Introduction

Authors have suggested that human computation mediated by microtask platforms be viewed as a first-class computing architecture. They call the human in human computation, an HPU, and call on the field to characterize the computational properties of HPUs in the same way that we would the CPU or GPU.

Psychological experiments tell us that people are susceptible to priming effects. When exposed to some prior stimulus, peoples response characteristics are altered. Psychologists have used this to probe the underlying mechanisms giving rise to responses, by seeign how prior stimuli can purturb them.

The study of human computation profits from the results of these studies, but if we are to satisfy the call of *authors, we need to translate this into a form that can be used to discuss the computational properties of HPUs. We know that, unlike their silicone counterparts, HPUs outputs are stochastic functions of their inputs. But priming constitutes not stochasticity, but hysteresis: meaning that an HPUs outputs are not only a function of their inputs, but also of the *history* of their inputs.

In the context of human computation, the task-designer (requester) will generally make all reasonable attempts to normalize the frame in which HPUs engage with a task. Administering tasks via a microtask platform arguably removes some of the potential variability in the task frame, and providing clear instructions provides a uniform priming treatment that might help to "wash out" the effects of different experiences that HPUs have had leading up to the task engagement.

Here we investigate the potential for tasks themselves to act as primes, influencing the output of the tasks which follow. This is a much subtler source of priming, and one which cannot be so simply eliminated.

We provide two contributions in the present work. The first is a demonstration of how to measure priming effects as a computational phenomenon, rather than as a psychological phenomenon. We apply information theoretic tools and machine learning concepts to characterize how priming can introduce bias into algorithms run on HPUs, and how to measure these effects in general.

Our second contribution is to measure HPU hysteresis during a canonical HPU task: qualitative image labeling.

The remainder of the article is structured as follows. Section 2 reviews information theoretic tools which are useful for measuring HPU hysteresis, and discusses interpretations of these measures. Section 3 describes the setup for an experiment we conducted which measures inter-task hysteresis during a canonical image-labelling task. In section 3, we present the results of the experiment. In section 4 we discuss the ramifications of these results, and of HPU hysteresis, to human computation in general, and make some recommendations, before providing some concluding remarks.

2. Characterizing HPU Hysteresis

First, we shall introduce notation to describe a stochastic input-output device, and then adapt this to describe HPU behavior. We will denote a stochastic input-output device as a function H, which acts on some non-random input x, to yield stochastic $P = H(x)$. The output H is a random variable that can assume values from its support $\mathcal{Y}$, with probability given by a density function $h(y)$.

HPUs are not identical. In general, HPUs are selected from a population with some distribution over prior knowledge, aptitudes, and temperaments, which we will simply call *propensities.* Other work deals with the screening of HPUs, to obtain a finer distribution of desired propensities. The remaining effects of uncertain HPU propensity (after screening if any) influence the particular shape of $p(y)$. We illustrate in the supplementary material that it is analytically valid to account for residual HPU propensity simply as part of the uncertainty of $p(y)$.

Now, to account for hysteresis, we introduce an indicator variable z which indicates the prime that an HPU received. To indicate that an HPU received prime z, we write $H_z(x)$. Thus, in this notation, the assertion that HPU output is subject to hysteresis can be stated formally:

$$H_0(x) \not\sim H_1(x) \tag{1}$$

The finding that $H_0(x) \not\sim H_1(x)$ tells us that $z = 0$ and $z = 1$ are rightly regarded as different primes, but it does not tell us *how different* they are. It would be of more practical interest to understand the impact to the output of some HPU algorithm $\mathcal{A}$, brought about by running $\mathcal{A}$ on H_0 instead of on H_1. We shall characterize this effect, but first, we define a simple HPU algorithm.

Definition A *simple HPU algorithm*, denoted $\mathcal{A}(H)$, is an ordinary algorithm that incorporates the result of a single human computation H at some point in its execution, and yields as output $\mathcal{A}(H) \in \{0, 1\}$. Without loss of generality, we regard H as an input to the algorithm.

There is no loss in generality because any algorithm that accesses H in the midst of execution can be converted into an algorithm that accepts H as an input and stores it until the point it is needed. This includes an algorithm whose sole instruction is H itself. We can now define the bias that results between simple HPU algorithms run using HPUs that have been primed one way H_0 vs another H_1.

