AIT Blog

Clusters, Information Distances, and Left-Total Machines

Samuel Epstein*

October 11, 2022

Clustering

In this blog entry, I talk about clustering using the information distance. I'll also take this opportunity to showcase the benefit of left-total machines.

In cluster analysis, the goal is grouping objects in such a way that objects that are "similar" in some way are in the same group. It is a ubiquitous technique found in many research fields including computer vision, machine learning, information retrieval, and bioinformatics. Clustering is not a specific algorithms, but a collection of different approaches. However, in general there are two components, a distance (or similarity) measure and a dataset containing elements of a similar kind, such as songs or DNA sequences.

A question one might ask is whether there is a "core" to each cluster. That is, is there realizable object distilling the common components of the members? This makes more sense in clusters of DNA sequences rather than clusters of songs. To prove the existence of such cores, we need to prove it with respect to a distance measure. Just like in the previous blog, where I used a universal prior, in this blog, I'll use a universal distance. In AIT, there is such a notion, and it is known as the information distance [BGL+98],

$$\mathbf{E}_1(x,y) = \max\{\mathbf{K}(y|x), \mathbf{K}(x|y)\}.$$

The term $\mathbf{K}$ is the prefix free Kolmogorov complexity. $\mathbf{E}_1$ is universal over so called *admissible* distances. An admissible distance $\mathbf{D}$, is symmetric, satisfies the triangle inequality, is upper-semicomputable and normalized, that is

$$\sum_{y:y\neq x} 2^{-\mathbf{D}(x,y)} \leq 1.$$

Theorem ([BGL+98]) *For an appropriate constant c, let $\mathbf{E}(x,y) = \mathbf{E}_1(x,y) + c$ if $x \neq y$ and 0 otherwise. Then $\mathbf{E}(x,y)$ is a universal admissible metric in that every admissible distance $D(x,y)$ we have*

$$\mathbf{E}(x,y) <^+ \mathbf{D}(x,y).$$

*JP Theory Group. samepst@jptheorygroup.org

In fact there is a large literature about the real world applications of information distances. This involves replacing $\mathbf{E}_1$ with computable measures such as compression algorithms [CV05] or using the Google search engine [CV07].

Information Cores

Let $\mathbf{I}(x;\mathcal{H}) = \mathbf{K}(x) - \mathbf{K}(x|\mathcal{H})$ be the amount of information that x has with the halting sequene $\mathcal{H}$. We say $X \subset \{0,1\}^*$ is a (m,ℓ)-*cluster* if for all $x, y \in X$, $\mathbf{E}_1(x,y) \leq m$ and $\log|X| \geq \ell$. The following recent theorem shows that all clusters of a certain size with low information distance will contain an information core.

Theorem 1 ([Rom22]) *Let X be a (m,ℓ) cluster. There exists $z \in \{0,1\}^*$ where for all $x \in X$, $\mathbf{K}(z|x) <^+ O(m-\ell) + \mathbf{K}(m)$ and $\mathbf{K}(x|z) <^+ m + O(m-\ell) + \mathbf{K}(m)$.*

In fact, for non-exotic clusters, the information core will be a member of the cluster. One canonical cluster is as follows. Let z be a random string of length n, and X be all strings of length $2n$ that start with z. Then the string $z0^n$ can be seen as an information core that of X.

Theorem 2 ([Eps21]) *Let X be a (m,ℓ) cluster. There exists $z \in X$ where for all $x \in X$, $\mathbf{K}(z|x) <^{\log} 2(m-\ell) + \mathbf{I}(X;\mathcal{H})$.*

The last result states that non-exotic clusters will contain members that "clump" together with respect to the information distance. Thus there will exist two members of a cluster that are very close together, provided that the cluster is a large enough size and the information distance between members is small enough.

