(T)}$ better, where T is the time horizon. Empirical results show that our algorithm outperforms state-of-the-art methods in learning with expert advice and metric learning scenarios.

1 Introduction

Machine learning has made heavy use of the i.i.d. assumption, but this assumption does not hold in many applications. For example, in online portfolio management, stock price trends can vary unexpectedly, and the ability to track changing trends and adapt to them are crucial in maximizing one's profit. Another example is seen in product reviews, where words describing product quality may change over time as products and customer's taste evolve. Keeping track of the changes in the metric describing the relationship between review text and rating is crucial for improving analysis and quality of recommendations.

We consider the problem of adapting to a changing environment in the online learning context. Let $\mathcal{D}$ be the decision space, $\mathcal{L}$ be loss functions that map $\mathcal{D}$ to $\mathbb{R}$, and T be the target time horizon. Let $\mathcal{A}$ be an online learning algorithm and $\mathcal{W} \subseteq \mathcal{D}$ be the set of comparator decisions. (Often, $\mathcal{W} = \mathcal{D}$.) We define the online learning problem in Figure 1. The usual goal of online learning is to find a strategy that compares favorably with the best fixed comparator in $\mathcal{W}$, in hindsight. Specifically, we seek a low value of the following (cumulative) static regret objective: $\text{Regret}_T^{\mathcal{A}} := \sum_{t=1}^T f_t(\mathbf{x}_t^{\mathcal{A}}) - \min_{\mathbf{w}\in\mathcal{W}} \sum_{t=1}^T f_t(\mathbf{w})$.

Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54.

At each time $t = 1, 2, \ldots, T$,
- The learner $\mathcal{A}$ picks a decision $\mathbf{x}_t^{\mathcal{A}} \in \mathcal{D}$.
- The environment reveals a loss function $f_t \in \mathcal{L}$.
- The learner $\mathcal{A}$ suffers loss $f_t(\mathbf{x}_t^{\mathcal{A}})$.

Figure 1: Online learning protocol

When the environment is changing, static regret is not a suitable measure, since it compares the learning strategy against a decision that is fixed. We need to make use of stronger notions of regret that allow for comparators to change over time. To define such notions, we introduce the notation $[T] := \{1, \ldots, T\}$ and $[A..B] = \{A, A+1, \ldots, B\}$. Daniely et al. [5] defined *strongly adaptive regret (SA-Regret)*, which requires an algorithm to have low (cumulative) regret over any contiguous time interval $I = [I_1..I_2] \subseteq [T]$.[1] Another notion, m-shift regret [10], measures instead the regret w.r.t. a comparator that changes at most m times in T time steps. Note that the SA-Regret is a stronger notion than the m-shift regret since the latter can be derived directly from the former [12, 5], as we show in our supplementary material. We define SA-Regret and m-shift regret precisely in Section 1.1.

Several generic online algorithms that adapt to a changing environment have been proposed recently. Rather than being designed for a specific learning problem, these are "meta algorithms" that take *any* online learning algorithm as a black-box and turn it into an adaptive one. We summarize the SA-Regret of existing meta algorithms in Table 2. In particular, the pioneering work of Hazan et al. [9] introduced the *adaptive regret*, that is a slightly weaker notion than the SA-Regret, and proposed two meta algorithms called FLH and AFLH. However, their SA-Regret depends on T rather than $|I|$. The SAOL approach of [5] improves the SA-Regret to $O\left(\sqrt{(I_2 - I_1)\log^2(I_2)}\right)$.

In this paper, we propose a new meta algorithm called *Coin Betting for Changing Environment* (CBCE) that

[1] Strongly adaptive regret is similar to the notion of adaptive regret introduced by [9], but emphasizes the dependency on the interval length $|I|$.

Algorithm	m-shift regret	Time	Agnostic to m
Fixed Share [10, 3]	$\sqrt{mT(\log N + \log T)}$	NT	✗
	$\sqrt{m^2T(\log N + \log T)}$	NT	✓
GeneralTracking⟨EXP⟩ [8]	$\sqrt{mT(\log N + m\log^2 T)}$	$NT\log T$	✓
	$\sqrt{mT(\log N + \log^2 T)}$	$NT\log T$	✗
($\gamma \in (0,1)$)	$\sqrt{\frac{1}{\gamma}mT(\log N + m\log T)}$	$NT^{1+\gamma}\log T$	✓
	$\sqrt{\frac{1}{\gamma}mT(\log N + \log T)}$	$NT^{1+\gamma}\log T$	✗
ATV [12]	$\sqrt{mT(\log N + \log T)}$	NT^2	✓
SAOL⟨MW⟩ [5]	$\sqrt{mT(\log N + \log^2 T)}$	$NT\log T$	✓
CBCE⟨CB⟩ (ours)	$\sqrt{mT(\log N + \log T)}$	$NT\log T$	✓

Table 1: m-shift regret bounds of LEA algorithms. Our proposed algorithm achieves the best regret among those with the same time complexity and does not need to know m. Each quantity omits constant factors. Agnostic to m means that an algorithm does not need to know the number m of switches in the best expert.

Algorithm	SA-Regret order	Time factor
FLH [9]	$\sqrt{T\log T}$	T
AFLH [9]	$\sqrt{T\log T\log(I_2 - I_1)}$	$\log T$
SAOL [5]	$\sqrt{(I_2 - I_1)\log^2(I_2)}$	$\log T$
CBCE (ours)	$\sqrt{(I_2 - I_1)\log(I_2)}$	$\log T$

Table 2: SA-Regret bounds of meta algorithms on $I \subseteq [T]$. Our proposed algorithm achieves the best SA-Regret. We show the part of the regret due to the meta algorithm only, not the black-box. The last column is the multiplicative factor in the time complexity introduced by the meta algorithm.

combines the sleeping bandits idea [2, 6] with the Coin Betting (CB) algorithm [13]. The SA-Regret of CBCE is better by a factor $\sqrt{\log(I_2)}$ than that of SAOL, as shown in Table 2. We present our extension of CB to sleeping bandits and prove its regret bound in Section 3. This result leads to the improved SA-Regret bound of CBCE in Section 4.

Our improved bound yields a number of improvements in various online learning problems. In describing these improvements, we denote by $\mathcal{M}\langle\mathcal{B}\rangle$ a complete algorithm assembled from meta algorithm $\mathcal{M}$ and black-box $\mathcal{B}$.

Consider the learning with expert advice (LEA) problem with N experts. CBCE with black-box CB (CBCE⟨CB⟩, in our notation) has the m-shift regret

$$O\sqrt{mT(\log N + \log T)}$$

and time complexity $O(NT\log T)$. This regret is a factor $\sqrt{\log T}$ better than those algorithms with the same time complexity. Although AdaNormalHedge.TV (ATV) and Fixed Share achieve the same regret, the former has larger time complexity, and the latter requires prior knowledge of the number of shifts m. We summarize the m-shift regret bounds of various algorithms in Table 1.

In Online Convex Optimization (OCO) with G-Lipschitz loss functions over a convex set $D \in \mathbb{R}^d$ of diameter B, online gradient descent has regret $O(BG\sqrt{T})$. CBCE with black-box OGD (CBCE⟨OGD⟩) then has the following SA-Regret:

$$O((BG + \sqrt{\log(I_2)})\sqrt{|I|}) ,$$

which improves by a factor $\sqrt{\log(I_2)}$ over SAOL⟨OGD⟩.

In Section 5, we compare CBCE empirically to a number of meta algorithms within a changing environment in two online learning problems: (i) LEA and (ii) Mahalanobis metric learning. We observe that CBCE outperforms the state-of-the-art methods in both tasks, thus confirming our theoretical findings.

1.1 Preliminaries

In this section we define some concepts that will be used in the rest of the paper.

A learner's SA-Regret is obtained by evaluating static regret on all (contiguous) time intervals $I = [I_1..I_2] \subseteq [T]$ of a given length τ. Specifically, the SA-Regret of an algorithm $\mathcal{A}$ at time T for length τ is

$$\text{SA-Regret}_T^{\mathcal{A}}(\tau) := \max_{I\subseteq[T]:|I|=\tau} \left(\sum_{t\in I} f_t(\mathbf{x}_t^{\mathcal{A}}) - \min_{\mathbf{w}\in\mathcal{W}} \sum_{t\in I} f_t(\mathbf{w})\right) . \quad (1)$$

We call an algorithm *strongly adaptive* if it has a low value of SA-Regret. We call $\mathbf{w}_{1:T} := \{\mathbf{w}_1, \dots, \mathbf{w}_T\}$ an *m-shift sequence* if it changes at most m times, that is, $\sum_{j=1}^{T-1} \mathbb{1}\{\mathbf{w}_j \neq \mathbf{w}_{j+1}\} \leq m$. We define

$$m\text{-Shift-Regret}_T^{\mathcal{A}} := \sum_{t=1}^{T} f_t(\mathbf{x}_t^{\mathcal{A}}) - \min_{\mathbf{w}_{1:T}\in\mathcal{W}^T \,:\, m\text{-shift seq.}} \sum_{t=1}^{T} f_t(\mathbf{w}_t) .$$

2 A Meta Algorithm for Changing Environments

Let $\mathcal{B}$ be a black-box online learning algorithm following the protocol in Figure 1. A trick commonly used in

designing a meta algorithm $\mathcal{M}$ for changing environments is to initiate a new instance of $\mathcal{B}$ at every time step [9, 8, 1]. That is, we run $\mathcal{B}$ independently for each interval J in $\{[t..\infty] \mid t = 1, 2, \ldots\}$. Denote by $\mathcal{B}_J$ the run of black-box $\mathcal{B}$ on interval J. A meta algorithm at time t combines the decisions from the runs $\{\mathcal{B}_J\}_{J \ni t}$ by weighted average. The key idea is that at time t, some of the outputs $\mathcal{B} \in \{\mathcal{B}_J\}_{J \ni t}$ are not based on any data prior to time $t' < t$, so that if the environment changes at time t', those outputs may be given a larger weight by the meta algorithm, allowing it to adapt more quickly to the change. This trick requires updating of t instances of the black-box algorithm at each time step t, leading to a factor-of-t increase in the time complexity. This factor can be reduced to $O(\log t)$ by restarting black-box algorithms on a carefully designed set of intervals such as the geometric covering intervals [5] (GC) or the data streaming technique [9, 8] (DS) that is a special case of a more general set of intervals considered in [14]. While both GC and DS achieve the same goal as we show in our supplementary material,[2] we use the former as our starting point for ease of exposition.

Geometric Covering Intervals. Define $\mathcal{J}_k := \{[(i \cdot 2^k) .. ((i+1) \cdot 2^k - 1)] : i \in \mathbb{N}\}$, $\forall k \in \{0, 1, \ldots\}$ to be the collection of intervals of length 2^k. The geometric covering intervals [5] are

$$\mathcal{J} := \bigcup_{k \in \{0,1,\ldots\}} \mathcal{J}_k \ .$$

That is, $\mathcal{J}$ is the set of intervals of doubling length, with intervals of size 2^k exactly partitioning the set $\mathbb{N} \backslash \{1, \ldots, 2^k - 1\}$, see Figure 2.

Define the set of intervals that includes time t as $\text{Active}(t) := \{J \in \mathcal{J} : t \in J\}$. One can easily show that $|\text{Active}(t)| = \lfloor \log_2(t) \rfloor + 1$. Since at most $O(\log(t))$ intervals contain any given time point t, the time complexity of the meta algorithm is a factor $O(\log(t))$ larger than that of the black-box $\mathcal{B}$.

The key result of the geometric covering intervals strategy is the following Lemma from [5], which shows that an arbitrary interval I can be partitioned into a sequence of smaller blocks whose lengths successively double, then successively halve.