Definition The *bias* induced in $\mathcal{A}$ due to the (possibly) differently primed HPUs H_0 and H_1 is equal to the unsigned expected difference between the outputs of $\mathcal{A}(H_0)$ and $\mathcal{A}(H_1)$, and is denoted by $\theta_{\mathcal{A}}(H_0, H_1)$:

$$\theta_{\mathcal{A}}(H_0, H_1) = \Big|\mathrm{E}\big\{\mathcal{A}(H_0) - \mathcal{A}(H_1)\big\}\Big| \tag{2}$$

The bias introduced into $\mathcal{A}$ depends on the fact that $h_0(y)$ and $h_1(y)$ differ, and also on how $\mathcal{A}$ makes use of H. The lower limit of θ is always 0, because a trivial algorithm which yields 0 no matter the value of H_z necesarily has $\theta = 0$. We can, however, provide an upper bound to θ regardless of the nature of $\mathcal{A}$. To do so, we require an auxiliary algorithm which we call a *distinguisher*:

Definition A *distinguisher* for the random variables H_0 and H_1 accepts $H \in \mathcal{Y}$, where $\mathcal{Y}$ is the union of the supports of H_0 and H_1, and yields as output a *guess* $D(H) \in \{0, 1\}$.

One can think of a distinguisher as trying to determine whether its input H came from H_0 or H_1. However, nothing in the definition requires this. Note that when H is the result of a human computation, $\mathcal{D}$ is a simple HPU algorithm. We can characterize the performance of a distinguisher using a *validation test*, which we define as follows:

Definition A *validation test* is an experiment involving two random variables H_0 and H_1, and a distinghuisher $\mathcal{D}$. A uniform random bit $Z \sim \mathrm{B}(0.5)$ is sampled, and then, accordingly, H_Z is sampled. The distinguisher is given H_Z and yields $\mathcal{D}(H_Z)$. If $\mathcal{D}(H_Z) = Z$ the *value of the test*, denoted $V_{\mathcal{D}}(H_0, H_1)$, is 1, otherwise it is 0.

The accuracy of $\mathcal{D}$ is then simply $\mathrm{E}\{V_{\mathcal{D}}(H_0, H_1)\}$. Note that a distinguisher which guesses randomly, or always outputs the same guess, will have accuracy $\frac{1}{2}$. Thus, let us define the performance of a distinguisher to be a shifted, scaled version of accuracy:

Definition The performance of a distinguisher $\mathcal{D}$ in the validation test with H_0, and H_1 is denoted $\nu_{\mathcal{D}}(H_0, H_1)$, and is given by:

$$\nu_{\mathcal{D}}(H_0, H_1) = 2\mathrm{E}\{V_{\mathcal{D}}(H_0, H_1)\} - 1 \tag{3}$$

Note that ν ranges between 0 and 1, and measures how much better the distinguisher is than chance, which is evident when the accuracy is expressed in terms of ν, as $\frac{1+\nu}{2}$.

Next, let us define the *cannonical distinguisher of H_0 and H_1*, denoted $\mathcal{D}^*$, to be the distinguisher with the best possible accuracy, and denote its performance as $\nu^*(H_0, H_1)$. Formally,

$$\mathcal{D}^* = \arg\max_{\mathcal{D}} \mathrm{E}\{V_{\mathcal{D}}(H_0, H_1)\} \tag{4}$$

$$\nu^*(H_0, H_1) = 2\mathrm{E}\{V_{\mathcal{D}^*}(H_0, H_1)\} - 1 \tag{5}$$

Now suppose we have a simple HPU algorithm $\mathcal{A}(H)$ which is susceptible to the different priming treatments $z = 0$ and $z = 1$. In other words, $\theta_{\mathcal{A}}(H_0, H_1) > 0$. We can use $\mathcal{A}$ to build a distinguisher $\mathcal{D}$, with performance $\nu_{\mathcal{D}}(H_0, H_1) = \theta_{\mathcal{A}}(H_0, H_1)$, as follows. If $\mathrm{E}\{\mathcal{A}(H_1)\} > \mathrm{E}\{\mathcal{A}(H_0)\}$, we define $\mathcal{D}$ to be $\mathcal{A}$ itself. Otherwise $\mathcal{D}$ takes in H and runs $\mathcal{A}(H)$ as a subroutine, then outputs $1 - \mathcal{A}(H)$. Assuming without loss of generality the case with $\mathcal{D} = \mathcal{A}$,

$$\begin{aligned}
\mathrm{E}\{V_{\mathcal{D}}(H_0, H_1)\} &= \Pr\{\mathcal{D}(H_Z) = 1 | Z = 1\}\Pr\{Z = 1\} + \Pr\{\mathcal{D}(H_Z) = 0 | Z = 0\}\Pr\{Z = 0\} && (6)\\
&= \frac{1}{2}\left(1 + \mathrm{E}\{\mathcal{D}(H_1)\} - \mathrm{E}\{\mathcal{D}(H_0)\}\right) && (7)\\
&= \frac{1}{2}\left(1 + \mathrm{E}\{\mathcal{A}(H_1)\} - \mathrm{E}\{\mathcal{A}(H_0)\}\right) && (8)\\
& && (9)
\end{aligned}$$