Theorem 3 ([Eps22a]) *Let X be an (m,ℓ) cluster. Then there exists $x, y \in X$ with $\mathbf{K}(y|x) <^{\log} \lceil l - 2k \rceil^+ + \mathbf{I}(X;\mathcal{H}) + \mathbf{K}(m,\ell)$.*

Left-Total Machines

There are many proofs in the literature that use the bits Chaitin's Omega, $\Omega \in \mathbb{R}$. This includes finite prefixes of Ω, sometimes denoted $\Omega(1\ldots t)$, as well as a lower approximation $\Omega^t < \Omega$ given some time resource t. Take for example, Theorem 3.3.1, in [GÍ3], Proposition 20 in [VS17], or the previous blog entry. However as manipulations become more involved, there is utility in denoting with these concepts in another way using so-called teft-total machines, first appearing in [EL11], also appearing in [Lev16]. In this blog, I provide a concise description of left-total machines. For more details on left-total machines, we refer readers to the self-contained Section 7 in [Eps22a].

We say $x \in \{0,1\}^*$ is total with respect to a machine if the machine halts on all sufficiently long extensions of x. More formally, x is total with respect to T_y for some $y \in \{0,1\}^{*\infty}$ iff there exists a finite prefix-free set of strings $Z \subset \{0,1\}^*$ where $\sum_{z\in Z} 2^{-\|z\|} = 1$ and $T_y(xz) \neq \perp$ for all $z \in Z$. We say (finite or infinite) string $\alpha \in \{0,1\}^{*\infty}$ is to the "left" of $\beta \in \{0,1\}^{*\infty}$ and use the notation $\alpha \lhd \beta$ if there exists an $x \in \{0,1\}^*$ such that $x0 \sqsubseteq \alpha$ and $x1 \sqsubseteq \beta$. A machine T is left-total if for all auxiliary strings $\alpha \in \{0,1\}^{*\infty}$ and for all $x, y \in \{0,1\}^*$ with $x \lhd y$, one has that $T_\alpha(y) \neq \perp$ implies that x is total with respect to T_α. For the rest of the paper, we assume that the universal Turing machine U is left-total. Let $(p0)^- = (p1)^- = p$.

Proposition 1 *There exists a unique infinite sequence $\mathcal{B}$ with the following properties.*

1. *All the finite prefixes of $\mathcal{B}$ have total and nontotal extensions.*
2. *If a finite string has total and nontotal extensions, then it is a prefix of $\mathcal{B}$.*
3. *If a string b is total and b^- is not, then $b^- \sqsubset \mathcal{B}$.*

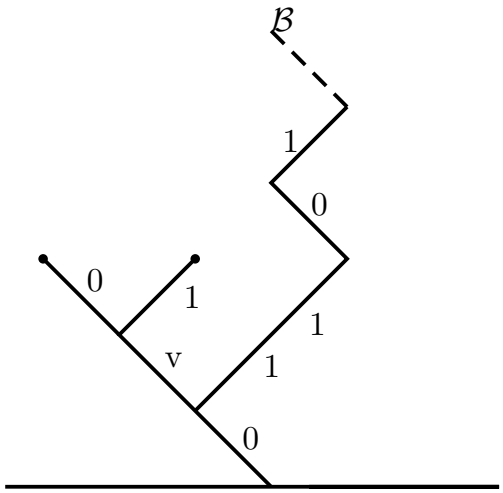

Figure 1: The above diagram represents the domain of the universal left-total algorithm U, with 0 bits branching to the left and 1 bits branching to the right. The strings in the above diagram, $0v0$ and $0v1$, are halting inputs to U with $U(0v0) \neq \perp$ and $U(0v1) \neq \perp$. Therefore, $0v$ is a total string. The infinite border sequence $\mathcal{B} \in \{0,1\}^\infty$ represents the unique infinite sequence such that all its finite prefixes have total and nontotal extensions. All finite strings branching to the right of $\mathcal{B}$ will cause U to diverge.

We call this infinite sequence $\mathcal{B}$, "border" because for any string $x \in \{0,1\}^*$, $x \lhd \mathcal{B}$ implies that x is total with respect to U and $\mathcal{B} \lhd x$ implies that U will never halt when given x as an initial input. It is equal to the binary expansion of Chaitin's Omega. Figure 1 shows the domain of left-total U with respect to $\mathcal{B}$. For total string b, we define the busy beaver function,

$$\mathbf{bb}(b) = \max\{\|x\| : U(p) = x, p \lhd b \text{ or } p \sqsupseteq b\}.$$

Using Figure 1, $\mathbf{bb}(v0) \leq \mathbf{bb}(v1)$ and $\mathbf{bb}(v0) \leq \mathbf{bb}(v)$.