Lemma 1. ([5, Lemma 5]) *Any interval $I \subseteq \mathbb{N}$ can be partitioned into two finite sequences of disjoint and consecutive intervals, denoted $\{J^{(-a)}, J^{(-a+1)}, \ldots, J^{(0)}\}$ and $\{J^{(1)}, J^{(2)}, \ldots, J^{(b)}\}$ where $\forall i \in [(-a)..b]$, we have $J^{(i)} \in \mathcal{J}$ and $J^{(i)} \subset I$, such that*

$$|J^{(-i)}|/|J^{(-i+1)}| \leq 1/2, \quad i = 1, 2, \ldots, a;$$

[2]Except for a subtle case, which we also discuss in our supplementary material.

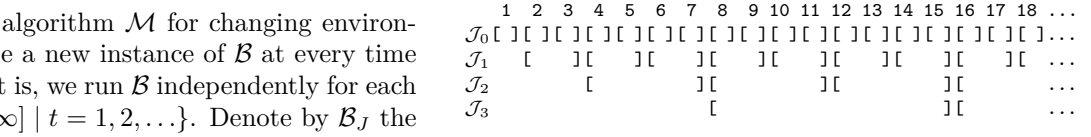


Figure 2: Geometric covering intervals. Each interval is denoted by `[ ]`.

$$|J^{(i+1)}|/|J^{(i)}| \leq 1/2, \quad i = 1, 2, \ldots, b-1 \ .$$

Regret Decomposition. We show now how to use the geometric covering intervals to decompose the SA-Regret of a complete algorithm $\mathcal{M}\langle\mathcal{B}\rangle$. We use the notation

$$R_I^{\mathcal{A}}(\mathbf{w}) := \sum_{t \in I} f_t(\mathbf{x}_t^{\mathcal{A}}) - \sum_{t \in I} f_t(\mathbf{w}) \ ,$$

and from (1) we see that $\text{SA-Regret}_T^{\mathcal{A}}(\tau) = \max_{I \subseteq [T]:|I|=\tau} \max_{\mathbf{w} \in \mathcal{W}} R_I^{\mathcal{A}}(\mathbf{w})$.

Denote by $\mathbf{x}_t^{\mathcal{B}_J}$ the decision from black-box $\mathcal{B}_J$ at time t and by $\mathbf{x}_t^{\mathcal{M}\langle\mathcal{B}\rangle}$ the combined decision of the meta algorithm. Since $\mathcal{M}\langle\mathcal{B}\rangle$ is a combination of a meta $\mathcal{M}$ and a black-box $\mathcal{B}$, its regret depends on both $\mathcal{M}$ and $\mathcal{B}$. Perhaps surprisingly, we can decompose the two sources of regret additively through the geometric covering $\mathcal{J}$, as we now describe. Choose some $I \subseteq [T]$, and let $\bigcup_{i=-a}^{b} J^{(i)}$ be the partition of I obtained from Lemma 1. Then, the regret of $\mathcal{M}\langle\mathcal{B}\rangle$ on I can be decomposed as follows:

$$\begin{aligned}
R_I^{\mathcal{M}\langle\mathcal{B}\rangle}(\mathbf{w}) &= \sum_{t \in I} \left(f_t(\mathbf{x}_t^{\mathcal{M}\langle\mathcal{B}\rangle}) - f_t(\mathbf{w}) \right) \\
&= \sum_{i=-a}^{b} \Bigg(\sum_{t \in J^{(i)}} f_t(\mathbf{x}_t^{\mathcal{M}\langle\mathcal{B}\rangle}) - f_t(\mathbf{x}_t^{\mathcal{B}_{J^{(i)}}}) + \\
&\qquad f_t(\mathbf{x}_t^{\mathcal{B}_{J^{(i)}}}) - f_t(\mathbf{w}) \Bigg) \\
&= \underbrace{\sum_{i=-a}^{b} \underbrace{\sum_{t \in J^{(i)}} \left(f_t(\mathbf{x}_t^{\mathcal{M}\langle\mathcal{B}\rangle}) - f_t(\mathbf{x}_t^{\mathcal{B}_{J^{(i)}}}) \right)}_{=:(\text{meta regret on } J^{(i)})} +}_{=:(\text{meta regret on } I)} \\
&\qquad \sum_{i=-a}^{b} \underbrace{\sum_{t \in J^{(i)}} \left(f_t(\mathbf{x}_t^{\mathcal{B}_{J^{(i)}}}) - f_t(\mathbf{w}) \right)}_{=:(\text{black-box regret on } J^{(i)})} \ . \qquad (2)
\end{aligned}$$

(We purposely use symbol J for intervals in $\mathcal{J}$ and I for a generic interval that is not necessarily in $\mathcal{J}$.) The black-box regret on $J = [J_1..J_2] \in \mathcal{J}$ is exactly the standard regret for $T = |J|$, since the black-box run $\mathcal{B}_J$ was started from time J_1. Thus, in order to prove that a meta algorithm $\mathcal{M}$ suffers low SA-Regret, we must show two things.

1. $\mathcal{M}$ has low regret on interval $J \in \mathcal{J}$.
2. The outer sum over i in (2) is small for both the meta and the black-box.

Daniely et al. [5] address the second issue above efficiently in their analysis. They show that if the black-box regret on $J^{(i)}$ scales like $\tilde{O}(\sqrt{|J^{(i)}|})$ (where $\tilde{O}$ ignores logarithmic factors), then the second double summation of (2) is[3] $\tilde{O}(\sqrt{|I|})$, which is perhaps the best one can hope for. The same holds true for the meta algorithm. Thus, it remains to focus on the first issue above, which is our main contribution.

In the next two sections, we show how to design our meta algorithm. In Section 3 we propose a novel method that incorporates sleeping bandits and the coin betting framework. Section 4 describes how our method can be used as a meta algorithm with strongly adaptive regret guarantees.

3 Coin Betting Meets Sleeping Experts

Our meta algorithm is an extension of the coin-betting framework [13] based on sleeping experts [2, 6]. It is parameter-free (there is no explicit learning rate) and has near-optimal regret. Our construction, described below, might also be of independent interest.

Sleeping Experts. In the learning with expert advice (LEA) framework, the decision set is $\mathcal{D} = \Delta^N$, an N-dimensional probability simplex of weights assigned to the experts. To distinguish LEA from the general online learning problem, we use notation $\mathbf{p}_t$ in place of $\mathbf{x}_t$ and h_t in place of f_t. Let $\boldsymbol{\ell}_t := (\ell_{t,1}, \dots, \ell_{t,N})^\top \in [0,1]^N$ be the vector of loss values of experts at time t that is provided by the environment. The learner's loss function is $h_t(\mathbf{p}) := \mathbf{p}^\top \boldsymbol{\ell}_t$.

Since $\mathbf{p} \in \mathcal{D}$ is a probability vector, the learner's decision can be viewed as hedging between the N alternatives. Let $\mathbf{e}_i$ be an indicator vector for dimension i; e.g., $\mathbf{e}_2 = (0, 1, 0, \dots, 0)^\top$. In this notation, the comparator set $\mathcal{W}$ is $\{\mathbf{e}_1, \dots, \mathbf{e}_N\}$. Thus, the learner aims to compete with a strategy that commits to a single expert for the entire time $[1..T]$.

The decision set may be nonconvex, for example, when $\mathcal{D} = \{\mathbf{e}_1, \dots, \mathbf{e}_N\}$ [4, Section 3]. In this case, no hedging is allowed; the learner must pick an expert. To choose an element of this set, one could first choose an element p_t from Δ^N, then make a decision $\mathbf{e}_i \in \mathcal{D}$ with probability $p_{t,i}$. For such a scheme, the regret guarantee is the same as in the standard LEA, but with *expected* regret.

[3] This is essentially the same argument as the "doubling trick" described in [4, Section 2.3]

Recall that each black-box run $\mathcal{B}_J$ is on a different interval J. The meta algorithm's role is to hedge bets over multiple black-box runs. Thus, it is natural to treat each run $\mathcal{B}_J$ as an *expert* and use an existing LEA algorithm to combine decisions from each expert $\mathcal{B}_J$. The loss incurred on run $\mathcal{B}_J$ is $\ell_{t,\mathcal{B}_J} := f_t(\mathbf{x}_t^{\mathcal{B}_J})$.

The challenge is that each expert $\mathcal{B}_J$ may not output decisions at time steps outside the interval J. This problem can be reduced to the *sleeping experts* problem studied in [2, 6], where experts are not required to provide decisions at every time step; see [12] for detail. We introduce a binary indicator variable $\mathcal{I}_{t,i} \in \{0, 1\}$, which is set to 1 if expert i is awake (that is, outputting a decision) at time t, and zero otherwise. Define $\boldsymbol{\mathcal{I}}_t := [\mathcal{I}_{t,1}, \mathcal{I}_{t,2}, \dots, \mathcal{I}_{t,N}]^\top$ where N can be countably infinite. Note that the algorithm is aware of $\boldsymbol{\mathcal{I}}_t$ and must assign zero weight to the experts that are sleeping: $\mathcal{I}_{t,i} = 0 \implies p_{t,i} = 0$. We would like to have a guarantee on the regret w.r.t. expert i, but only for the time steps where expert i is awake. Following [12], we aim to have a regret bound w.r.t. $\mathbf{u} \in \Delta^N$ as follows:

$$\mathrm{Regret}_T(\mathbf{u}) := \sum_{t=1}^{T} \sum_{i=1}^{N} \mathcal{I}_{t,i} u_i (\langle \boldsymbol{\ell}_t, \mathbf{p}_t \rangle - \ell_{t,i}) \,. \qquad (3)$$

If we set $\mathbf{u} = \mathbf{e}_j$ for some j, the above is simply regret w.r.t. expert j while that expert is awake. Furthermore, if $\mathcal{I}_{t,j} = 1$ for all $t \in [T]$, then it recovers the standard static regret in LEA.

Coin Betting for LEA. We consider the coin betting framework of Orabona and Pál [13], where one can construct an LEA algorithm based on the so-called *coin betting potential* function F_t. A player starts from the initial endowment 1. At each time step, the adversary tosses a coin arbitrarily, with the player deciding upon which side to bet (heads or tails). Then the outcome is revealed. The adversary can manipulate the weight of the coin in $[0, 1]$ as well, in a manner not known to the player before betting.

We encode a coin flip at iteration t as $\widetilde{g}_t \in [-1, 1]$ where positive (negative) means heads (tails), and $|\widetilde{g}_t|$ indicates the weight. Let Wealth_{t-1} be the total money the player possesses after time step $t-1$. The player decides which side and how much money to bet. We encode the player's decision as the signed betting fraction $\beta_t \in (-1, 1)$, where the positive (negative) sign indicates head (tail) and the absolute value $|\beta_t| \in [0, 1)$ indicates the fraction of his money to bet. Thus, the actual amount of betting is $w_t := \beta_t \mathrm{Wealth}_{t-1}$. Once the weighted coin flip $\widetilde{g}_t$ is revealed, the player's wealth changes: $\mathrm{Wealth}_t = \mathrm{Wealth}_{t-1} + \widetilde{g}_t \beta_t \mathrm{Wealth}_{t-1}$. The player makes (loses) money when the betted side is correct (wrong), and the amount of wealth change depends on both the flip

weight $|\widetilde{g}_t|$ and his betting amount $|\beta_t|$.

In the coin betting framework, the betting fraction β_t is determined by a potential function F_t, and we can simplify w_t as follows:

$$z_t := \sum_{\tau=1}^{t-1} \widetilde{g}_\tau$$
$$\beta_t(z_t) := \frac{F_t(z_t+1) - F_t(z_t-1)}{F_t(z_t+1) + F_t(z_t-1)} \tag{4}$$
$$w_t = \beta_t(z_t) \cdot \left(1 + \sum_{\tau=1}^{t-1} \widetilde{g}_\tau w_\tau\right) .$$

We use β_t in place of $\beta_t(\sum_{\tau=1}^{t-1} \widetilde{g}_\tau)$ when it is clear from the context. A sequence of coin betting potentials $F_1, F_2, \dots$ satisfies the following key condition (the complete list of conditions can be found in [13]): F_t must lower-bound the wealth of a player who bets by (4):

$$\forall t,\ F_t\left(\sum_{\tau=1}^{t} \widetilde{g}_\tau\right) \le 1 + \sum_{\tau=1}^{t} \widetilde{g}_\tau w_\tau . \tag{5}$$

This bound becomes useful in regret analysis. We emphasize that the term w_t is decided before $\widetilde{g}_t$ is revealed, yet the inequality (5) holds for any $\widetilde{g}_t \in [-1, 1]$.