And so,

$$\begin{aligned}
\nu_{\mathcal{D}}(H_0, H_1) &= 2\mathrm{E}\{V_{\mathcal{D}}(H_0, H_1)\} - 1 && (10)\\
&= 2 \cdot \frac{1}{2}\left(1 + \mathrm{E}\{\mathcal{A}(H_1)\} - \mathrm{E}\{\mathcal{A}(H_0)\}\right) - 1 && (11)\\
&= \Big|\mathrm{E}\{\mathcal{A}(H_1)\} - \mathrm{E}\{\mathcal{A}(H_0)\}\Big| && (12)\\
&= \theta_{\mathcal{A}}(H_0, H_1) && (13)
\end{aligned}$$

This implies that the worst possible bias for a simple HPU algorithm $\mathcal{A}(H_z)$ due to the alternate primes $z = 0, z = 1$, is bounded by the performance of the best distinguisher $\mathcal{D}$ for H_0 and H_1:

$$\theta_{\mathcal{A}}(H_0, H_1) \le \nu^*(H_0, H_1), \qquad \forall \mathcal{A} \tag{14}$$

This is a meaningful bound, because the performance of the best classifier is inherently limited by how different the distributions of H_0, and H_1 are. Various measures of divergence exist to quantify the difference between two probability distributions. One of the most straightforward is the total variation, TV, which is defined to be the total area between the probaility density (mass) functions:

$$TV(H_0, H_1) \equiv \sup_{Y \subset \mathcal{Y}} \left| \sum_{y \in Y} h_0(y) - h_1(y) \right| = \frac{1}{2} \sum_{y \in \mathcal{Y}} \Big| h_0(y) - h_1(y) \Big|, \tag{15}$$

where $h_z(y) = \Pr\{H_z = y\}$. It turns out that $TV(H_0, H_1) = \nu^*(H_0, H_1)$ which allows us to bound the maximum bias induced by primes $z = 0$ and $z = 1$:

$$\theta_{\mathcal{A}}(H_0, H_1) \le TV(H_0, H_1), \qquad \forall \mathcal{A} \tag{16}$$

We provide a proof in the supplementary material. This reduces the task of measuring the potential for bias to measuring the TV divergence between the outputs of H_0 and H_1.

Another measure of divergence, the Jensen-Shannon divergence (JSD), yields an intuitive information theoretic interpretation. JSD is defined as follows:

$$JSD(H_0 || H_1) = \frac{1}{2} D(H_0 || M) + \frac{1}{2} D(H_1 || M) \tag{17}$$

where M is a random variable from the mixed distribution $m(x) = \frac{1}{2}h_0(y) + \frac{1}{2}h_1(y)$, and where D stands for the Kullback-Leibler divergence:

$$D(H_z||M) = \sum_{y \in \mathcal{Y}} h_z(y) \lg \left(\frac{h_z(y)}{m(y)} \right) \tag{18}$$

As we rewiew in more detail in the supplementary material, JSD provides an upper bound to the amount of information about the priming treatment Z that "leaks out of" $\mathcal{A}(H)$. Formally:

$$I(\mathcal{A}(H_Z); Z) \leq JSD, \tag{19}$$

where $I(\mathcal{A}(H_Z); Z)$ represents the mutual information between $\mathcal{A}(H_Z)$ and Z.

In general, the task of measuring TV using only samples, and asuming no prior knowledge of $h_0(y)$ and $h_1(y)$ is difficult, and closely related to the task of building a cannonical classifier. A result of this is that knowing a very good classifier for H_0 and H_1 allows one to construct a good estimator for TV, and vice-versa.

Thus, measuring TV allows one to immediately bound θ. Without changing it's syntax, but reinterpreting $\mathcal{A}$ as a classifier which attempts to guess the value of b (z or z') on input H_z, we obtain an alternate interpretation of θ as the amount by which the classifier exceeds a the accuracy of a coin-flip when guessing b. Thus, a classifier's accuracy, expressed as $\frac{1+\theta}{2}$, is subject to the bound in Eq 16. TV therefore tells us the performance of the best possible classifier. It is helpful to keep this in mind, because calculating TV is not easy in general, but we can directly apply knowledge how to classify objects from the domain $\mathcal{Y}$. It turns out that calculating TV (i.e. bounding θ) and building a classifier for the classes $\{z, z'\}$ given observations in $\mathcal{Y}$ are one and the same problem.

thet syntax of $\mathcal{A}$ and θ unchanged, but reinterpre TV is equal to the accuracy of the best possible classifier that takes as input H_b and tries to guess whether $b = z$ or z'.