Contents

List of Tables

List of Figures

1 Introduction

Optimizing predictive models datasets obtained from citizen science projects can be computationally expensive due to limited parallelism offered by Central Processing Units (CPUs). Since these crowd-sourced datasets grow with time as the researchers warrant more information, running models can prove increasingly difficult even on the fastest CPUs available in the market. However, Graphical Processing Units (GPUs), which offer multiple cores to parallelize computation, can outperform CPUs in computing such predictive models given the operations are arithmetically simple (though large in number) and boolean logic-free to the maximum extent.

One such predictive model is Avicaching [1] in the eBird platform [cite], which aims to "maximize the utility and accessibility of the vast numbers of bird observations made each year by recreational and professional bird watchers" [cite website]. Since the eBird dataset is heterogeneous geographically, Avicaching tries to homogenize future observations by providing observer-citizens incentives to visit under-sampled locations [1]. Therefore, the Avicaching game can be modeled as two separate problems - identifying parameters of citizens' behavior on a specific set of rewards (incentives), and pooling rewards from a budget to locations such that the citizens' predicted behavior is homogeneous [2].

1.1 Identification Problem

This subroutine will learn parameters of the change in citizens' behavior due to rewards. Specifically, given datasets $\mathbf{y_t}$ and $\mathbf{x_t}$ of citizens' visit densities with and without the rewards (r_t), we want to find weights $\mathbf{w}$ that caused the change from $\mathbf{x_t}$ to $\mathbf{y_t}$, factoring in possible influence due to environmental factors $\mathbf{f}$ and distances between locations $\mathbf{d}$. Therefore, the model is:

$$\underset{\mathbf{w}}{\text{minimize}} \quad Z_I(\mathbf{w}) = \sum_t (u_t(\mathbf{y_t} - \mathbf{P}(\mathbf{f}, \mathbf{r_t}; \mathbf{w})\mathbf{x_t}))^2 \tag{1}$$

where u_t is the total number of submissions at time t for a given citizen and elements $p_{u,v}$ of P are given as:

$$p_{u,v} = \frac{\exp\left(\mathbf{w} \cdot [d_{u,v}, \mathbf{f_{u,v}}, r_u]\right)}{\sum_{u'} \exp\left(\mathbf{w} \cdot [d_{u',v}, \mathbf{f_{u',v}}, r_{u'}\right]} \tag{2}$$

1.2 Pricing Problem

The Pricing subroutine will attempt to allocate rewards to all locations such that the predicted behavior is least heterogeneous, given the learned parameters from Section 1.1. This optimization problem can be written as:

$$\begin{aligned} \underset{\mathbf{r}}{\text{minimize}} \quad & Z_P(\mathbf{r}) = \frac{1}{n}\|\mathbf{y} - \overline{\mathbf{y}}\| \\ \text{subject to} \quad & \mathbf{y} = \mathbf{P}(\mathbf{f}, \mathbf{r}; \mathbf{w})\,\mathbf{x} \\ & \sum_i r_i \leq \mathcal{R} \\ & r_i \geq 0 \end{aligned} \tag{3}$$

where $\mathcal{R}$ is total reward budget and $\mathbf{P}$ is defined as in Equation 2.

1.3 Computation Using GPUs

2 Problem Formulation

Since NVIDIA General Purpose GPUs enable faster computation on matrices, accelerated through CUDA and cuDNN, both the Identification Problem (Section 1.1) and Pricing Problem (Section 1.2) were formulated as 2-layer Neural Networks using the PyTorch platform.

2.1 Design for Identification Problem

The Identification Problem (Section 1.1) can be modeled as a typical 2-layered Neural Network, which learns the weights through multiple iterations of back-propagating the loss value $Z_I(\mathbf{w})$ (equation 1) using gradient descent.

2.1.1 Structure of Input Dataset

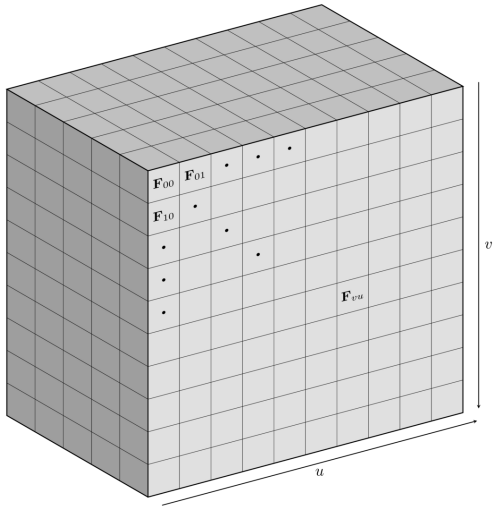

Figure 1: A Tensor representing the Input Dataset

The input dataset comprising of environmental features $\mathbf{f}$ and given rewards $\mathbf{r_t}$ must be combined into a tensor (figure 1) that can be readily operated on by NVIDIA GPUs. Another advantage of building the dataset as discussed comes with the PyTorch library, which provides convenient handling of tensors residing on CPUs as well as GPUs. Algorithm 1 describes the steps to construct this dataset.