Orabona and Pál [13] have devised a reduction of LEA to the simple coin betting problem described above. The idea is to instantiate a coin betting problem for each expert i where the signed coin flip $\widetilde{g}_{t,i}$ is set as a conditionally truncated regret w.r.t. expert i, rather than being set by an adversary. We denote by $\beta_{t,i}$ the betting fraction for expert i and by $w_{t,i}$ the amount of betting for expert i, $\forall i \in [N]$.

We apply the same treatment under the sleeping experts setting and propose a new algorithm **Sleeping CB**. Since some experts may not output a decision at time t, Sleeping CB requires a different definition of β_t. We define $S_{t,i} := 1 + \sum_{\tau=1}^{t-1} \mathcal{I}_{\tau,i}$ and define the following modifications of (4)

$$z_{t,i} := \sum_{\tau=1}^{t-1} \mathcal{I}_{t,i} \widetilde{g}_{\tau,i}$$
$$\beta_{t,i}(z_{t,i}) := \frac{F_{S_{t,i}}(z_{t,i}+1) - F_{S_{t,i}}(z_{t,i}-1)}{F_{S_{t,i}}(z_{t,i}+1) + F_{S_{t,i}}(z_{t,i}-1)} .$$

Further, we denote by $\boldsymbol{\pi}_{\boldsymbol{\mathcal{I}}_t}$ the prior $\boldsymbol{\pi}$ restricted to experts that are awake ($\mathcal{I}_{t,i} = 1$), and define $[x]_+ := \max\{x, 0\}$. Algorithm 1 specifies the Sleeping CB algorithm.

The regret of Sleeping CB is bounded in Theorem 1. (All proofs appear as supplementary material.) Unlike the standard CB, in which all the experts use F_t at time t, expert i in Sleeping CB uses $F_{S_{t,i}}$, which is different for each expert. For this reason, the proof of the CB regret in [13] does not transfer easily to the regret (3) of Sleeping CB, and a solution to it is the cornerstone of an improved strongly adaptive regret bound.

Algorithm 1 Sleeping CB

Input: Number of experts N, prior distribution $\boldsymbol{\pi} \in \Delta^N$
for $t = 1, 2, \dots$ **do**
  For each $i \in$ Active(t), set
    $w_{t,i} \leftarrow \beta_{t,i}(z_{t,i}) \cdot (1 + \sum_{\tau=1}^{t-1} z_{\tau,i} w_{\tau,i})$.
  For each $i \in$ Active(t), set $\widehat{p}_{t,i} \leftarrow \pi_i \mathcal{I}_{t,i} [w_{t,i}]_+$.
  Predict with $\mathbf{p}_t \leftarrow \begin{cases} \widehat{\mathbf{p}}_t / ||\widehat{\mathbf{p}}_t||_1 & \text{if } ||\widehat{\mathbf{p}}_t||_1 > 0 \\ \boldsymbol{\pi}_{\boldsymbol{\mathcal{I}}_t} & \text{if } ||\widehat{\mathbf{p}}_t||_1 = 0. \end{cases}$
  Receive loss vector $\boldsymbol{\ell}_t \in [0,1]^N$.
  The learner suffers loss $h_t(\mathbf{p}_t) = \langle \boldsymbol{\ell}_t, \mathbf{p}_t \rangle_{\boldsymbol{\mathcal{I}}_t}$.
  For each $i \in$ Active(t), set
    $\widetilde{g}_{t,i} \leftarrow \begin{cases} h_t(\mathbf{p}_t) - \ell_{t,i} & \text{if } w_{t,i} > 0 \\ [h_t(\mathbf{p}_t) - \ell_{t,i}]_+ & \text{if } w_{t,i} \le 0. \end{cases}$
end for

Theorem 1. (Regret of Sleeping CB) *Let $\{F_t\}_{t\ge1}$ be a sequence of potential functions that satisfies* (5)*. Assume that F_t is even (symmetric around zero) $\forall t \ge 1$. Suppose $\log F_{S_{T,i}}(x) \ge h_{S_{T,i}}(x) := c_1 \frac{x^2}{S_{T,i}} + c_{2,i}$ for some $c_1 > 0$ and $c_{2,i} \in \mathbb{R}$ for all $i \in [N]$. Then, Algorithm 1 satisfies*

$$\text{Regret}_T(\mathbf{u}) \le \sqrt{c_1^{-1} \cdot \left(\sum_{i=1}^N u_i S_{T,i}\right) \cdot \left(\text{KL}(\mathbf{u}||\boldsymbol{\pi}) - \sum_{i=1}^N u_i c_{2,i}\right)} .$$

Note that if $\mathbf{u} = \mathbf{e}_j$, then the regret scales with $S_{T,j}$, which is essentially the number of time steps at which expert j is awake.

Among multiple choices for the potential, we use the Krichevsky-Trofimov (KT) potential [13] that satisfies (5) (see [13] for the proof):

$$F_t(x) = \frac{2^t \cdot \Gamma(\delta+1) \cdot \Gamma(\frac{t+\delta+1}{2} + \frac{x}{2}) \Gamma(\frac{t+\delta+1}{2} - \frac{x}{2})}{\Gamma(\frac{\delta+1}{2})^2 \cdot \Gamma(t+\delta+1)},$$

where $\delta \ge 0$ is a time shift parameter that we set to 0 in this work. One can show that the betting fraction β_t defined in (4) for KT potential exhibits a simpler form: $\beta_t = \frac{\sum_{\tau=1}^{t-1} \widetilde{g}_\tau}{t+\delta}$ [13] and, for Sleeping CB, $\beta_t = \frac{\sum_{\tau=1}^{t-1} \mathcal{I}_{\tau,i} \widetilde{g}_{\tau,i}}{S_{t,i}+\delta}$. We present the regret of Algorithm 1 with the KT potential in Corollary 1.

Corollary 1. *Let $\delta = 0$. The regret of Algorithm 1 with the KT potential is*

$$\text{Regret}_T(\mathbf{u})$$

Algorithm 2 Coin Betting for Changing Environment (CBCE)

Input: A black-box algorithm $\mathcal{B}$ and a prior distribution $\boldsymbol{\pi} \in \Delta^{|\mathcal{J}|}$ over $\{\mathcal{B}_J \mid J \in \mathcal{J}\}$.
for $t = 1$ **to** T **do**
For each $J \in \text{Active}(t)$, set
$w_{t,\mathcal{B}_J} \leftarrow \beta_{t,\mathcal{B}_J}(z_{t,\mathcal{B}_J}) \cdot (1 + \sum_{\tau=1}^{t-1} z_{\tau,\mathcal{B}_J} w_{\tau,\mathcal{B}_J})$
Set $\widehat{p}_{t,\mathcal{B}_J} \leftarrow \pi_{\mathcal{B}_J} \mathcal{I}_{t,\mathcal{B}_J}[w_{t,\mathcal{B}_J}]_+$ for $J \in \text{Active}(t)$ and 0 for $J \notin \text{Active}(t)$.
Compute $\mathbf{p}_t \leftarrow \begin{cases} \widehat{\mathbf{p}}_t / ||\widehat{\mathbf{p}}_t||_1 & \text{if } ||\widehat{\mathbf{p}}_t||_1 > 0 \\ [\pi_{\mathcal{B}_J}]_{J \in \text{Active}(t)} & \text{if } ||\widehat{\mathbf{p}}_t||_1 = 0. \end{cases}$
The black-box run $\mathcal{B}_J$ picks a decision $\mathbf{x}_t^{\mathcal{B}_J} \in \mathcal{D}$, $\forall J \in \text{Active}(t)$.
The learner picks a decision $\mathbf{x}_t = \sum_{J \in \mathcal{J}} p_{t,\mathcal{B}_J} \mathbf{x}_t^{\mathcal{B}_J}$.
Each black-box run $\mathcal{B}_J$ that is awake ($J \in \text{Active}(t)$) suffers loss $\ell_{t,\mathcal{B}_J} := f_t(\mathbf{x}_t^{\mathcal{B}_J})$.
The learner suffers loss $f_t(\mathbf{x}_t)$.
For each $J \in \text{Active}(t)$, set
$\widetilde{g}_{t,\mathcal{B}_J} \leftarrow \begin{cases} f_t(\mathbf{x}_t) - \ell_{t,\mathcal{B}_J} & \text{if } w_{t,\mathcal{B}_J} > 0 \\ [f_t(\mathbf{x}_t) - \ell_{t,\mathcal{B}_J}]_+ & \text{if } w_{t,\mathcal{B}_J} \le 0. \end{cases}$
end for

$$\le \sqrt{2\left(\sum_{i=1}^{N} u_i S_{T,i}\right) \cdot \left(\mathrm{KL}(\mathbf{u}||\boldsymbol{\pi}) + \frac{1}{2}\ln(T) + 2\right)} \,.$$

4 Coping with a Changing Environment by Sleeping CB

In this section, we synthesize the results in Sections 2 and 3 to specify and analyze our algorithm. Recall that a meta algorithm must efficiently aggregate decisions from multiple black-box runs that are active at time t. We treat each black-box run as an expert. Since we run a black-box instance for each interval $J \in \mathcal{J}$, there are a countably infinite number of experts. Thus, one can use Sleeping CB (Algorithm 1) as the meta algorithm, with geometric covering intervals. The complete algorithm, which we call **Coin Betting for Changing Environment (CBCE)**, is shown in Algorithm 2.

We make use of the following assumption.

Assumption A1. *The loss function f_t is convex and maps to $[0,1]$, $\forall t \in \mathbb{N}$.*

Nonconvex loss functions can be accommodated by randomized decisions: We choose the decision $\mathbf{x}_t^{\mathcal{B}_J}$ from black-box $\mathcal{B}_J$ with probability $p_{t,\mathcal{B}_J}$. It is not difficult to show that the same regret bound holds, but now in expectation. When loss functions are unbounded, they can be scaled and restricted to $[0,1]$. Although this leads to possible nonconvexity, we can still obtain an expected regret bound from the randomized decision process just described.

We define our choice of prior $\bar{\boldsymbol{\pi}} \in \Delta^{|\mathcal{J}|}$ as follows:

$$\bar{\pi}_{\mathcal{B}_J} := Z^{-1}\left(\frac{1}{J_1^2(1 + \lfloor \log_2 J_1 \rfloor)}\right), \ \forall J \in \mathcal{J} \,, \quad (6)$$

where Z is a normalization factor. Note that $Z < \pi^2/6$ since there exist at most $1 + \lfloor \log_2 J_1 \rfloor$ distinct intervals starting at time J_1, so Z is less than $\sum_{t=1}^{\infty} t^{-2} = \pi^2/6$.

We bound the meta regret w.r.t. a black-box run $\mathcal{B}_J$ as follows.

Lemma 2. (Meta regret of CBCE) *Assume A1. Suppose we run CBCE (Algorithm 2) with a black-box algorithm $\mathcal{B}$, prior $\bar{\boldsymbol{\pi}}$, and $\delta = 0$. The meta regret of CBCE$\langle\mathcal{B}\rangle$ on interval $J = [J_1..J_2] \in \mathcal{J}$ is*

$$\sum_{t \in J} f_t(\mathbf{x}_t^{\text{CBCE}\langle\mathcal{B}\rangle}) - f_t(\mathbf{x}_t^{\mathcal{B}_J}) \le \sqrt{|J|\,(7\ln(J_2) + 5)} = O(\sqrt{|J| \log J_2}) \,.$$

We now present the bound on the SA-Regret $R_I^{\text{CBCE}\langle\mathcal{B}\rangle}(\mathbf{w})$ w.r.t. $\mathbf{w} \in \mathcal{W}$ on intervals $I \subseteq [T]$ that are not necessarily in $\mathcal{J}$.

