Contents

List of Tables

List of Figures

1 Introduction

Optimizing predictive models on datasets that are obtained from citizen-science projects can be computationally expensive as these datasets grow in size with time. Consequently, models based on multiple-layered neural networks, Integer Programming and other optimization routines can prove increasingly difficult as the number of parameters increase, despite using the faster Central Processing Units (CPUs) in the market. Incidentally, it becomes difficult for citizen-science projects to scale if the organizers use CPUs to run optimization models. However, Graphical Processing Units (GPUs), which offer multiple cores to parallelize computation, can outperform CPUs in computing such predictive models if these models heavily rely on large-scale matrix multiplications. By using GPUs over CPUs to accelerate computation on a citizen-science project, the model could achieve better optimization in less

time, enabling the project to scale.

Part of the eBird project, which aims to "maximize the utility and accessibility of the vast numbers of bird observations made each year by recreational and professional bird watchers", Avicaching is a incentive-driven game trying to homogenize the spatial distribution of citizens' (agents') observations [cite website]. Since the dataset of agents' observations in eBird is geographically heterogeneous (concentrated in some places like cities and sparse in others), Avicaching homogenizes the observation set by rewarding agents who visit under-sampled locations [1]. To accomplish this task of specifying rewards at different locations based on the historical records of observations, Avicaching would learn the change in agents' behavior when a certain sample of rewards were applied to the set of locations, and then distribute a newer set of rewards across the locations based on those learned parameters [2]. This requirement naturally translates into a predictive optimization problem, which was implemented using multiple-layered neural networks and linear programming.

1.1 Identification Problem

As discussed in Section 1, the model would need to learn parameters that caused the change in agents' behavior when a certain set of rewards was applied to locations in the experiment region. Specifically, given datasets $\mathbf{y_t}$ and $\mathbf{x_t}$ of agents' visit densities with and without the rewards $\mathbf{r_t}$, we want to find weights $\mathbf{w_1}$ and $\mathbf{w_2}$ that caused the change from $\mathbf{x_t}$ to $\mathbf{y_t}$, factoring in possible influence from environmental factors $\mathbf{f}$ and distances between locations $\mathbf{D}$. Although the original model proposed to learn a single set of weights $\mathbf{w}$ [2], this proposed model considers two sets of weights $\mathbf{w_1}$ and $\mathbf{w_2}$ as it may theoretically result into higher accuracy and lower loss. Mathematically, our model can be formulated as:

$$\underset{\mathbf{w}}{\text{minimize}} \quad Z_I(\mathbf{w_1}, \mathbf{w_2}) = \sum_t (\omega(\mathbf{y_t} - \mathbf{P}(\mathbf{f}, \mathbf{r_t}; \mathbf{w_1}, \mathbf{w_2})\mathbf{x_t}))^2 \tag{1}$$

where ω is a set of weights at time t capturing penalties relative to the importance of homogenizing at different locations and elements $p_{u,v}$ of $\mathbf{P}$ are given as:

$$p_{u,v} = \frac{\exp(\mathbf{w_2} \cdot \text{reLU}(\mathbf{w_1} \cdot [d_{u,v}, \mathbf{f_u}, r_u]))}{\sum_{u'} \exp(\mathbf{w_2} \cdot \text{reLU}(\mathbf{w_1} \cdot [d_{u',v}, \mathbf{f_{u'}}, r_{u'}]))} = \frac{\exp(\Gamma_{u,v})}{\sum_{u'} \exp(\Gamma_{u',v})} = \text{softmax}(\Gamma_{u,v}) \tag{2}$$

In the expression for $p_{u,v}$ (Equation 2), softmax$(\cdot)$ is the function: $\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_i \exp(z_i)}$ and the reLU$(\cdot)$ function is a "rectified Linear Unit" defined as: $\text{reLU}(z) = \max(0, z)$.