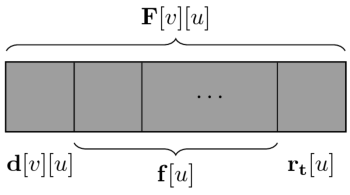

Figure 2: Contents of $\mathbf{F}_{vu}$

Algorithm 1 Constructing the Input Dataset

$\mathbf{d} \leftarrow \text{NORMALIZE}(\mathbf{d})$ ▷ $\mathbf{d}[u][v]$ is the distance between locations u and v
$\mathbf{f} \leftarrow \text{NORMALIZE}(\mathbf{f}, axis = 0)$ ▷ $\mathbf{f}[u]$ is a vector of env. features at location u
$\mathbf{r_t} \leftarrow \text{NORMALIZE}(\mathbf{r_t}, axis = 0)$ ▷ $\mathbf{r_t}[u]$ is the reward at location u
for $v = 1, 2, \ldots J$ **do**
    for $u = 1, 2, \ldots J$ **do**
        $\mathbf{F}[v][u] \leftarrow [\mathbf{d}[v][u], \mathbf{f}[u], \mathbf{r_t}[u]]$ 2
    end for
end for

2.1.2 Algorithm for Identification Problem

For learning the weights $\mathbf{w}$ in equation 1, a Neural Network was built using the PyTorch library, which minimized the loss function using gradient descent routines. Figure 3 illustrates the framework of the network used, calculating the vector $\mathbf{p_v}$. This network was fed in with data from different locations v, resulting into the matrix $\mathbf{P}$, which was then used to calculate the loss.

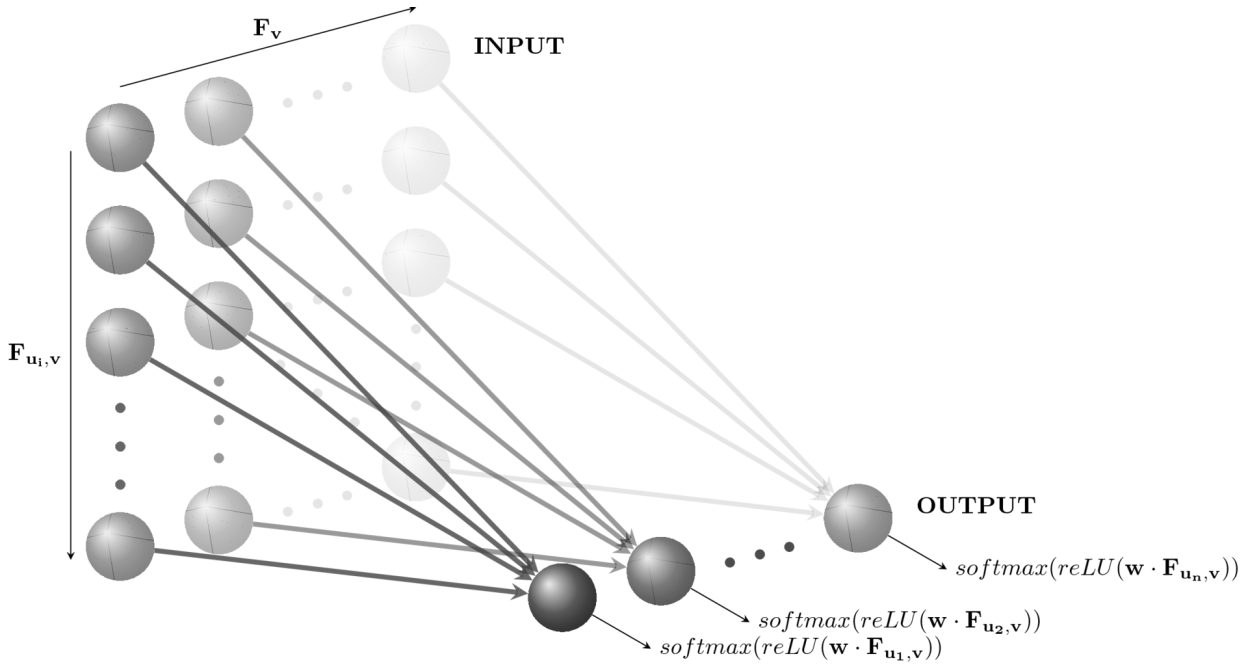

Figure 3: Neural Network designed for Identification Problem

An algorithm for the optimization problem is described in Algorithm 2.

Algorithm 2 Optimizing for the Identification Problem

1: $a \leftarrow b$

3 Experiments

4 Results

5 Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

References

[1] Y. Xue, I. Davies, D. Fink, C. Wood, and C. P. Gomes, "Avicaching: A two stage game for bias reduction in citizen science," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, (Richland, SC), pp. 776–785, International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[2] Y. Xue, I. Davies, D. Fink, C. Wood, and C. P. Gomes, "Behavior identification in two-stage games for incentivizing citizen science exploration," in *Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings*, pp. 701–717, 2016.