Theorem 2. (SA-Regret of CBCE$\langle\mathcal{B}\rangle$) *Assume A1 and that the black-box algorithm $\mathcal{B}$ has regret $R_T^{\mathcal{B}}$ bounded by cT^{α}, where $\alpha \in (0,1)$. Let $I = [I_1..I_2]$. The SA-Regret of CBCE with black-box $\mathcal{B}$ on the interval I w.r.t. any $\mathbf{w} \in \mathcal{W}$ is bounded as follows:*

$$R_I^{\text{CBCE}\langle\mathcal{B}\rangle}(\mathbf{w}) \le \frac{4}{2^{\alpha}-1} c|I|^{\alpha} + 8\sqrt{|I|(7\ln(I_2) + 5)} = O(c|I|^{\alpha} + \sqrt{|I| \ln I_2}) \,.$$

For the standard LEA problem, one can run the algorithm CB with KT potential (equivalent to Sleeping CB with $\mathcal{I}_{t,i} = 1, \forall t, i$), which achieves static regret $O(\sqrt{T \log(NT)})$ [13]. Using CB as the black-box algorithm, the regret of CBCE$\langle\mathcal{B}\rangle$ on I is $R_I^{\text{CBCE}\langle\text{CB}\rangle}(\mathbf{w}) = O(\sqrt{|I| \log(NI_2)})$, and so SA-Regret$_T^{\text{CBCE}\langle\text{CB}\rangle}(|I|) = O(\sqrt{|I| \log(NT)})$. It follows that the m-shift regret of CBCE$\langle$CB$\rangle$ is $O(\sqrt{mT \log(NT)})$ using the technique presented our supplementary material.

As said above, our bound improves over the best known result with the same time complexity in [5]. The key ingredient that allows us to get a better bound is the Sleeping CB Algorithm 1, that achieves a better SA-Regret than the one of [5]. In the next section, we will show that the empirical results also confirm the theoretical gap of these two algorithms.

Discussion. Note that one can obtain the same result using the data streaming intervals (DS) [9, 8] in place of the geometric covering intervals (GC). Section F of our supplementary material elaborates on

this with a Lemma stating that DS induces a partition of an interval I in a very similar way to GC (a sequence of intervals of doubling lengths).

Our improved bound has another interesting implication. In designing strongly adaptive algorithms for LEA, there is a well known technique called "restarts" or "sleeping experts" that has time complexity $O(NT^2)$ [9, 12], and several studies used DS or GC to reduce the time complexity to $O(NT \log T)$ [9, 8, 5]. However, it was unclear whether it is possible to achieve both an m-shift regret of $O(\sqrt{mT(\log N + \log T)})$ and a time complexity of $O(NT \log T)$ without knowing m. Indeed, every study on m-shift regret with time $O(NT \log T)$ results in suboptimal m-shift regret bounds [5, 8, 9], to our knowledge. Furthermore, some studies (e.g., [12, Section 5]) speculated that perhaps applying the data streaming technique would increase its SA-Regret by a logarithmic factor. Our analysis implies that one can reduce the overall time complexity to $O(NT \log T)$ without sacrificing the order of SA-Regret and m-shift regret.

5 Experiments

We now turn to an empirical evaluation of algorithms for changing environments. We compare the performance of the meta algorithms under two online learning problems: (i) learning with expert advice (LEA) and (ii) metric learning (ML). We compare CBCE with SAOL [5] and AdaNormalHedge.TV (ATV) [12]. Although ATV was originally designed for LEA only, it is not hard to extend it to a meta algorithm and show that it has the same order of SA-Regret as CBCE using the same techniques.

For our empirical study, we replace the geometric covering intervals (GC) with the data streaming intervals (DS) [9, 8]. Let $u(t)$ be a number such that $2^{u(t)}$ is the largest power of 2 that divides t; e.g., $u(12) = 2$. The data streaming intervals are $\mathcal{J} = \{[t..(t+g\cdot 2^{u(t)}-1)] : t = 1, 2, \ldots\}$ for some $g \geq 1$. DS is an attractive alternative, unlike GC, (i) DS initiates one and only one black-box run at each time, and (ii) it is more flexible in that the parameter g can be increased to enjoy smaller regret in practice while increasing the time complexity by a constant factor.

For both ATV and CBCE, we set the prior $\boldsymbol{\pi}$ over the black-box runs as the uniform distribution. Note that this does not break the theoretical guarantees since the number of black-box runs are never actually infinite; we used $\bar{\boldsymbol{\pi}}$ (6) in Section 4 for ease of exposition.

5.1 Learning with Expert Advice (LEA)

We consider LEA with linear loss. That is, the loss function at time t is $h_t(\mathbf{p}) = \boldsymbol{\ell}_t^\top \mathbf{p}$. We draw linear loss $\boldsymbol{\ell}_t \in [0,1]^N, \forall t = 1, \ldots, 600$ for $N = 1000$ experts from Uniform$(0, 1)$ distribution. Then, for time $t \in [1, 200]$, we reduce loss of expert 1 by subtracting $1/2$ from its loss: $\ell_{t,1} \leftarrow [\ell_{t,1} - 1/2]_+$. For time $t \in [201, 400]$ and $t \in [401, 600]$, we perform the same for expert 2 and 3, respectively. Thus, the best expert is 1, 2, and 3 for time segment [1,200], [201,400], and [401,600], respectively. We use the data streaming intervals with $g = 2$. In all our experiments, DS with $g = 2$ outperforms GC while spending roughly the same time.

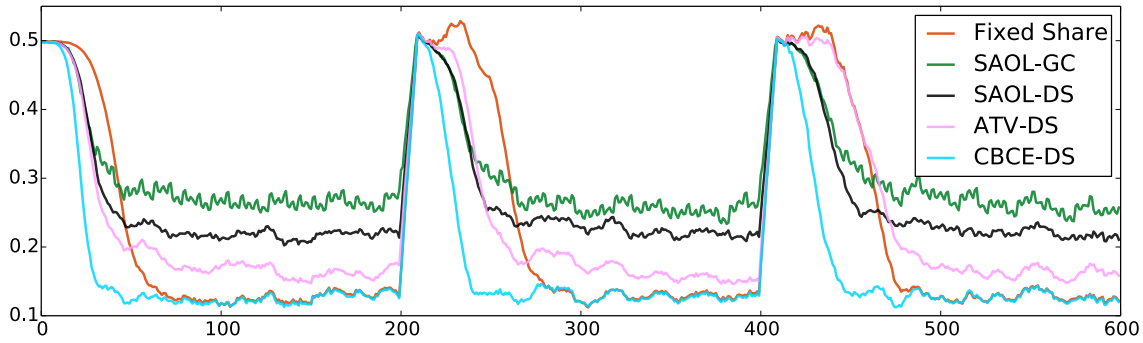
For each meta algorithm, we use the CB with KT potential [13] as the black-box algorithm. We warm-start each black-box run at time $t \geq 2$ by setting its prior to the decision $\mathbf{p}_{t-1}$ chosen by the meta algorithm at time step $t - 1$. We repeat the experiment 50 times and plot their average loss by computing moving mean with window size 10 in Figure 3(a). Overall, we observe that CBCE (i) catches up with the environmental shift faster than any other algorithms and (ii) has the lowest loss when the shift has settled down. ATV is the second best, outperforming SAOL. Note that SAOL with GC (SAOL-GC) tends to incur larger loss than the SAOL with DS. We observe that this is true for every meta algorithm, so we omit the result here to avoid clutter. We also run Fixed Share using the parameters recommended by Corollary 5.1 of [4], which requires to know the target time horizon $T = 600$ and the true number of switches $m = 2$. Such a strong assumption is often unrealistic in practice. We observe that Fixed Share is the slowest in adapting to the environmental changes. Nevertheless, Fixed Share remains attractive since (i) after the switch has settled down its loss is competitive to CBCE, and (ii) its time complexity is lower than other algorithms ($O(NT)$ rather than $O(NT \log T)$).

5.2 Metric Learning

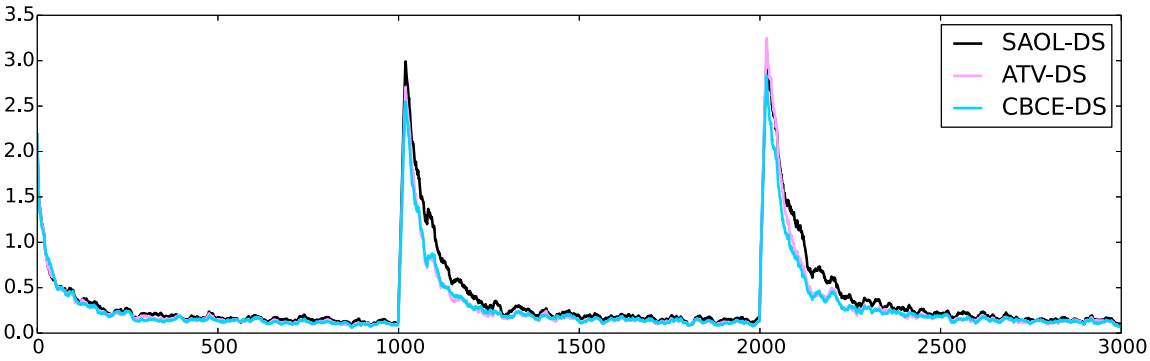
We consider the problem of learning squared Mahalanobis distance from pairwise comparisons using the mirror descent algorithm [11]. The data point at time t is $(\mathbf{z}_t^{(1)}, \mathbf{z}_t^{(2)}, y_t)$, where $y_t \in \{1, -1\}$ indicates whether or not $\mathbf{z}_t^{(1)} \in \mathbb{R}^d$ and $\mathbf{z}_t^{(2)} \in \mathbb{R}^d$ belongs to the same class. The goal is to learn a squared Mahalanobis distance parameterized by a positive semi-definite matrix $\mathbf{M}$ and a bias μ that have small loss $f_t([\mathbf{M}; \mu]) :=$

$$[1 - y_t(\mu - (\mathbf{z}_t^{(1)} - \mathbf{z}_t^{(2)})^\top \mathbf{M}(\mathbf{z}_t^{(1)} - \mathbf{z}_t^{(2)}))]_+ + \rho||\mathbf{M}||_* \,,$$

where μ is the bias parameter and $||\cdot||_*$ is the trace norm. Such a formulation encourages predicting y_t with large margin and low rank in $\mathbf{M}$. A learned matrix $\mathbf{M}$ that has low rank can be useful in a number of machine learning tasks; e.g., distance-based classifications, clusterings, and low-dimensional embeddings. We refer to [11] for details.



(a) Learning with expert advice



(b) Metric learning

Figure 3: Experiment results: Our method CBCE outperforms several baseline methods.

We create a scenario that exhibits shifts in the metric, which is inspired by [7]. Specifically, we create a mixture of three Gaussians in $\mathbb{R}^3$ whose means are well-separated, and mixture weights are .5, .3, and .2. We draw 2000 points from it while keeping a record of their memberships. We repeat this three times independently and concatenate these three vectors to have 2000 9-dimensional vectors. Finally, we append to each point a 16-dimensional vector filled with Gaussian noise to have 25-dimensional vectors. Such a construction implies that for each point there are three independent cluster memberships. We run each algorithm for 1500 time steps. For time 1 to 500, we randomly pick a pair of points from the data pool and assign $y_t = 1$ ($y_t = -1$) if the pair belongs to the same (different) cluster under the first clustering. For time 501 to 1000 (1001 to 1500), we perform the same but under the second (third) clustering. In this way, a learner faces tracking the change in metric, especially the important low-dimensional subspaces for each time segment.

