Finite automata with continuous input

András Kornai

HAS Computer and Automation Research Institute
H-1111 Budapest Kende u 13-17, Hungary
andras@kornai.com

Abstract
Euclidean Automata (EA) are finite state computational devices that take continuous parameter vectors as input. We investigate decompositions of EA into simpler Euclidean and classical finite state automata, relate them to better known computational devices such as artificial neural nets and electronic devices, and show that they are plausible Big Mechanism candidates.

1. Introduction

Euclidean automata (EA) were introduced, and motivated from the perspective of Artificial Intelligence, in [8]. For convenience, we repeat some of the definitions and examples here, but the focus of the current paper is with the decomposition of EA and relating the concept to better known computational frameworks, subjects that could not be discussed in the earlier work. EA are modified Finite State Automata (FSA) that draw their input not from a finite alphabet as usual, but rather from vectors from a parameter space P, typically $\mathbb{R}^n$. For quantum applications, $\mathbb{C}^n$ would also be of interest, but we concentrate on the real case.

Definition 1.1 A *Euclidean automaton* (EA) over a parameter space P is defined as a 4-tuple $(\mathcal{P}, I, F, T)$ where $\mathcal{P} \subset 2^P$ is a finite set of states given as subsets of P; $I \subset \mathcal{P}$ is the set of initial states; $F \subset \mathcal{P}$ is the set of accepting states; and $T : P \times \mathcal{P} \to \mathcal{P}$ is the transition function that assigns for each parameter setting $\vec{v} \in P$ and each state $s \in \mathcal{P}$ a next state $t = T(\vec{v}, s)$ that satisfies $\vec{v} \in t$.

Thus, while the input is drawn (in discrete steps) from a continuous domain, EA are still finite in that *states* are simply subsets P_i of P indexed from a finite index set S. If $P_i \cap P_j = \emptyset$ for all $i, j \in S$ we call the EA *deterministic*, if $\bigcup_{i \in S} P_i = P$ we call it *complete*. If all P_i are open subsets of the the Euclidean space, we call the EA *open*.

In problems of linguistic pattern classification the number of classes (e.g. the set of valid characters in optical character recognition (OCR) or the set of phonemes in automated speech recognition (ASR)) is finite, and small continuous deformations of the input leave the output unchanged, suggesting that the classifier is an open EA. The main problem with this is that we cannot partition $\mathbb{R}^n$ or $\mathbb{C}^n$ into finitely many disjoint open sets. Approximate solutions thus

[†]Work supported by OTKA grant #82333 and by the European Union and the European Social Fund through project FuturICT.hu (grant # TAMOP-4.2.2.C-11/1/KONV-2012-0013).

must give up non-overlapping, e.g. by permitting probabilistic or fuzzy outcomes, or exhaustiveness, e.g. by leaving 'gray areas' near decision boundaries where the system produces no output. As discussed in [8], EA take the first route, sacrificing determinism (non-overlapping). This enables modeling of the nondeterminism actually seen in cases of perceptual hysteresis [12]. In fact, we observe that the appearance of hysteresis in such situations is not an accident:

Observation 1.1 There is no deterministic classification of a connected parameter space $P \subset \mathbb{R}^n$ by some open EA.

Proof By definition, connected subspaces of Euclidean space cannot be the discrete union of open sets.

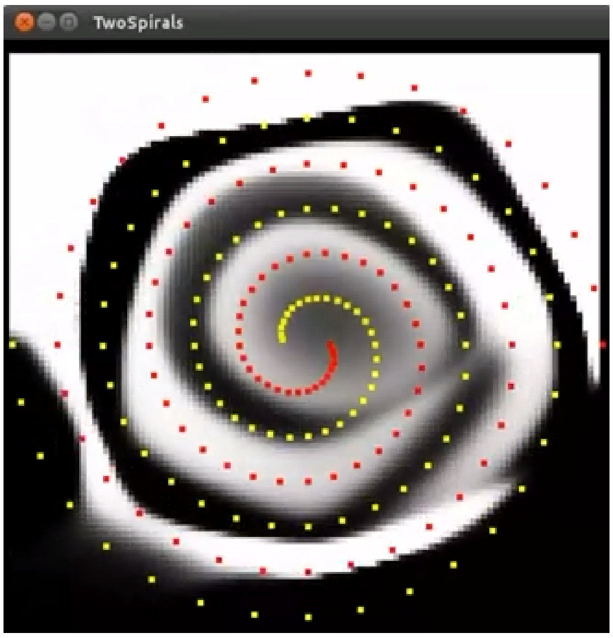

Figure 1: Decision boundary in 2-20-10-1 layer perceptron

As an example, consider the multilayer perceptron depicted in Fig. 1, trained in [5]. In spite of the complexity of the decision boundary, the EA with equivalent behavior has only two states, corresponding to the black and the white subsets of the image. The input vectors are two-dimensional, and there is no output to speak of (we could designate one of the two states final). To account for output as well, consider the following definitions (repeated from [8] for convenience):