1.2 Pricing Problem

After learning the set of weights $\mathbf{w_1}$ and $\mathbf{w_2}$ highlighting the change in agents' behavior to collect observations, the Pricing Problem would redistribute rewards to the all locations such that the predicted behavior of agents influenced by the new set of rewards is homogeneous. Thus, given a budget of rewards $\mathcal{R}$, this optimization problem can be expressed as:

$$\begin{aligned} \underset{\mathbf{r}}{\text{minimize}} \quad & Z_P(\mathbf{r}) = \frac{1}{n}\|\mathbf{y} - \overline{\mathbf{y}}\| \\ \text{subject to} \quad & \mathbf{y} = \mathbf{P}(\mathbf{f}, \mathbf{r}; \mathbf{w_1}, \mathbf{w_2})\,\mathbf{x} \\ & \sum_i r_i \leq \mathcal{R} \\ & r_i \geq 0 \end{aligned} \tag{3}$$

where elements of $\mathbf{P}$ are defined as in Equation 2.

1.3 Computation Using GPUs

2 Problem Formulation

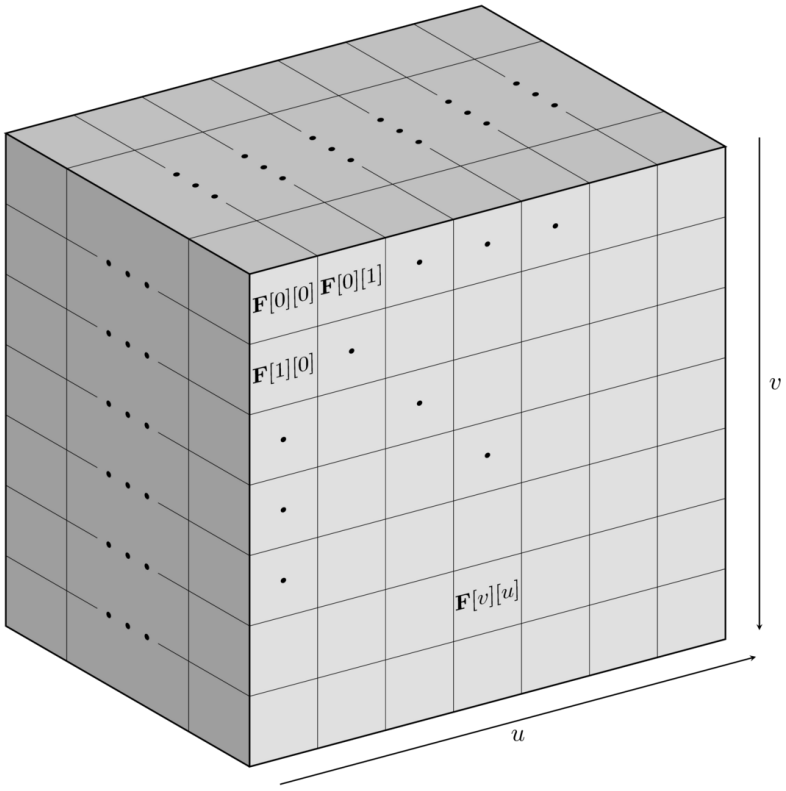
Since NVIDIA General Purpose GPUs enable faster computation on matrices, accelerated through CUDA and cuDNN, both the Identification (Section 1.1) and the Pricing Problem (Section 1.2) were formulated as 3-layered neural networks using the PyTorch library.

2.1 Design for Identification Problem

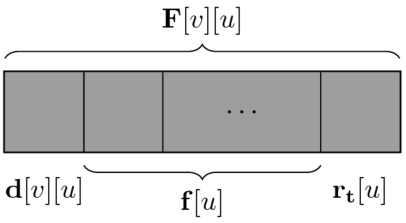
To optimize the loss value $Z_I(\mathbf{w_1}, \mathbf{w_2})$ (Equation 1), the neural network learns the set of weights through multiple iterations of backpropagating the loss using gradient descent. Furthermore, the program would process the dataset before feeding to the network to avoid unnecessary sub-iterations and promote batch operations on matrices.