Since the loss of the metric learning is unbounded, we scale the loss by multiplying $1/5$ and then capping it above at 1 as in [7]. Although the randomized decision discussed in Section 4 can be used to maintain the theoretical guarantee, we stick to the weighted average since the event that the loss being capped at 1 is rare in our experiments. As in our LEA experiment, we use the data streaming intervals with $g = 2$ and initialize each black-box algorithm with the decision of the meta algorithm at the previous time step. We repeat the experiment 50 times and plot their average loss in Figure 3(b) by moving mean with window size 20. We observe that CBCE and ATV both outperforms SAOL. This confirms the improved regret bound of CBCE and ATV.

6 Future Work

Among a number of interesting directions, we are interested in reducing the time complexity in the online learning within a changing environment. For LEA, Fixed Share has the best time complexity. However, Fixed Share is inherently not parameter-free; especially, it requires the knowledge of the number of shifts m. Achieving the best m-shift regret bound without knowing m or the best SA-Regret bound in time $O(NT)$ would be an interesting future work. The same direction is interesting for the online convex optimization (OCO) problem. It would be interesting if an OCO algorithm such as online gradient descent can have the same SA-Regret as CBCE$\langle$OGD$\rangle$ without paying extra order of computation.

Acknowledgements

This work was supported by NSF Award IIS-1447449 and NIH Award 1 U54 AI117924-01. The authors thank András György for providing constructive feedback and Kristjan Greenewald for providing the metric learning code.

References

[1] D. Adamskiy, W. M. Koolen, A. Chernov, and V. Vovk, "A Closer Look at Adaptive Regret," in *Proceedings of the International Conference on Algorithmic Learning Theory (ALT)*, 2012, pp. 290–304.

[2] A. Blum and A. Blum, "Empirical Support for Winnow and Weighted-Majority Algorithms: Results on a Calendar Scheduling Domain," *Machine Learning*, vol. 26, no. 1, pp. 5–23, 1997.

[3] N. Cesa-Bianchi, P. Gaillard, G. Lugosi, and G. Stoltz, "Mirror descent meets fixed share (and feels no regret)," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 980–988.

[4] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[5] A. Daniely, A. Gonen, and S. Shalev-Shwartz, "Strongly Adaptive Online Learning," *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1–18, 2015.

[6] Y. Freund, R. E. Schapire, Y. Singer, and M. K. Warmuth, "Using and combining predictors that specialize," *Proceedings of the ACM symposium on Theory of computing (STOC)*, vol. 37, no. 3, pp. 334–343, 1997.

[7] K. Greenewald, S. Kelley, and A. O. Hero, "Dynamic metric learning from pairwise comparisons," *54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2016.

[8] A. György, T. Linder, and G. Lugosi, "Efficient tracking of large classes of experts," *IEEE Transactions on Information Theory*, vol. 58, no. 11, pp. 6709–6725, 2012.

[9] E. Hazan and C. Seshadhri, "Adaptive Algorithms for Online Decision Problems," *IBM Research Report*, vol. 10418, pp. 1–19, 2007.

[10] M. Herbster and M. K. Warmuth, "Tracking the Best Expert," *Mach. Learn.*, vol. 32, no. 2, pp. 151–178, 1998.

[11] G. Kunapuli and J. Shavlik, "Mirror descent for metric learning: A unified approach," in *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Database (ECML/PKDD)*, 2012, pp. 859–874.

[12] H. Luo and R. E. Schapire, "Achieving All with No Parameters: AdaNormalHedge," in *Proceedings of the Conference on Learning Theory (COLT)*, 2015, pp. 1286–1304.

[13] F. Orabona and D. Pal, "Coin betting and parameter-free online learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 577–585.

[14] J. Veness, M. White, M. Bowling, and A. György, "Partition tree weighting," in *Proceedings of the 2013 Data Compression Conference*. IEEE Computer Society, 2013, pp. 321–330.

Human Memory Search as Initial-Visit Emitting Random Walk

Kwang-Sung Jun[*], **Xiaojin Zhu**[†], **Timothy Rogers**[‡]
[*]Wisconsin Institute for Discovery, [†]Department of Computer Sciences, [‡]Department of Psychology
University of Wisconsin-Madison
kjun@discovery.wisc.edu, jerryzhu@cs.wisc.edu, ttrogers@wisc.edu

Zhuoran Yang
Department of Mathematical Sciences
Tsinghua University
yzr11@mails.tsinghua.edu.cn

Ming Yuan
Department of Statistics
University of Wisconsin-Madison
myuan@stat.wisc.edu

Abstract

Imagine a random walk that outputs a state only when visiting it for the first time. The observed output is therefore a repeat-censored version of the underlying walk, and consists of a permutation of the states or a prefix of it. We call this model initial-visit emitting random walk (INVITE). Prior work has shown that the random walks with such a repeat-censoring mechanism explain well human behavior in memory search tasks, which is of great interest in both the study of human cognition and various clinical applications. However, parameter estimation in INVITE is challenging, because naive likelihood computation by marginalizing over infinitely many hidden random walk trajectories is intractable. In this paper, we propose the first efficient maximum likelihood estimate (MLE) for INVITE by decomposing the censored output into a series of absorbing random walks. We also prove theoretical properties of the MLE including identifiability and consistency. We show that INVITE outperforms several existing methods on real-world human response data from memory search tasks.

1 Human Memory Search as a Random Walk

A key goal for cognitive science has been to understand the mental structures and processes that underlie human semantic memory search. Semantic fluency has provided the central paradigm for this work: given a category label as a cue (e.g. animals, vehicles, etc.) participants must generate as many example words as possible in 60 seconds without repetition. The task is useful because, while exceedingly easy to administer, it yields rich information about human semantic memory. Participants do not generate responses in random order but produce "bursts" of related items, beginning with the highly frequent and prototypical, then moving to subclusters of related items. This ordinal structure sheds light on associative structures in memory: retrieval of a given item promotes retrieval of a related item, and so on, so that the temporal proximity of items in generated lists reflects the degree to which the two items are related in memory [14, 5]. The task also places demands on other important cognitive contributors to memory search: for instance, participants must retain a mental trace of previously-generated items and use it to refrain from repetition, so that the task draws upon working memory and cognitive control in addition to semantic processes. For these reasons the task is a central tool in all commonly-used metrics for diagnosing cognitive dysfunction (see e.g. [6]). Performance is generally sensitive to a variety of neurological disorders [19], but different syndromes also give rise to different patterns of impairment, making it useful for diagnosis [17]. For these reasons the task has been widely employed both in basic science and applied health research.

Nevertheless, the representations and processes that support category fluency remain poorly understood. Beyond the general observation that responses tend to be clustered by semantic relatedness,

it is not clear what ordinal structure in produced responses reveals about the structure of human semantic memory, in either healthy or disordered populations. In the past few years researchers in cognitive science have begun to fill this gap by considering how search models from other domains of science might explain patterns of responses observed in fluency tasks [12, 13, 15]. We review related works in Section 4.

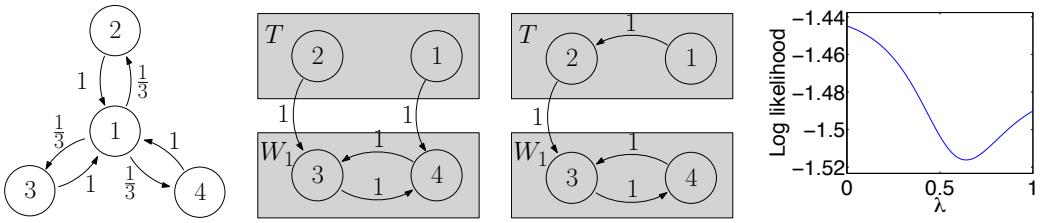
In the current work we build on these advances by considering, not how search might operate on a pre-specified semantic representation, but rather how the representation itself can be learned from data (i.e., human-produced semantic fluency lists) given a specified model of the list-generation process. Specifically, we model search as a random walk on a set of states (e.g. words) where the transition probability indicates the strength of association in memory, and with the further constraint that node labels are only generated when the node is first visited. Thus, repeated visits are censored in the output. We refer to this generative process as the *initial-visit emitting* (INVITE) random walk. The repeat-censoring mechanism of INVITE was first employed in Abbott et al. [1]. However, their work did not provide a tractable method to compute the likelihood nor to estimate the transition matrix from the fluency responses. The problem of estimating the underlying Markov chain from the lists so produced is nontrivial because once the first two items in a list have been produced there may exist infinitely many pathways that lead to production of the next item. For instance, consider the produced sequence "dog" $\rightarrow$ "cat" $\rightarrow$ "goat" where the underlying graph is fully connected. Suppose a random walk visits "dog" then "cat". The walk can then visit "dog" and "cat" arbitrarily many times before visiting "goat"; there exist infinitely many walks that outputs the given sequence. How can the transition probabilities of the underlying random walk be learned?

A solution to this problem would represent a significant advance from prior works that estimate parameters from a separate source such as a standard text corpus [13]. First, one reason for verbal fluency's enduring appeal has been that the task appears to reveal important semantic structure that may not be discoverable by other means. It is not clear that methods for estimating semantic structure based on another corpus do a very good job at modelling the structure of human semantic representations generally [10], or that they would reveal the same structures that govern behavior specifically in this widely-used fluency task. Second, the representational structures employed can vary depending upon the fluency category. For instance, the probability of producing "chicken" after "goat" will differ depending on whether the task involves listing "animals", "mammals", or "farm animals". Simply estimating a single structure from the same corpus will not capture these task-based effects. Third, special populations, including neurological patients and developing children, may generate lists from quite different underlying mental representations, which cannot be independently estimated from a standard corpus.

In this work, we make two important contributions on the INVITE random walk. First, we propose a tractable way to compute the INVITE likelihood. Our key insight in computing the likelihood is to turn INVITE into a series of absorbing random walks. This formulation allows us to leverage the fundamental matrix [7] and compute the likelihood in polynomial time. Second, we show that the MLE of INVITE is consistent, which is non-trivial given that the convergence of the log likelihood function is not uniform. We formally define INVITE and present the two main contributions as well as an efficient optimization method to estimate the parameters in Section 2. In Section 3, we apply INVITE to both toy data and real-world fluency data. On toy data our experiments empirically confirm the consistency result. On actual human responses from verbal fluency INVITE outperforms off-the-shelf baselines. The results suggest that INVITE may provide a useful tool for investigating human cognitive functions.

2 The INVITE Random Walk

INVITE is a probabilistic model with the following generative story. Consider a random walk on a set of n states S with an initial distribution $\boldsymbol{\pi} > 0$ (entry-wise) and an arbitrary transition matrix $\mathbf{P}$ where P_{ij} is the probability of jumping from state i to j. A surfer starts from a random initial state drawn from $\boldsymbol{\pi}$. She outputs a state if it is the first time she visits that state. Upon arriving at an already visited state, however, she does not output the state. The random walk continues indefinitely. Therefore, the output consists of states in the order of their first-visit; the underlying entire walk trajectory is hidden. We further assume that the time step of each output is unobserved. For example, consider the random walk over four states in Figure 1(a). If the underlying random walk takes the trajectory $(1, 2, 1, 3, 1, 2, 1, 4, 1, \ldots)$, the observation is $(1, 2, 3, 4)$.



(a) (b) (c) (d)

Figure 1: (a-c) Example Markov chains (d) Example nonconvexity of the INVITE log likelihood

We say that the observation produced by INVITE is a *censored* list since non-initial visits are censored. It is easy to see that a censored list is a permutation of the n states or a prefix thereof (more on this later). We denote a censored list by $\mathbf{a} = (a_1, a_2, \ldots, a_M)$ where $M \leq n$. A censored list is not Markovian since the probability of a transition in censored list depends on the whole history rather than just the current state. It is worth noting that INVITE is distinct from Broder's algorithm for generating random spanning trees [4], or the self-avoiding random walk [9], or cascade models of infection. We discuss the technical difference to related works in Section 4.