Definition 1.2 A *Euclidean transducer* (ET) over a parameter space P is defined as a 5-tuple $(\mathcal{P}, I, F, T, E)$ where $\mathcal{P}, I, F$, and T are as in Def. 1.1 and E is an emission function that assigns a string (possibly empty) over a finite alphabet Σ to each transition defined by T.

Definition 1.3 A *Euclidean Eilenberg Machine* (EEM) over a parameter space P is defined as a 5-tuple $(\mathcal{P}, I, F, T, R)$ where $\mathcal{P}, I, F$, and T are as in Def. 1.1 and R is a mapping $P \times \mathcal{P} \rightarrow P$ which assigns to each transition a (not necessarily linear, or even deterministic) transformation of the parameter space.

2. Decomposition

The definition of EA leaves open the possibility that the parameter space P is embedded in $\mathbb{R}^n$ in a partially discrete manner, e.g. as indexed subsets of lower-dimensional spaces. Consider, for example, a three-stop elevator running from the basement to the top (first) floor of some building. This will have both continuous parameters, such as the reading from the position sensor, and discrete parameters, such as the reading from the engine sensor, with only three possible values "going up", "stopped", and "going down". Some of the parameters, such as the reading from the weight sensor, are seemingly continuous but effectively quantized to two discrete values, "above safety limit" or "below safety limit". In such cases, we may want to select *canonical representatives* $\vec{p_i}$ from P_i. (Other input values, pertaining to the state of call buttons at each floor and inside the cab, to accelerometer readings, or to sensors for AC power quality could be added, but we don't aim at realistic detail here.)

As long as the parameters can be isolated from one another we can view P as being the direct product of smaller parameter spaces P_i, some Euclidean, some discrete. Isolating the parameters is easy enough for elevators with different sensors, but not at all trivial in pattern recognition tasks where the individual coordinates, such as spectral peaks in ASR, can show all kinds of interdependence. There is notable uncertainty how we wish to embed the discrete spaces in $\mathbb{R}$, for example two-valued parameters are often encoded as 0 or 1, but often as -1 or $+1$, and there is no easy way to select a canonical embedding. In many applications, n-valued parameters are encoded as $0, \ldots, n-1$, in others as $1, \ldots, n$, in yet others as $0, 1/(n-1), 2/(n-1), \ldots, 1$. Let us first consider a family M of $P \to P$ mappings with the goal of replacing one conventional encoding by another. As the examples show, such mappings are typically taken from continuous/differentiable families, but are not necessarily linear.

Definition 2.1 An EA Ψ is the *homeomorphic image* of Φ under a mapping $m \in M$ iff for any sequence of inputs $\vec{v}_1, \ldots, \vec{v}_n$ we have $\Psi(m(\vec{v}_1), \ldots m(\vec{v}_n)) = m(\Phi(\vec{v}_1, \ldots, \vec{v}_n))$.

Here we assume that both Φ and Ψ are started from the same unique initial state (if we permit several initial states the definition needs to be complicated accordingly) and that equality means equality of result state. This is meaningful, since m naturally maps not just inputs on inputs, but also EA states (subsets of parameter vectors) on one another. We will say Φ and Ψ are *isomorphic* if they are homomorphic images of each other under some m and m^{-1}.

Definition 2.2 The *skeleton* of an EA Φ is a standard (Mealy) FSA whose alphabet corresponds to canonical representatives from each Boolean atom of $\mathcal{P}$.

In the deterministic case, this is also a Moore automaton, as there is a one to one correspondence between input letters and automaton states. As is clear from Definition 1.1, the sequential behavior of EA is relatively simple in this case, since the result state depends only on the input, and not on the previous state. In the nondeterministic case (which is the more interesting case as per Observation 1.1) we may not be able to select distinct canonical representatives for each state P_i, or even for the set of Boolean atoms formed by the P_i. Here skeleta must be replaced by what we will call *subjective* EA, unique only up to canonical isomorphism, but sacrificing the one to one correspondence between state set and input set.