Total Computable Conversion Function

Left-total machines can be used in proofs to change partial recursive functions into total computable ones. Similarly, for more advanced proofs, it can turn lower-computable semi-measures (namely the algorithmic probability $\mathbf{m}$) into recursive ones. This was done in the proofs of all the theorems of the first blog post.

Definition 1 *We define the following two functions, representing the conversion distance between two strings.* $\mathbf{E}_0$ *was introduced in [BGL+98].*

- $\mathbf{E}_0(x,y) = \min\{\mathbf{K}(p) : p(x) = y, p(y) = x\}$.
- $\mathbf{F}_0(x,y) = \min\{\mathbf{K}(f) : f \text{ is total computable}, f(x) = y, f(y) = x\}$.

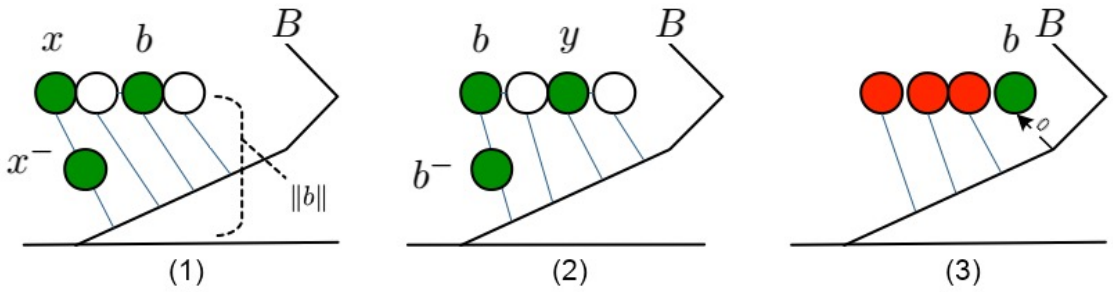

Figure 2: The above diagram represents the domain of the universal left-total Turing machine U and uses the same conventions as Figure 1, with 0s branching to the left and 1s branching to the right. It shows all the total strings of length $\|b\|$, including b. The large diagonal line is the border sequence, B. A string c is marked green if $(x, y) \in$ MATCHING(k, c). By definition, b is a shortest green string. If x is green and total, and $x \lhd y$, and y is total, then y is green, since $\mathbf{bb}(x) \leq \mathbf{bb}(y)$. Furthermore, if x is green and total and x^- is total, then x^- is green, as $\mathbf{bb}(x) \leq \mathbf{bb}(x^-)$. It cannot be that there is a green $x \lhd b$ with $\|x\| = \|b\|$. Otherwise, x^- is total, and thus, it is green, causing a contradiction because it is shorter than b. This is shown in part (1). Furthermore, there cannot be a green y, with $b \lhd y$ and $\|y\| = \|b\|$. Otherwise, b^- is total and thus green, contradicting the definition of b. This is shown in part (2). Thus, b is unique, and since b^- is not total, by Proposition 1, b^- is a prefix of the border, as shown in part (3). Thus, an algorithm returning a green string of length $\|b\|$ will return b.

We start by recreating the proof showing that the min conversion distance is equal, up to a logarithmic error term, to $\mathbf{E}_1$.

Theorem 4 ([BGL+98])

$$\mathbf{E}_0(x, y) =^{\log} \mathbf{E}_1(x, y).$$

Proof. The first inequality is straitforward. Let $k = \mathbf{E}_1(x, y) + 1$. We describe the MATCHING(k) algorithm. Let $G = (V, E)$ be an infinite labelled graph where every string has a corresponding vertex and undirected edges are enumerated and labeled with 2^{k+1} colors. The algorithm enumerates undirected edge (x, y) if $\mathbf{K}(y|x) < k$ and $\mathbf{K}(x|y) < k$. For any vertex x, there are at most 2^k edges (x, y) enumerated, because an edge implies $\mathbf{K}(y|x) < k$. Thus one can dynamically color the graph with 2^{k+1} colors such that no two edges with a common vertex share a color.