We characterize the type of output INVITE is capable of producing, given that the underlying uncensored random walk continues indefinitely. A state s is said to be *transient* if a random walk starting from s has nonzero probability of not returning to itself in finite time and *recurrent* if such probability is zero. A set of states A is *closed* if a walk cannot exit A; i.e., if $i \in A$ and $j \notin A$, then a random walk from i cannot reach j. A set of states B is *irreducible* if there exists a path between every pair of states in B; i.e., if $i, j \in B$, then a random walk from i can reach j. Define $[M] = \{1, 2, \ldots, M\}$. We use $a_{1:M}$ as a shorthand for $a_1, \ldots, a_M$. Theorem 1 states that a finite state Markov chain can be uniquely decomposed into disjoint sets, and Theorem 2 states what a censored list should look like. All proofs are in the supplementary material.

Theorem 1. *[8] If the state space S is finite, then S can be written as a disjoint union $T \cup W_1 \cup \ldots \cup W_K$, where T is a set of transient states that is possibly empty and each W_k, $k \in [K]$, is a nonempty closed irreducible set of recurrent states.*

Theorem 2. *Consider a Markov chain $\mathbf{P}$ with the decomposition $S = T \cup W_1 \cup \ldots \cup W_K$ as in Theorem 1. A censored list $\mathbf{a} = (a_{1:M})$ generated by INVITE on $\mathbf{P}$ has zero or more transient states, followed by all states in one and only one closed irreducible set. That is, $\exists \ell \in [M]$ s.t. $\{a_{1:\ell-1}\} \subseteq T$ and $\{a_{\ell:M}\} = W_k$ for some $k \in [K]$.*

As an example, when the graph is fully connected INVITE is capable of producing all $n!$ permutations of the n states as the censored lists. As another example, in Figure 1 (b) and (c), both chains have two transient states $T = \{1, 2\}$ and two recurrent states $W_1 = \{3, 4\}$. (b) has no path that visits both 1 and 2, and thus every censored list must be a prefix of a permutation. However, (c) has a path that visits both 1 and 2, thus can generate (1,2,3,4), a full permutation.

In general, each INVITE run generates a permutation of n states, or a prefix of a permutation. Let $\mathrm{Sym}(n)$ be the symmetric group on $[n]$. Then, the data space $\mathcal{D}$ of censored lists is $\mathcal{D} \equiv \{(a_{1:k}) \mid \mathbf{a} \in \mathrm{Sym}(n), k \in [n]\}$.

2.1 Computing the INVITE likelihood

Learning and inference under the INVITE model is challenging due to its likelihood function. A naive method to compute the probability of a censored list $\mathbf{a}$ given $\boldsymbol{\pi}$ and $\mathbf{P}$ is to sum over all uncensored random walk trajectories $\mathbf{x}$ which produces $\mathbf{a}$: $\mathbb{P}(\mathbf{a}; \boldsymbol{\pi}, \mathbf{P}) = \sum_{\mathbf{x} \text{ produces } \mathbf{a}} \mathbb{P}(\mathbf{x}; \boldsymbol{\pi}, \mathbf{P})$. This naive computation is intractable since the summation can be over an infinite number of trajectories $\mathbf{x}$'s that might have produced the censored list $\mathbf{a}$. For example, consider the censored list $\mathbf{a} = (1, 2, 3, 4)$ generated from Figure 1(a). There are infinite uncensored trajectories to produce $\mathbf{a}$ by visiting states 1 and 2 arbitrarily many times before visiting state 3, and later state 4.

The likelihood of $\boldsymbol{\pi}$ and $\mathbf{P}$ on a censored list $\mathbf{a}$ is

$$\mathbb{P}(\mathbf{a}; \boldsymbol{\pi}, \mathbf{P}) = \begin{cases} \pi_{a_1} \prod_{k=1}^{M-1} \mathbb{P}(a_{k+1} \mid a_{1:k}; \mathbf{P}) & \text{if } \mathbf{a} \text{ cannot be extended} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Note we assign zero probability to a censored list that is not completed yet, since the underlying random walk must run forever. We say a censored list $\mathbf{a}$ is *valid* (*invalid*) under $\boldsymbol{\pi}$ and $\mathbf{P}$ if $\mathbb{P}(\mathbf{a}; \boldsymbol{\pi}, \mathbf{P}) > 0 \ (= 0)$.

We first review the *fundamental matrix* in the absorbing random walk. A state that transits to itself with probability 1 is called an *absorbing state*. Given a Markov chain $\mathbf{P}$ with absorbing states, we can rearrange the states into $\mathbf{P}' = \begin{pmatrix} \mathbf{Q} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$, where $\mathbf{Q}$ is the transition between the nonabsorbing states, $\mathbf{R}$ is the transition from the nonabsorbing states to absorbing states, and the rest trivially represent the absorbing states. Theorem 3 presents the *fundamental matrix*, the essential tool for the tractable computation of the INVITE likelihood.

Theorem 3. *[7] The fundamental matrix of the Markov chain $\mathbf{P}'$ is $\mathbf{N} = (\mathbf{I} - \mathbf{Q})^{-1}$. N_{ij} is the expected number of times that a chain visits state j before absorption when starting from i. Furthermore, define $\mathbf{B} = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{R}$. Then, B_{ik} is the probability of a chain starting from i being absorbed by k. In other words, $B_{i\cdot}$ is the absorption distribution of a chain starting from i.*

As a tractable way to compute the likelihood, we propose a novel formulation that turns an INVITE random walk into a series of absorbing random walks. Although INVITE itself is not an absorbing random walk, each segment that produces the next item in the censored list can be modeled as one. That is, for each $k = 1 \dots M-1$ consider the segment of the uncensored random walk starting from the previous output a_k until the next output a_{k+1}. For this segment, we construct an absorbing random walk by keeping $a_{1:k}$ nonabsorbing and turning the rest into the absorbing states. A random walk starting from a_k is eventually absorbed by a state in $S \setminus \{a_{1:k}\}$. The probability of being absorbed by a_{k+1} is exactly the probability of outputting a_{k+1} after outputting $a_{1:k}$ in INVITE. Formally, we construct an absorbing random walk $\mathbf{P}^{(k)}$:

$$\mathbf{P}^{(k)} = \begin{pmatrix} \mathbf{Q}^{(k)} & \mathbf{R}^{(k)} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}, \tag{2}$$

where the states are ordered as $a_{1:M}$. Corollary 1 summarizes our computation of the INVITE likelihood.

Corollary 1. *The k-th step INVITE likelihood for $k \in [M-1]$ is*

$$\mathbb{P}(a_{k+1} \mid a_{1:k}, \mathbf{P}) = \begin{cases} [(\mathbf{I} - \mathbf{Q}^{(k)})^{-1}\mathbf{R}^{(k)}]_{k1} & \text{if } (\mathbf{I} - \mathbf{Q}^{(k)})^{-1} \text{ exists} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Suppose we observe m independent realizations of INVITE: $D_m = \left\{ \left(a_1^{(1)}, ..., a_{M_1}^{(1)}\right), ..., \left(a_1^{(m)}, ..., a_{M_m}^{(m)}\right) \right\}$, where M_i is the length of the i-th censored list. Then, the INVITE log likelihood is $\ell(\boldsymbol{\pi}, \mathbf{P}; D_m) = \sum_{i=1}^{m} \log \mathbb{P}(\mathbf{a}^{(i)}; \boldsymbol{\pi}, \mathbf{P})$.

2.2 Consistency of the MLE

Identifiability is an essential property for a model to be consistent. Theorem 4 shows that allowing self-transitions in $\mathbf{P}$ cause INVITE to be unidentifiable. Then, Theorem 5 presents a remedy. The proof for both theorems are presented in our supplementary material. Let $\text{diag}(\mathbf{q})$ be a diagonal matrix whose i-th diagonal entry is q_i.

Theorem 4. *Let $\mathbf{P}$ be an $n \times n$ transition matrix without any self-transition ($P_{ii} = 0, \forall i$), and $\mathbf{q} \in [0,1)^n$. Define $\mathbf{P}' = \text{diag}(\mathbf{q}) + (\mathbf{I} - \text{diag}(\mathbf{q}))\mathbf{P}$, a scaled transition matrix with self-transition probabilities $\mathbf{q}$. Then, $\mathbb{P}(\mathbf{a}; \boldsymbol{\pi}, \mathbf{P}) = \mathbb{P}(\mathbf{a}; \boldsymbol{\pi}, \mathbf{P}')$, for every censored list $\mathbf{a}$.*

For example, consider a censored list $\mathbf{a} = (1, j)$ where $j \neq 1$. Using the fundamental matrix, $\mathbb{P}(a_2 \mid a_1; \mathbf{P}) = (1 - P_{11})^{-1} P_{1j} = (\sum_{j' \neq 1} P_{1j'})^{-1} P_{1j} = (\sum_{j' \neq 1} cP_{1j'})^{-1} cP_{1j}, \forall c$. This implies that multiplying a constant c to P_{1j} for all $j \neq 1$ and renormalizing the first row $\mathbf{P}_{1\cdot}$ to sum to 1 does not change the likelihood.

Theorem 5. *Assume the initial distribution $\boldsymbol{\pi} > 0$ elementwise. In the space of transition matrices $\mathbf{P}$ without self-transitions, INVITE is identifiable.*

Let $\Delta^{n-1} = \{\mathbf{p} \in \mathbb{R}^n \mid p_i \geq 0, \forall i, \sum_i p_i = 1\}$ be the probability simplex. For brevity, we pack the parameters of INVITE into one vector $\boldsymbol{\theta}$ as follows: $\boldsymbol{\theta} \in \boldsymbol{\Theta} = \{(\boldsymbol{\pi}^\top, \mathbf{P}_{1\cdot}, \dots, \mathbf{P}_{n\cdot})^\top \mid \boldsymbol{\pi}, \mathbf{P}_{i\cdot} \in \Delta^{n-1}, P_{ii} = 0, \forall i\}$. Let $\boldsymbol{\theta}^* = (\boldsymbol{\pi}^{*\top}, \mathbf{P}^*_{1\cdot}, \dots, \mathbf{P}^*_{n\cdot})^\top \in \boldsymbol{\Theta}$ be the true model. Given a set of m censored lists D_m generated from $\boldsymbol{\theta}^*$, the average log likelihood function and its pointwise limit are

$$\widehat{\mathcal{Q}}_m(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \log \mathbb{P}(\mathbf{a}^{(i)}); \boldsymbol{\theta}) \qquad \text{and} \qquad \mathcal{Q}^*(\boldsymbol{\theta}) = \sum_{\mathbf{a}\in\mathcal{D}} \mathbb{P}(\mathbf{a}; \boldsymbol{\theta}^*) \log \mathbb{P}(\mathbf{a}; \boldsymbol{\theta}). \tag{4}$$

For brevity, we assume that the true model $\boldsymbol{\theta}^*$ is strongly connected; the analysis can be easily extended to remove it. Under the Assumption A1, Theorem 6 states the consistency result.