Besides the well understood serial and parallel modes of composition (see 3.2 for examples), EA admit a further possibility. This can already be illustrated in the simplest case of a mixed parameter space, where P_c, the continuous part of the parameter space, is just $\mathbb{R}$, and P_d, the discrete part, is just a binary choice $\top, \bot$. We may think of an EA Φ over $P = P_c \times P_d$ as being composed of two simpler EA $\Phi_\top$ and $\Phi_\bot$ by means of a real parameter p that *influences* whether the $\Phi_\top$ or the $\Phi_\bot$ behavior dominates. Importantly, the parameter that does the influencing may just be the input parameter, providing a crude form of memory, as in Example 3 of [8].

3. Relating EA to other computational mechanisms

3.1 Artificial neural nets In classification tasks our interest is with the inverse images C_i of the possible outputs i. As our example in Fig. 1 shows, EA offer a method for directly encoding the information concerning the shape of the C_i where it belongs, in $\mathbb{R}^n$, where n is the dimension of the input parameter vector, rather than in $\mathbb{R}^{m \times m}$, the matrix of connection strengths. As is well known, in pattern recognition a great deal depends on the preprocessing of the signal, and using EA can make this dependence explicit. For example, consider the "two circles" data set presented in [10] reproduced here as Fig. 2: while it is evident that no linear separator (simple NN) exists, transforming the data to a system of polar coordinates around the center of gravity of the datapoints would make the task trivial. In ASR, we routinely apply a far more elaborate sequence of data transformation steps (power cepstra [1], mel warping [3], and delta cepstra [6]) to make the data manageable. Altogether, the use of EA is expected to bring new insights, especially for the increasingly popular but not yet well understood "deep learning" neural net architectures such as LSTM [7].

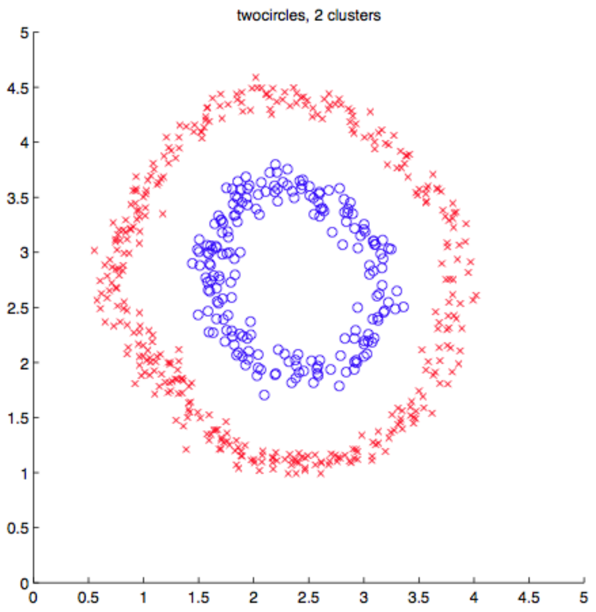

Figure 2: "Two circles" data from Ng (2001)

3.2 Electronics For many applications it makes sense to define the initial state as a parameter region P_0 that has no overlap with the other states of the automaton (even for EA not otherwise deterministic), since this will guarantee that we can *reset* the EA to the initial state by making sure that there are outbound transitions from every P_i. If we have another region we can reset to, we obtain an EA corresponding to the classical *flip-flop* (or *latch*) circuit. As discussed in [8], we can also obtain classical circuits with hystersis, such as a Schmitt trigger [13]. By creating EA corresponding to elementary building blocks such as transistors, all forms of logic circuitry operating on continuous variables such as voltages could be recast as networks of EEMs. Modeling of transient behavior is not facilitated by the framework.

As is standard in logic design, digital circuitry can be conceptualized as series-parallel composition of standard FSTs, either with output string length limited to 1 (otherwise issues of timing and synchrony become paramount) or with clock signals added in. For semi-analog circuitry, where the outputs of each buiding block can be characterized as constant values (or values with very little variation) the same series-parallel conceptualization is available with Eucldean transducers (ETs), as long as we take the outputs of the upstream ETs to be the canonical representatives of the inputs of the downstream ETs. This means that in principle all physical models of digital computation, realized by discrete electronics as they are in current computers, are within scope of the EA/ET/EEM model as defined in D1.1-3. These are theoretical models, unlikely to gain much traction in circuit design, where transitional behavior and synchrony are highly relevant, but the connection makes clear that EA don't suffer from the kind of realizability problems that plague many theoretical computing devices from quantum gates to memristors.

