Avicaching: Reducing Bias in Citizen Science with a Scalable Principle-Agent Game

Anmol Kabra* [1] Yexiang Xue [2] Carla P. Gomes [3]

[1]ak2426@cornell.edu [2]yexiang@purdue.edu [3]gomes@cs.cornell.edu

Introduction

Citizen science programs outsource large-scale data collection, thus enabling researchers to expedite scientific discovery and environmental conservation.

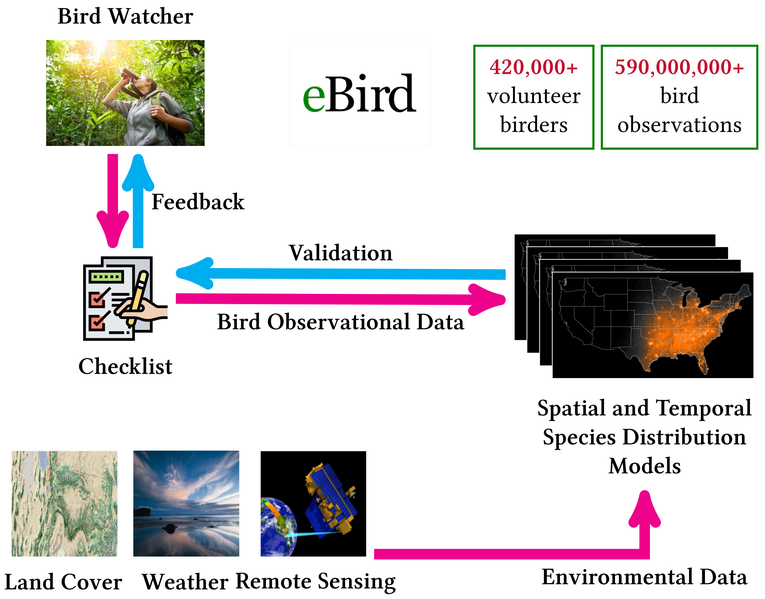

Figure 1. A flow diagram for *eBird*, a citizen science program by the Cornell Lab of Ornithology.

Bias in Citizen Science

Although citizens assist in large-scale data collection, they collect data according to their own motivations, rather than systematically providing scientific observations. These biases introduce sampling bias, and manifest in the form of undesirable spatio-temporal clustering of submitted observations.

This sampling bias can be tackled with Principal-agent games. In these games, researchers (principal) deploy incentives, which could motivate citizens (agents) to complete the most crucial tasks, by factoring in the citizens' reasoning process. For example, *eBird* recently introduced the *Avicaching* game to reward citizen scientists for visiting previously under-sampled locations.

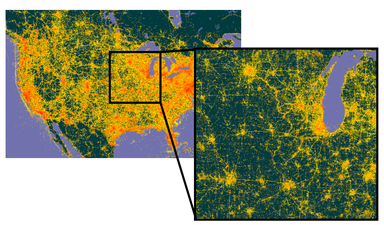

Figure 2. Spatial clustering in *eBird* submissions before 2014 in the US.

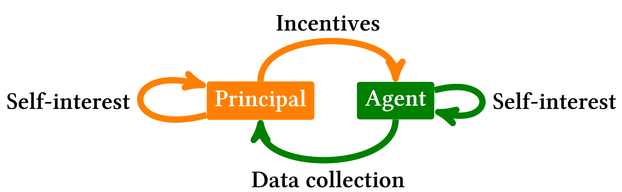

Figure 3. A Principal-agent game. The principal incentivizes agents to complete crucial tasks.

*Corresponding author; presenter.
Poster also available at anmolkabra.com/docs/avicaching-scalable-sp19-poster.pdf

Reducing Spatial Clustering

The principal must factor in agents' preferences for tasks when allocating rewards.