Assumption A1. *Let $\boldsymbol{\theta}^* = (\boldsymbol{\pi}^{*\top}, \mathbf{P}^*_{1\cdot}, \ldots, \mathbf{P}^*_{n\cdot})^\top \in \boldsymbol{\Theta}$ be the true model. $\boldsymbol{\pi}^*$ has no zero entries. Furthermore, $\mathbf{P}^*$ is strongly connected.*

Theorem 6. *Assume A1. The MLE of INVITE $\widehat{\boldsymbol{\theta}}_m \equiv \max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \widehat{\mathcal{Q}}_m(\boldsymbol{\theta})$ is consistent.*

We provide a sketch here. The proof relies on Lemma 6 and Lemma 2 that are presented in our supplementary material. Since $\boldsymbol{\Theta}$ is compact, the sequence $\{\widehat{\boldsymbol{\theta}}_m\}$ has a convergent subsequence $\{\widehat{\boldsymbol{\theta}}_{m_j}\}$. Let $\boldsymbol{\theta}' = \lim_{j\to\infty} \widehat{\boldsymbol{\theta}}_{m_j}$. Since $\widehat{\mathcal{Q}}_{m_j}(\boldsymbol{\theta}^*) \le \widehat{\mathcal{Q}}_{m_j}(\widehat{\boldsymbol{\theta}}_{m_j})$,

$$\mathcal{Q}^*(\boldsymbol{\theta}^*) = \lim_{j\to\infty} \widehat{\mathcal{Q}}_{m_j}(\boldsymbol{\theta}^*) \le \lim_{j\to\infty} \widehat{\mathcal{Q}}_{m_j}(\widehat{\boldsymbol{\theta}}_{m_j}) = \mathcal{Q}^*(\boldsymbol{\theta}'),$$

where the last equality is due to Lemma 6. By Lemma 2, $\boldsymbol{\theta}^*$ is the unique maximizer of $\mathcal{Q}^*$, which implies $\boldsymbol{\theta}' = \boldsymbol{\theta}^*$. Note that the subsequence was chosen arbitrarily. Since every convergent subsequence converges to $\boldsymbol{\theta}^*$, $\widehat{\boldsymbol{\theta}}_m$ converges to $\boldsymbol{\theta}^*$.

2.3 Parameter Estimation via Regularized Maximum Likelihood

We present a regularized MLE (RegMLE) of INVITE. We first extend the censored lists that we consider. Now we allow the underlying walk to terminate after finite steps because in real-world applications the observed censored lists are often truncated. That is, the underlying random walk can be stopped before exhausting every state the walk could visit. For example, in verbal fluency, participants have limited time to produce a list. Consequently, we use the *prefix likelihood*

$$\mathcal{L}(\mathbf{a}; \boldsymbol{\pi}, \mathbf{P}) = \pi_{a_1} \prod_{k=1}^{M-1} \mathbb{P}(a_{k+1} \mid a_{1:k}; \mathbf{P}). \tag{5}$$

We find the RegMLE by maximizing the prefix log likelihood plus a regularization term on $\boldsymbol{\pi}, \mathbf{P}$. Note that, $\boldsymbol{\pi}$ and $\mathbf{P}$ can be separately optimized. For $\boldsymbol{\pi}$, we place a Dirichlet prior and find the maximum a posteriori (MAP) estimator $\widehat{\boldsymbol{\pi}}$ by $\widehat{\pi}_j \propto \sum_{i=1}^m \mathbb{1}_{a_1^{(i)}=j} + C_{\boldsymbol{\pi}}, \forall j$.

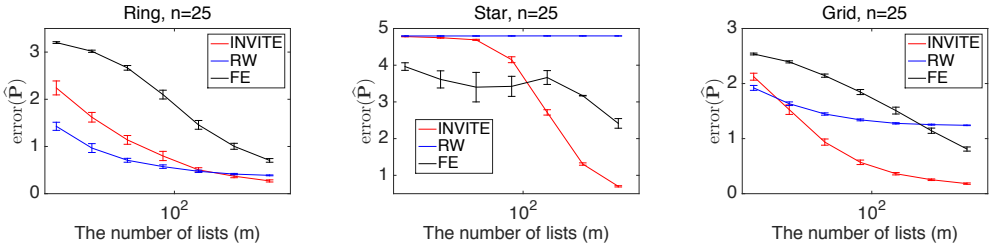
Directly computing the RegMLE of $\mathbf{P}$ requires solving a constrained optimization problem, because the transition matrix $\mathbf{P}$ must be row stochastic. We re-parametrize $\mathbf{P}$ which leads to a more convenient unconstrained optimization problem. Let $\boldsymbol{\beta} \in \mathbb{R}^{n\times n}$. We exponentiate $\boldsymbol{\beta}$ and row-normalize it to derive $\mathbf{P}$: $P_{ij} = e^{\beta_{ij}} / \sum_{j'=1}^n e^{\beta_{ij'}}, \forall i, j$. We fix the diagonal entries of $\boldsymbol{\beta}$ to $-\infty$ to disallow self-transitions. We place squared ℓ_2 norm regularizer on $\boldsymbol{\beta}$ to prevent overfitting. The unconstrained optimization problem is:

$$\min_{\boldsymbol{\beta}} \quad -\sum_{i=1}^m \sum_{k=1}^{M_i-1} \log \mathbb{P}(a_{k+1}^{(i)} \mid a_{1:k}^{(i)}; \boldsymbol{\beta}) + \tfrac{1}{2} C_{\boldsymbol{\beta}} \sum_{i\neq j} \beta_{ij}^2 \quad , \tag{6}$$

where $C_{\boldsymbol{\beta}} > 0$ is a regularization parameter. We provide the derivative of the prefix log likelihood w.r.t. $\boldsymbol{\beta}$ in our supplementary material. We point out that the objective function of (6) is not convex in $\boldsymbol{\beta}$ in general. Let $n = 5$ and suppose we observe two censored lists $(5, 4, 3, 1, 2)$ and $(3, 4, 5, 1, 2)$. We found with random starts two different local optima $\boldsymbol{\beta}^{(1)}$ and $\boldsymbol{\beta}^{(2)}$ of (6). We plot the prefix log likelihood of $(1-\lambda)\boldsymbol{\beta}^{(1)} + \lambda\boldsymbol{\beta}^{(2)}$, where $\lambda \in [0, 1]$ in Figure 1(d). Nonconvexity of this 1D slice implies nonconvexity of the prefix log likelihood surface in general.

Efficient Optimization using Averaged Stochastic Gradient Descent Given a censored list $\mathbf{a}$ of length M, computing the derivative of $\mathbb{P}(a_{k+1} \mid a_{1:k})$ w.r.t. $\boldsymbol{\beta}$ takes $O(k^3)$ time for matrix inversion. There are n^2 entries in $\boldsymbol{\beta}$, so the time complexity per item is $O(k^3 + n^2)$. This computation needs to be done for $k = 1, ..., (M-1)$ in a list and for m censored lists, which makes the overall time complexity $O(mM(M^3 + n^2))$. In the worst case, M is as large as n, which makes it $O(mn^4)$. Even the state-of-the-art batch optimization method such as LBFGS takes a very long time to find the solution for a moderate problem size such as $n \approx 500$. For a faster computation of the RegMLE (6), we turn to averaged stochastic gradient descent (ASGD) [20, 18]. ASGD processes the lists sequentially by updating the parameters after every list. The per-round objective function for $\boldsymbol{\beta}$ on the i-th list is

$$f(\mathbf{a}^{(i)}; \boldsymbol{\beta}) \equiv -\sum_{k=1}^{M_i-1} \log \mathbb{P}(a_{k+1}^{(i)} \mid a_{1:k}^{(i)}; \boldsymbol{\beta}) + \frac{C_{\boldsymbol{\beta}}}{2m} \sum_{i\neq j} \beta_{ij}^2.$$



(a) (b) (c)

Figure 2: Toy experiment results where the error is measured with the Frobenius norm.

We randomly initialize $\boldsymbol{\beta}_0$. At round t, we update the solution $\boldsymbol{\beta}_t$ with $\boldsymbol{\beta}_t \leftarrow \boldsymbol{\beta}_{t-1} - \eta_t \nabla f(a^{(i)}; \boldsymbol{\beta})$ and the average estimate $\overline{\boldsymbol{\beta}}_t$ with $\overline{\boldsymbol{\beta}}_t \leftarrow \frac{t-1}{t}\overline{\boldsymbol{\beta}}_{t-1} + \frac{1}{t}\boldsymbol{\beta}_t$. Let $\eta_t = \gamma_0(1+\gamma_0 a t)^{-c}$. We use $a = C_{\boldsymbol{\beta}}/m$ and $c = 3/4$ following [3] and pick γ_0 by running the algorithm on a small subsample of the train set. We run ASGD for a fixed number of epochs and take the final $\overline{\boldsymbol{\beta}}_t$ as the solution.

3 Experiments

We compare INVITE against two popular estimators of $\mathbf{P}$: naive random walk (RW) and First-Edge (FE). RW is the regularized MLE of the naive random walk, pretending the censored lists are the underlying uncensored walk trajectory: $\widehat{P}_{rc}^{(RW)} \propto \left(\sum_{i=1}^{m}\sum_{j=1}^{M_i-1} \mathbb{1}_{(a_j^{(i)}=r)\wedge(a_{j+1}^{(i)}=c)}\right) + C_{RW}$. Though simple and popular, RW is a biased estimator due to the model mismatch. FE was proposed in [2] for graph structure recovery in cascade model. FE uses only the first two items in each censored list: $\widehat{P}_{rc}^{(FE)} \propto \left(\sum_{i=1}^{m} \mathbb{1}_{(a_1^{(i)}=r)\wedge(a_2^{(i)}=c)}\right) + C_{FE}$. Because the first transition in a censored list is always the same as the first transition in its underlying trajectory, FE is a consistent estimator of $\mathbf{P}$ (assuming $\boldsymbol{\pi}$ has no zero entries). In fact, FE is equivalent to the RegMLE of the length two prefix likelihood of the INVITE model. However, we expect FE to waste information since it discards the rest of the censored lists. Furthermore, FE cannot estimate the transition probabilities from an item that does not appear as the first item in the lists, which is common in real-world data.

3.1 Toy Experiments

Here we compare the three estimators INVITE, RW, and FE on toy datasets, where the observations are indeed generated by an initial-visit emitting random walk. We construct three undirected, unweighted graphs of $n = 25$ nodes each: (i) **Ring**, a ring graph, (ii) **Star**, $n-1$ nodes each connected to a "hub" node, and (iii) **Grid**, a 2-dimensional $\sqrt{n} \times \sqrt{n}$ lattice.

The initial distribution $\boldsymbol{\pi}^*$ is uniform, and the transition matrix $\mathbf{P}^*$ at each node has an equal transition probability to its neighbors. For each graph, we generate datasets with $m \in \{10, 20, 40, 80, 160, 320, 640\}$ censored lists. Each censored list has length n. We note that, in the star graph a censored list contains many apparent transitions between leaf nodes, although such transitions are not allowed in its underlying uncensored random walk. This will mislead RW. This effect is less severe in the grid graph and the ring graph.

For each estimator, we perform 5-fold cross validation (CV) for finding the best smoothing parameters $C_{\boldsymbol{\beta}}, C_{RW}, C_{FE}$ on the grid $10^{-2}, 10^{-1.5}, \ldots, 10^{1}$, respectively, with which we compute each estimator. Then, we evaluate the three estimators using the Frobenius norm between $\widehat{\mathbf{P}}$ and the true transition matrix $\mathbf{P}^*$: $\text{error}(\widehat{\mathbf{P}}) = \sqrt{\sum_{i,j}(\widehat{P}_{ij} - P^*_{ij})^2}$. Note the error must approach 0 as m increases for consistent estimators. We repeat the same experiment 20 times where each time we draw a new set of censored lists.

Figure 2 shows how $\text{error}(\widehat{\mathbf{P}})$ changes as the number of censored lists m increases. The error bars are 95% confidence bounds. We make three observations: (1) *INVITE tends towards 0 error*. This is expected given the consistency of INVITE in Theorem 6. (2) *RW is biased*. In all three plots, RW tends towards some positive number, unlike INVITE and FE. This is because RW has the wrong model on the censored lists. (3) *INVITE outperforms FE*. On the ring and grid graphs INVITE dominates FE for every training set size. On the star graph FE is better than INVITE with a small m, but INVITE eventually achieves lower error. This reflects the fact that, although FE is unbiased, it discards most of the censored lists and therefore has higher variance compared to INVITE.

		Animal	Food
n		274	452
m		4710	4622
Length	Min.	2	1
	Max.	36	47
	Mean	18.72	20.73
	Median	19	21

Table 1: Statistics of the verbal fluency data.