3.3 WFSA, Bayes/Markov nets Here the relationship is much more tenuous, in that both weighted FSA (WFSA) and Bayesian nets take discrete inputs (in the Bayesian case just 0-1, for WFSA strings from an arbitrary finite alphabet) and produce continuous values on output, not the other way round as in EA. In the case of WFSA and WFST, there is no way to pass the weights to later automata, so in investigating how automata can influence each other we are restricted to communication by means of strings from a discrete alphabet. While the dificulties are not insurmounatble (after all, we can always encode good approximations of continuous values in discrete strings) the system is highly unnatural, requiring a great deal of decoding and re-encoding to express any simple arithmetical relation such as the activation of a node being determined by the sum of the inbound activation levels.

Bayesian/Markov nets offer a highly coherent way of thinking about probabilistic influences, but here the technical difficulties are even worse. There is nothing to say, at least in principle, that the random variables at the nodes couldn't be continuous, but to compute the marginal density of a child node we have to integrate out all the parents. Really the only parametric family that makes sense in this setting is (multivariate) normal [9], which excludes many systems of great interest to which we now turn.

3.4 Big Mechanism Following the success of Big Data, DARPA has started a search towards a *Big Mechanism* that is capable of modeling cancer signaling pathways, the brain, climate, the economy, and other large interconnected systems, focusing initial attention on cancer signaling pathways. Acquiring the data from medical journals and databases, requiring both considerable standardization of terminology and sophisticated natural language processing (NLP) capabilities, are seen as part of the task of developing Big Mechanisms, which are "large, explanatory models of complicated systems in which interactions have important causal effects" [2]. In the graphical language standard for systems biology [11], which notes the functional analogy with electronic circuit diagrams and algorithm block diagrams, we see both discrete, FSA-like elements such as a channel being open or closed, a muscle being tense or relaxed, and continuous, additive elements such as the level of various chemicals. Eilenberg Machines [4] were developed, for the discrete case, with block diagrams in mind, and EEM provide the kind of continuous generalization that makes e.g. stochiometry possible to model.

References

[1] B. P. BOGERT, M. J. HEALY, J. W. TUKEY, The quefrency alanysis of time series for echoes: cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking. In: M. ROSENBLATT (ed.), *Proceedings of the Symposium on Time Series Analysis*. Wiley, 1963, 209–243.

[2] P. COHEN, Big Mechamism Program. 2014. DARPA BAA, accessed 5/2014.
`http://www.darpa.mil/Our_Work/I2O/Programs/Big_Mechanism.aspx`

[3] S. B. DAVIS, P. MERMELSTEIN, Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28 (1980) 4, 357–366.

[4] S. EILENBERG, *Automata, Languages, and Machines*. A, Academic Press, 1974.

[5] A. FABISCH, *Two Spirals Problem Solved in Compressed Weight Space*, 2011.
`https://www.youtube.com/watch?v=MkLJ-9MubKQ`

[6] S. FURUI, Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34 (1986) 1, 52–59.

[7] S. HOCHREITER, J. SCHMIDHUBER, Long Short-Term Memory. *Neural Computation* 9 (1997) 8, 1735–1780.

[8] A. KORNAI, Euclidean Automata. In: M. WASER (ed.), *Implementing Selves with Safe Motivational Systems and Self-Improvement*. Proc. AAAI Spring Symposium, AAAI Press, 2014, 25–30.

[9] S. LAUR, Bayesian Networks with Continious Distributions. 2009. Unpublished class notes, accessed 5/2014.
`http://bit.ly/1jJs2M6`

[10] A. Y. NG, M. I. JORDAN, Y. WEISS, On Spectral Clustering: Analysis and an algorithm. In: *Advances in neural information processing systems*. MIT Press, 2001, 849–856.

[11] N. L. NOVÈRE, The Systems Biology Graphical Notation. *Nature Biotechnology* 27 (2009), 735–741.

[12] S. POLTORATSKI, F. TONG, Hysteresis in the Perception of Objects and Scenes. *Journal of Vision* 13 (2013) 9, 672.

[13] O. H. SCHMITT, A thermionic trigger. *Journal of Scientific Instruments* 15 (1938) 1, 24.