In this two-stage problem, the problem of learning the agents' behavior, with respect to their preferences v and principal's rewards r, is embedded in the problem of distributing rewards to maximize the principal's utility U_p. The behavior and rewards are subject to constraints $B_a(v)$ and $B_p(r)$ respectively.

$$\begin{aligned} \textbf{(Principal)}\quad & r^* = \text{argmax}_r\ U_p(v^*, r) \\ & \text{subject to}\quad B_p(r) \\ \textbf{(Agents)}\quad & v^* = \text{argmax}_v\ U_a(v, r) \\ & \text{subject to}\quad B_a(v) \end{aligned}$$

Problem Formulation

Identification Problem

Consider *eBird* as an example. We extract the number of agents' visits to a location u from the dataset before and after a reward $r_{t,u}$ was applied during time period t. The visit densities are denoted as $\boldsymbol{x_t}, \boldsymbol{y_t} \in \mathbb{R}^n$ respectively.

To learn how agents' visits changed during t, we can learn a mapping $\boldsymbol{P} : \boldsymbol{x_t} \mapsto \boldsymbol{y_t}$ such that $\boldsymbol{P}\boldsymbol{x_t} \approx \boldsymbol{y_t}$. The entry $p_{u,v}$ in $\boldsymbol{P}$ is the probability of agents' efforts shifting from location v to u; the entry is thus a function of environmental features $\boldsymbol{f_u}$, reward $r_{t,u}$, and the distance of v and u, all transformed by g with parameters $\boldsymbol{w}$.

$$\begin{aligned} p_{u,v} &= \frac{\exp(g(\boldsymbol{f_u}, d_{u,v}, r_{t,u}; \boldsymbol{w}))}{\sum_{u'} \exp(g(\boldsymbol{f_{u'}}, d_{u',v}, r_{t,u'}; \boldsymbol{w}))} \\ &= \text{softmax}_v(g)[u] \end{aligned} \qquad \boldsymbol{w}^* = \text{argmin}_{\boldsymbol{w}} \sum_t \left\| \boldsymbol{y_t} - \boldsymbol{P}_{f,D,r_t;\boldsymbol{w}} \boldsymbol{x_t} \right\|_2^2$$

Pricing Problem

To reduce spatial clustering, the pricing problem finds rewards that minimize variance $\|\boldsymbol{y} - \overline{\boldsymbol{y}}\|_1$ in predicted visit densities $\boldsymbol{y} = \boldsymbol{P}\boldsymbol{x}$, where $\boldsymbol{P}$ is evaluated at $\boldsymbol{w}^*$ and $\boldsymbol{x} = \frac{\boldsymbol{x'}}{\|\boldsymbol{x'}\|_1}, \boldsymbol{x'} = \sum_t \boldsymbol{x_t}$.

Additionally, the rewards must be non-negative, sum to at most budget $\mathcal{R}$, and can only be placed at *Avicaching* locations given by $\boldsymbol{a}$, where $a_u = 1$ if u is an *Avicaching* location, and 0 otherwise.

$$\begin{aligned} \boldsymbol{r}^* = \text{argmin}_{\boldsymbol{r}}\ & \frac{1}{n} \|\boldsymbol{y} - \overline{\boldsymbol{y}}\|_1 \\ \text{subject to}\ & \boldsymbol{y} = \boldsymbol{P}_{f,D,\boldsymbol{w}^*;\boldsymbol{r}} \boldsymbol{x} \\ & \forall u, (1 - a_u) r_u = 0 \\ & \forall u, r_u \geq 0 \\ & \|\boldsymbol{r}\|_1 \leq \mathcal{R} \end{aligned}$$

Scaling *Avicaching* using Machine Learning

A previous study folded the identification problem as linear constraints into the pricing problem by defining g (in expression for $p_{u,v}$) as a linear transformation. The pricing problem was then solved using a Mixed-Integer Programming (MIP) solver, which scaled poorly as the number of locations in the game increased.

We generalize the formulation by using a deep neural network for g, and scale both problems efficiently by harnessing Graphical Processing Units (GPUs).

