Notes on Inference Devices

Santa Fe Institute

Edward G. Huang

Summer 2018

1 Notation and Definitions

Standard notation is taken from set theory and vector algebra. We clarify some notation specific to computation and inference devices.

1.1 Turing Machines

$\mathbb{B}^*$ The space of all finite bit strings.

Λ Symbol alphabet of a Turing Machine.

σ A symbol on a Turing Machine tape.

Q Set of finite states of a Turing Machine.

Δ Transition function of a Turing Machine.

k Number of tapes of a Turing Machine. The first tape is assumed to be read-only.

η Non-halting state of a Turing Machine.

1.2 Inference Devices

U Set of possible histories of the universe.

u A history of the universe in U.

X Setup function of an ID that maps $U \to X(U)$. A binary question concerning $\Gamma(u)$.

x A binary question and a member of image $X(U)$.

Y Single-valued conclusion function of an ID that maps $U \to \{-1, 1\}$. A binary answer of an ID for $X(u) = x$.

y A single-valued answer, and member of image $Y(U) = \{0, 1\}$.

Γ A function of the actual values of a physical variable over U, equivalent to $\Gamma(u) = S(t_i)(u)$.

γ Possible value of a physical variable, a member of the image $\Gamma(U)$.

δ Probe of any variable V parameterized by $v \in V$ such that :

$$\delta_v(v') = \begin{cases} 1 & \text{if } v = v' \\ -1 & \text{otherwise} \end{cases}$$

$\wp$ Set of probes over $\Gamma(U)$.

$\mathcal{D} = (X, Y)$ An inference device, consisting of functions X and Y.

$\bar{F}$ Inverse. Given a function F over U, $F^{-1} = \bar{F} \equiv \{\{u : F(u) = f\} : f \in F(U)\}$.

$>$ Weak inference: a device $\mathcal{D}$ weakly infers Γ *iff* $\forall \gamma \in \Gamma(U), \exists x \in X(U)$ s.t. $\forall u \in U$, $X(u) = x \implies Y(u) = \delta_\gamma(\Gamma(u))$.

$\gg$ Strong inference: a device (X, Y) strongly infers a functions (S, T) over U *iff* $\forall \delta \in \wp(T)$ and all $s \in S(U)$, $\exists x$ *such that* $X(u) = x \implies S(u) = s, Y(u) = \delta(T(u))$.

2 Turing Machines

2.1 Deterministic Turing Machines

Arora and Barak denote a Turing Machine (TM) as $T = (\Lambda, Q, \Delta)$ containing:

1. An *alphabet* Λ of a finite set of symbols that T's tapes can contain. We assume that Λ contains a special blank symbol B, start symbol S, and the symbols 0 and 1.

2. A finite set Q of possible states that T's register can be in. We assume that Q contains a special start state q_s and a special halt state q_h.

3. A transition function function $\Delta : Q \times \Lambda^k \to Q \times \Lambda^{k-1} \times \{L, S, R\}^k$, where $k \geq 2$, describing the rules T use in performing each step. The set $\{L, S, R\}$ denote the actions *Left, Stay,* and *Right*, respectively.

Suppose T is in state $q \in Q$ and $(\sigma_1, \sigma_2, \ldots, \sigma_k)$ are the symbols on the k tapes. Then $\Delta(q, (\sigma_1, \ldots, \sigma_k)) = (q', (\sigma_2', \ldots, \sigma_k'), z)$ where $z \in \{L, S, R\}^k$ and at the next step the σ symbols in the last $k-1$ tapes will be replaced by the σ' symbols, the machine will be in state q, and the k heads will move *Left, Right* or *Stay*. This is illustrated in Figure 1.

Figure 2.1. The transition function Δ for a k-tape Turing Machine

$(q, (\sigma_1, \ldots, \sigma_k))$				$(q', (\sigma_2', \ldots, \sigma_k'), z)$				
Input symbol	Work/output symbol read	...	Current state	New work/output tape symbol	...	Move work/output tape	...	New state
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
σ_1	σ_i	$\ddots$	q	σ_i'	$\ddots$	z_i	$\ddots$	q'
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$

Remark: Λ can be reduced to $\mathbb{B} = \{0, 1\}$ and k can be reduced to 1 without loss of computational power. Then, any Turing Machine can be expressed as a partial recursive function mapping $\mathbb{B}^* \rightarrow \mathbb{B}^* \cup \eta$, where η is the undefined non-halting output. Since $|\mathbb{B}^* \times \mathbb{B}^* \cup \eta| = |\mathbb{N} \times \mathbb{N}| = |\mathbb{N}|$, the set of all Turing Machines is countably infinite.

2.2 Non-deterministic Turing Machines

Non-deterministic Turing Machines (NDTM) differ from deterministic Turing Machines by having two transition functions Δ_0, Δ_1 and a special state q_{accept}. From Arora and Barak:

> When a NDTM M computes a function, we envision that at each computational step M makes an arbitrary choice as to which of its two transition functions to apply. For every input x, we say that $M(x) = 1$ if there exists some sequence of these choices (which we call nondeterministic choices of M) that would make M reach q_{accept} on input x. Otherwise - if every sequence of choices makes M halt without reaching q_{accept} - then we say that $M(x) = 0$.

3 Inference of Turing Machines

In the next two examples we examine strong inference of simple single-valued functions.