Let $c \in \{0, 1\}^{k+1}$ be the string corrosponding to the color of of the edge (x, y). There is a function f that runs MATCHING(k) and on input x follows c and outputs y. Similarly, on input y, the function follows c and outputs y. Thus $\mathbf{K}(f) <^+ \mathbf{K}(c, k) <^+ k + \mathbf{K}(k)$, implying $\mathbf{E}_0(x, y) <^{\log} \mathbf{E}_1(x, y)$.

□

Theorem 5

$$\mathbf{E}_1(x, y) <^+ \mathbf{F}_0(x, y) <^{\log} \mathbf{E}_1(x, y) + \mathbf{I}((x, y); \mathcal{H}).$$

Proof. The first inequality is straitforward. Let $k = \mathbf{E}_1(x, y) + 1$. Let MATCHING$(k, d)$ be the matching algorithm described above but it halts after $\mathbf{bb}(d)$ steps for some total string d. Let b be a shortest total string such that MATCHING(k, b) enumerates the link (x, y) with the

color c. Let $f(z)$ be the function defined as follows. It runs the $\textsc{Matching}(k, b)$ algorithm then, starting at z, it follows the edge marked c and returns the label of the adjacent vertex. If there is no edge marked c, then f returns 0. So f is total computable, $f(x) = y$, and $f(y) = x$. Thus $\mathbf{F}_0(x, y) <^+ \mathbf{K}(f) <^+ \mathbf{K}(k) + k + \mathbf{K}(b) <^{\log} \mathbf{E}_1(x, y) + \mathbf{K}(b)$. It remains to prove that $\mathbf{K}(b) <^{\log} \mathbf{I}((x, y); \mathcal{H}) + \mathbf{K}(k)$.

From Lemma 2 in [Eps22b], we have $\mathbf{I}(b; \mathcal{H}) <^+ \mathbf{I}((x, y); \mathcal{H}) + \mathbf{K}(b|(x, y))$. Furthermore since b is total and b^- is not, by Proposition 1, b^- is a prefix of border, which is the binary expansion of Chaitin's Omega. Thus $\mathbf{K}(b) <^{\log} \mathbf{I}(b; \mathcal{H})$. Furthermore $\mathbf{K}(b|(x, y)) <^+ \mathbf{K}(\|b\|, k)$, because there is a program, that enumerates total strings of length $\|b\|$ from left to right and return the first string d such that $(x, y) \in \textsc{Matching}(k, d)$. This returned string is b, as shown by Figure 2. Thus $\mathbf{K}(b) <^{\log} \mathbf{I}(b; \mathcal{H}) <^{\log} \mathbf{I}((x, y); \mathcal{H}) + \mathbf{K}(\|b\|, k) <^{\log} \mathbf{I}((x, y); \mathcal{H}) + \mathbf{K}(k)$. This is because since b is random, $\mathbf{K}(\|b\|) = O(\log \mathbf{K}(b))$. □

References

[BGL+98] C. Bennett, P. Gacs, M. Li, P. Vitanyi, and W. Zurek. Information distance. *IEEE Transactions on Information Theory*, 44(4):1407–1423, 1998.

[CV05] R. Cilibrasi and P. Vitanyi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

[CV07] R. Cilibrasi and P. Vitanyi. The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370–383, 2007.

[EL11] Samuel Epstein and Leonid Levin. On sets of high complexity strings. *CoRR*, abs/1107.1458, 2011.

[Eps21] S. Epstein. On the conditional complexity of sets of strings. *CoRR*, abs/1907.01018, 2021.

[Eps22a] S. Epstein. The kolmogorov birthday paradox. *CoRR*, abs/2208.11237, 2022.

[Eps22b] S. Epstein. The outlier theorem revisited. *CoRR*, abs/2203.08733, 2022.

[GÍ3] P. Gács. Lecture notes on descriptional complexity and randomness, 2013.

[Lev16] L. A. Levin. Occam bound on lowest complexity of elements. *Annals of Pure and Applied Logic*, 167(10):897–900, 2016.

[Rom22] A. Romashchenk. Clustering with respect to the information distance. *Theoretical Computer Science*, 929:164–171, 2022.

[VS17] Nikolay K. Vereshchagin and Alexander Shen. Algorithmic statistics: Forty years later. In *Computability and Complexity*, pages 669–737, 2017.