2.1.1 Structure of Input Dataset

The input dataset comprising of environmental features $\mathbf{f}$ and given rewards $\mathbf{r_t}$ must be combined into a tensor (Figure 1a) 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.



(a) A Tensor representing the Input Dataset



(b) Contents of $\mathbf{F}_{vu}$

Figure 1: Visual representation of the Input Dataset

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**
        $\mathbf{F}[v][u] \leftarrow [\mathbf{d}[v][u], \mathbf{f}[u], \mathbf{r_t}[u]]$ ▷ Looks like Figure 1b
    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 2 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.

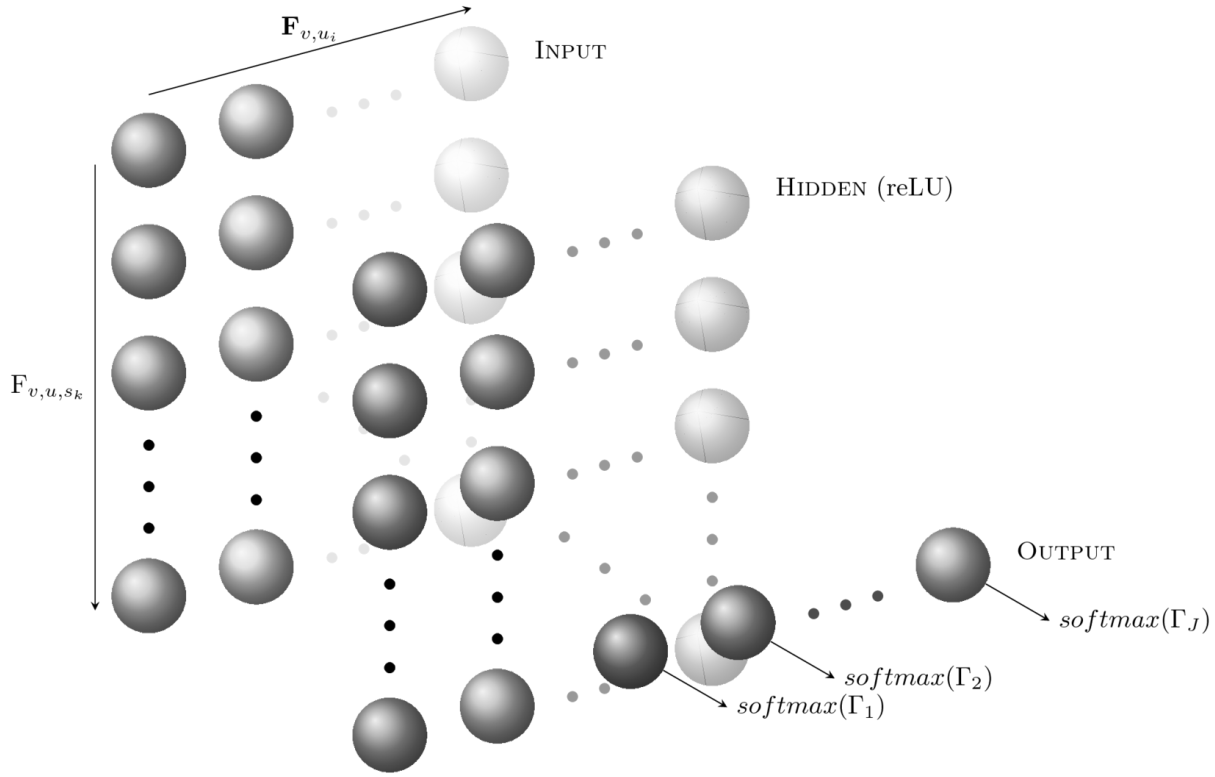


Figure 2: Neural Network designed for Identification Problem

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.