Example 3.1 Let $T(U) = \{0, 1\}$ and $S(U) = \{0, 1, 2\}$. We construct (X, Y) in the table at the left such that it strongly infers (S, T). The right table indicates x for each s, δ such that the definition of strong inference is satisfied:

u	$X(u)$	$Y(u)$	$S(u)$	$T(u)$
1	1	1	0	0
2	2	−1	0	0
3	3	1	1	0
4	4	−1	1	0
5	5	1	2	1
6	6	−1	2	1

(s, δ)	δ_0	δ_1
0	1	2
1	3	4
2	6	5

Example 3.2 Let $T(U) = \{1, 2, 3\}$ and $S(U) = \{1, 2, 3, 4, 5\}$. Again, we construct (X, Y) in the table at the left such that it strongly infers (S, T). The right table indicates x for each s, δ such that the definition of strong inference is satisfied:

u	$X(u)$	$Y(u)$	$S(u)$	$T(u)$
1	1	1	1	1
2	2	−1	2	1
3	3	−1	3	2
4	4	−1	4	2
5	5	−1	5	3
6	6	1	2	1
7	7	−1	1	1
8	8	1	3	2
9	9	1	4	2
10	10	1	5	3

(s, δ)	δ_1	δ_1	δ_1	δ_1	δ_1
1	1	7	7	7	7
2	6	2	2	2	2
3	3	8	3	3	3
4	4	9	4	4	4
5	5	5	10	5	5

Theorem *A deterministic Turing Machine can be strongly inferred by a device iff*

$$\forall s \in S(U),\ |S^{-1}(s)| \geq 2.$$

Proof We show that the condition holds for both the partial recursive function description and the instantaneous description of Turing machines. Let $U := \mathbb{N}$. Define $S : U \to \mathbb{B}^*$ to be a function that maps integers to binary bit strings and $T : U \to \mathbb{B}^* \cup \eta$ as a function that maps U to the space of bits strings union with the non-halting output η.

Lemma *There exists a single-valued function $f : N \to M$ for all nonempty countable sets N, M.*
Proof Choose an element $m \in M$. Construct f such that $f(n) = m, n \in N$. *Remark to Wolpert: I'm not even sure if I should state this as a lemma, it seems rather clear.*

Since $\mathbb{N}$, $\mathbb{B}^*$, and $\mathbb{B}^* \cup \eta$ are countable sets, by the lemma there exists a single-valued f for each single-valued S and T. We only consider the restriction of f to single-valued functions because deterministic Turing Machines can be entirely described by single-valued partial functions.

Let $V = \{u : S^{-1}(s) = u\}$ for a value of $s \in S(U)$. Since V has at least two elements, let the first element of V be v_1 and the second element be v_2. Set $Y(v_1) = 1$ and $Y(v_2) = -1$. Then for each pair $(s, \delta_{t \in T(U)})$ choose $X(v_1) = x_1$ if $T(v_1) = t$ or otherwise choose $X(v_2) = x_2$. Since the choice of s was arbitrary, this holds for all (s, δ_t) pairs.

Now examine the instantaneous description of Turing Machines. The instantaneous description can be described by a function Δ mapping $Q \times \Lambda^k \to Q \times \Lambda^{k-1} \times \{L, S, R\}^k$, $k \geq 2$. Consider the functions $S : U \to Q \times \Lambda^k$ and $T : U \to Q \times \Lambda^{k-1} \times \{L, S, R\}^k$. Since the set of all finite subsets of $\mathbb{N}$ are countable, it follows that $|Q \times \Lambda^k| = |Q \times \Lambda^{k-1} \times \{L, S, R\}^k| = |\mathbb{N}|$.

We proceed similarly to the partial function portion of the proof. By the lemma, we restrict our focus such that Δ, S, and T are single valued functions. Then let $V = \{u : S^{-1}(s) = u\}$ for a value of $s \in S(U)$ such that the first element of V is v_1 and the second element is v_2. Set $Y(v_1) = 1$ and $Y(v_2) = -1$. For each pair $(s, \delta_{t \in T(U)})$ choose $X(v_1) = x_1$ if $T(v_1) = t$ or otherwise choose $X(v_2) = x_2$. Since the choice of s was arbitrary, this holds for all (s, δ_t) pairs.

Now for either case, suppose that $|V| < 2$ for some s. If $V = \varnothing$ then there exists no x that can force $S = s$. If $|V| = 1$, then we can assign $Y(v) = y \in \{-1, 1\}$. However, whatever value we assign, we cannot guarantee that $\delta_t(T(v)) = Y(v)$ for all t since $|T(U)| \geq 1$. $\square$

4 Inference Complexity

Definition Let $\mathcal{D}$ be an inference device and Γ be a function over U where $X(U)$ and $\Gamma(U)$ are countable and $\mathcal{D} > \Gamma$. Let the **size** of $\gamma \in \Gamma(U)$ be written as $\mathcal{M}_{\mu:\Gamma(\gamma)} = -\ln[\int_{\Gamma^{-1}(\gamma)} d\mu(u)1]$. Then the **inference complexity** of Γ with respect to $\mathcal{D}$ and measure μ is defined as:

$$\mathcal{C}_\mu(\Gamma; \mathcal{D}) \triangleq \sum_{\delta \in \wp(\Gamma)} min_{x:X=x \implies Y=\delta(\Gamma)}[\mathcal{M}_{\mu,X}(x)]$$

Remark: This is only a working definition and may be revised depending on its behavior.