	Model	Test set mean neg. loglik.
Animal	INVITE	**60.18 ($\pm$1.75)**
	RW	69.16 ($\pm$2.00)
	FE	72.12 ($\pm$2.17)
Food	INVITE	**83.62 ($\pm$2.32)**
	RW	94.54 ($\pm$2.75)
	FE	100.27 ($\pm$2.96)

Table 2: Verbal fluency test set log likelihood.

3.2 Verbal Fluency

We now turn to the real-world fluency data where we compare INVITE with the baseline models. Since we do not have the ground truth parameter $\boldsymbol{\pi}$ and $\mathbf{P}$, we compare test set log likelihood of various models. Confirming the empirical performance of INVITE sheds light on using it for practical applications such as the dignosis and classification of the brain-damaged patient.

Data The data used to assess human memory search consists of two verbal fluency datasets from the Wisconsin Longitudinal Survey (WLS). The WLS is a longitudinal assessment of many sociodemographic and health factors that has been administered to a large cohort of Wisconsin residents every five years since the 1950s. Verbal fluency for two semantic categories, animals and foods, was administered in the last two testing rounds (2005 and 2010), yielding a total of 4714 lists for animals and 4624 lists for foods collected from a total of 5674 participants ranging in age from their early-60's to mid-70's. The raw lists included in the WLS were preprocessed by expanding abbreviations ("lab" $\rightarrow$ "labrador"), removing inflections ("cats" $\rightarrow$ "cat"), correcting spelling errors, and removing response errors like unintelligible items. Though instructed to not repeat, some human participants did occasionally produce repeated words. We removed the repetitions from the data, which consist of 4% of the word token responses. Finally, the data exhibits a Zipfian behavior with many idiosyncratic, low count words. We removed words appearing in less than 10 lists. In total, the process resulted in removing 5% of the total number of word token responses. The statistics of the data after preprocessing is summarized in Table 1.

Procedure We randomly subsample 10% of the lists as the test set, and use the rest as the training set. We perform 5-fold CV on the training set for each estimator to find the best smoothing parameter $C_{\boldsymbol{\beta}}, C_{RW}, C_{FE} \in \{10^1, 10^{\cdot 5}, 10^0, 10^{-.5}, 10^{-1}, 10^{-1.5}, 10^{-2}\}$ respectively, where the validation measure is the prefix log likelihood for INVITE and the standard random walk likelihood for RW. For the validation measure of FE we use the INVITE prefix log likelihood since FE is equivalent to the length two prefix likelihood of INVITE. Then, we train the final estimator on the whole training set using the fitted regularization parameter.

Result The experiment result is summarized in Table 2. For each estimator, we measure the average per-list negative prefix log likelihood on the test set for INVITE and FE, and the standard random walk per-list negative log likelihood for RW. The number in the parenthesis is the 95% confidence interval. Boldfaced numbers mean that the corresponding estimator is the best and the difference from the others is statistically significant under a two-tailed paired t-test at 95% significance level. In both animal and food verbal fluency tasks, the result indicates that human-generated fluency lists are better explained by INVITE than by either RW or FE. Furthermore, RW outperforms FE. We believe that FE performs poorly despite being consistent because the number of lists is too small (compared to the number of states) for FE to reach a good estimate.

4 Related Work

Though behavior in semantic fluency tasks has been studied for many years, few computationally explicit models of the task have been advanced. Influential models in the psychological literature, such as the widely-known "clustering and switching" model of Troyer et al. [21], have been articulated only verbally. Efforts to estimate the structure of semantic memory from fluency lists have mainly focused on decomposing the structure apparent in distance matrices that reflect the mean inter-item ordinal distances across many fluency lists [5]—but without an account of the processes that generate list structure it is not clear how the results of such studies are best interpreted. More recently, researchers in cognitive science have begun to focus on explicit model of the processes by which fluency lists are generated. In these works, the structure of semantic memory is first modelled either as a graph or as a continuous multidimensional space estimated from word co-occurrence statistics in large corpora of natural language. Researchers then assess whether structure in fluency data can be understood as resulting from a particular search process operating over the specified semantic

structure. Models explored in this vein include simple random walk over a semantic network, with repeated nodes omitted from the sequence produced [12], the PageRank algorithm employed for network search by Google [13], and foraging algorithms designed to explain the behavior of animals searching for food [15]. Each example reports aspects of human behavior that are well-explained by the respective search process, given accompanying assumptions about the nature of the underlying semantic structure. However, these works do not learn their model directly from the fluency lists, which is the key difference from our study.

Broder's algorithm **Generate** [4] for generating random spanning tree is similar to INVITE's generative process. Given an undirected graph, the algorithm runs a random walk and outputs each transition to an unvisited node. Upon transiting to an already visited node, however, it does not output the transition. The random walk stops after visiting every node in the graph. In the end, we observe an ordered list of transitions. For example, in Figure 1(a) if the random walk trajectory is (2,1,2,1,3,1,4), then the output is (2→1, 1→3, 1→4). Note that if we take the starting node of the first transition and the arriving nodes of each transition, then the output list reduces to a censored list generated from INVITE with the same underlying random walk. Despite the similarity, to the best of our knowledge, the censored list derived from the output of the algorithm Generate has not been studied, and there has been no parameter estimation task discussed in prior works.

Self-avoiding random walk, or non-self-intersecting random walk, performs random walk while avoiding already visited node [9]. For example, in Figure 1(a), if a self-avoiding random walk starts from state 2 then visits 1, then it can only visit states 3 or 4 since 2 is already visited. In not visiting the same node twice, self-avoiding walk is similar to INVITE. However, a key difference is that self-avoiding walk cannot produce a transition $i \to j$ if $P_{ij} = 0$. In contrast, INVITE can appear to have such "transitions" in the censored list. Such behavior is a core property that allows INVITE to *switch* clusters in modeling human memory search.

INVITE resembles cascade models in many aspects [16, 11]. In a cascade model, the information or disease spreads out from a seed node to the whole graph by infections that occur from an infected node to its neighbors. [11] formulates a graph learning problem where an observation is a list, or so-called trace, that contains infected nodes along with their infection time. Although not discussed in the present paper, it is trivial for INVITE to produce time stamps for each item in its censored list, too. However, there is a fundamental difference in how the infection occurs. A cascade model typically allows multiple infected nodes to infect their neighbors in parallel, so that infection can happen simultaneously in many parts of the graph. On the other hand, INVITE contains a *single surfer* that is responsible for all the infection via a random walk. Therefore, infection in INVITE is necessarily sequential. This results in INVITE exhibiting clustering behaviors in the censored lists, which is well-known in human memory search tasks [21].

5 Discussion

There are numerous directions to extend INVITE. First, more theoretical investigation is needed. For example, although we know the MLE of INVITE is consistent, the convergence rate is unknown. Second, one can improve the INVITE estimate when data is sparse by assuming certain cluster structures in the transition matrix $\mathbf{P}$, thereby reducing the degrees of freedom. For instance, it is known that verbal fluency tends to exhibit "runs" of semantically related words. One can assume a stochastic block model $\mathbf{P}$ with parameter sharing at the block level, where the blocks represent semantic clusters of words. One then estimates the block structure and the shared parameters at the same time. Third, INVITE can be extended to allow repetitions in a list. The basic idea is as follows. In the k-th segment we previously used an absorbing random walk to compute $\mathbb{P}(a_{k+1} \mid a_{1:k})$, where $a_{1:k}$ were the nonabsorbing states. For each nonabsorbing state a_i, add a "dongle twin" absorbing state a_i' attached only to a_i. Allow a small transition probability from a_i to a_i'. If the walk is absorbed by a_i', we output a_i in the censored list, which becomes a repeated item in the censored list. Note that the likelihood computation in this augmented model is still polynomial. Such a model with "reluctant repetitions" will be an interesting interpolation between "no repetitions" and "repetitions as in a standard random walk."

Acknowledgments

The authors are thankful to the anonymous reviewers for their comments. This work is supported in part by NSF grants IIS-0953219 and DGE-1545481, NIH Big Data to Knowledge 1U54AI117924-01, NSF Grant DMS-1265202, and NIH Grant 1U54AI117924-01.

References

[1] J. T. Abbott, J. L. Austerweil, and T. L. Griffiths, "Human memory search as a random walk in a semantic network," in *NIPS*, 2012, pp. 3050–3058.

[2] B. D. Abrahao, F. Chierichetti, R. Kleinberg, and A. Panconesi, "Trace complexity of network inference." *CoRR*, vol. abs/1308.2954, 2013.

[3] L. Bottou, "Stochastic gradient tricks," in *Neural Networks, Tricks of the Trade, Reloaded*, ser. Lecture Notes in Computer Science (LNCS 7700), G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Springer, 2012, pp. 430–445.

[4] A. Z. Broder, "Generating random spanning trees," in *FOCS*. IEEE Computer Society, 1989, pp. 442–447.

[5] A. S. Chan, N. Butters, J. S. Paulsen, D. P. Salmon, M. R. Swenson, and L. T. Maloney, "An assessment of the semantic network in patients with alzheimer's disease." *Journal of Cognitive Neuroscience*, vol. 5, no. 2, pp. 254–261, 1993.

[6] J. R. Cockrell and M. F. Folstein, "Mini-mental state examination." *Principles and practice of geriatric psychiatry*, pp. 140–141, 2002.

[7] P. G. Doyle and J. L. Snell, *Random Walks and Electric Networks*. Washington, DC: Mathematical Association of America, 1984.

[8] R. Durrett, *Essentials of stochastic processes*, 2nd ed., ser. Springer texts in statistics. New York: Springer, 2012.

[9] P. Flory, *Principles of polymer chemistry*. Cornell University Press, 1953.

[10] A. M. Glenberg and S. Mehta, "Optimal foraging in semantic memory," *Italian Journal of Linguistics*, 2009.

[11] M. Gomez Rodriguez, J. Leskovec, and A. Krause, "Inferring networks of diffusion and influence," Max-Planck-Gesellschaft. New York, NY, USA: ACM Press, July 2010, pp. 1019–1028.

[12] J. Goi, G. Arrondo, J. Sepulcre, I. Martincorena, N. V. de Mendizbal, B. Corominas-Murtra, B. Bejarano, S. Ardanza-Trevijano, H. Peraita, D. P. Wall, and P. Villoslada, "The semantic organization of the animal category: evidence from semantic verbal fluency and network theory." *Cognitive Processing*, vol. 12, no. 2, pp. 183–196, 2011.

[13] T. L. Griffiths, M. Steyvers, and A. Firl, "Google and the mind: Predicting fluency with pagerank," *Psychological Science*, vol. 18, no. 12, pp. 1069–1076, 2007.

[14] N. M. Henley, "A psychological study of the semantics of animal terms." *Journal of Verbal Learning and Verbal Behavior*, vol. 8, no. 2, pp. 176–184, Apr. 1969.

[15] T. T. Hills, P. M. Todd, and M. N. Jones, "Optimal foraging in semantic memory," *Psychological Review*, pp. 431–440, 2012.

[16] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '03. New York, NY, USA: ACM, 2003, pp. 137–146.

[17] F. Pasquier, F. Lebert, L. Grymonprez, and H. Petit, "Verbal fluency in dementia of frontal lobe type and dementia of Alzheimer type." *Journal of Neurology*, vol. 58, no. 1, pp. 81–84, 1995.

[18] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM J. Control Optim.*, vol. 30, no. 4, pp. 838–855, July 1992.

[19] T. T. Rogers, A. Ivanoiu, K. Patterson, and J. R. Hodges, "Semantic memory in Alzheimer's disease and the frontotemporal dementias: a longitudinal study of 236 patients." *Neuropsychology*, vol. 20, no. 3, pp. 319–335, 2006.

[20] D. Ruppert, "Efficient estimations from a slowly convergent robbins-monro process," Cornell University Operations Research and Industrial Engineering, Tech. Rep., 1988.

[21] A. Troyer, M. Moscovitch, G. Winocur, M. Alexander, and D. Stuss, "Clustering and switching on verbal fluency: The effects of focal fronal- and temporal-lobe lesions," *Neuropsychologia*, vol. 36, no. 6, 1998.