Neural Network Architecture

We build the input as follows: $\boldsymbol{F}[v]$ is a $n \times m$ matrix such that $\boldsymbol{F}[v][u] = [\boldsymbol{f_u}, d_{u,v}, r_{t,u}]$, a vector of m features that could influence agents to shift from v to u. This input $\boldsymbol{F}[v]$ is then fed forward into the network, which repeatedly takes linear combinations and applies non-linear functions to obtain $\boldsymbol{P}[v]$, as shown below.

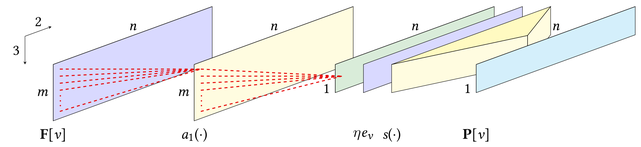

Figure 4. A 3-layer neural network for the identification problem. s is the softmax function, and a_i are non-linear "activation" functions. ηe_v is a regularizer ($e_v[u] = 1$ if $v = u$ and 0 otherwise) that controls how much $\boldsymbol{P}[v]$ looks like e_v.

The neural network model is essentially a sequence of batch-matrix operations that can be parallelized on GPUs. We can learn $\boldsymbol{w}^*$ and $\boldsymbol{r}^*$ using gradient descent updates on the same backpropagation framework. As shown below, the learned $\boldsymbol{w}^*$ from the identification problem become inputs to the pricing problem, which tunes the rewards $\boldsymbol{r}$ to minimize the pricing problem objective.

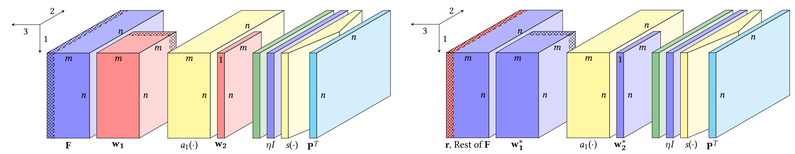

Figure 5. Tensor representations of the identification (left) and the pricing (right) models. Dark blue tensors are static and red are optimizable. On the top-left are directions of dimensions.

Results

We compare our GPU-run models' performance and speed with those from previous studies (n = 116 locations, m = 35 total features, T = 182 time periods).

	Loss	Runtime (s)
Random	1.014	—
Random Forest	0.491	26.4
BFGS	0.374	507.3
2-layer	0.366	48.0
6-layer	**0.358**	647.8

Table 1. Identification problem. We split the dataset along t as 75-5-20 train-valid-test sets and report the test loss values for different models (lower the better). Compared to the state-of-the-art BFGS-based model, our networks perform better while taking less time.

	2-layer	6-layer
Learned	1.073	**1.025**
MIP	1.110	—
Random	1.169	1.303

Table 2. Pricing problem. Comparing the normalized objective, $\frac{1}{n}\frac{\|\boldsymbol{y}-\overline{\boldsymbol{y}}\|_1}{\overline{\boldsymbol{y}}}$ (lower the better), on different reward allocation schemes, we see that 2- and 6-layer networks better optimize the objective than the MIP formulation. The MIP formulation takes ≥ 36,000 seconds to run, while the GPU-run 6-layer network completes 10,000 iterations in 44.45 seconds.

In both problems, we perform better than the previous state of the art while taking less time; **our models are more than 800x faster than MIP** in the pricing problem.

References

[1] Anmol Kabra, Yexiang Xue, and Carla P. Gomes. GPU-accelerated Principal-Agent Game for Scalable Citizen Science. Submitted to ACM COMPASS 2019.

Acknowledgements

We are grateful for funding by National Science Foundation (Grant Number 0832782, 1522054, 1059284, 1356308), ARO grant W911-NF-14-1-0498, the Leon Levy Foundation and the Wolf Creek Foundation.