Super Turing Machines

Kyle Doney[1]

College of Information and Computer Sciences
UMass Amherst

April 16th, 2020

Table of Contents

Turing Machines

A Turing Machine is composed of a finite state machine and an infinite read/write tape.

Definition

A Turing Machine (TM) M is the tuple $M = (Q, \Gamma, \delta, q_0)$, where Q is a finite set of states, Γ is the finite alphabet, $q_0 \in Q$ is the initial state, and δ the transition function.

Definition

The transition function $\delta : Q \times \Gamma \rightarrow Q \cup \{h\} \times \Gamma \times \{\leftarrow, \rightarrow, -\}$

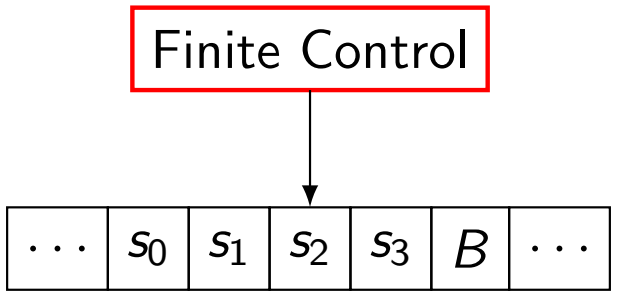


Figure: Example Turing Machine

Turing Computability

Definition

S is a *recursive* (*decidable*) set $\iff \exists$ TM M where $M(x)$ halts **AND** $M(x) = f_S(x)\ \forall x \in \mathbb{N}$

There exist *countably infinite* Turing Machines ($\aleph_0$).

From **Cantor's Theorem**, there exist *uncountably infinite* sets of Natural numbers ($> \aleph_0$).

Therefore there exist sets Turing machines can not compute.

Hypercomputation

Examples of Hypercomputation:

1. Turing machines that take *advice*.
2. Oracle Turing Machines
3. Super Turing (e.g. Analog Recurrent Neural Networks (ARNNs)) .
4. Many other models exist.

Definition

Turing Machines that take **Advice** have an *advice string* α_n for each input size n. This α_n is a secondary input to the TM.

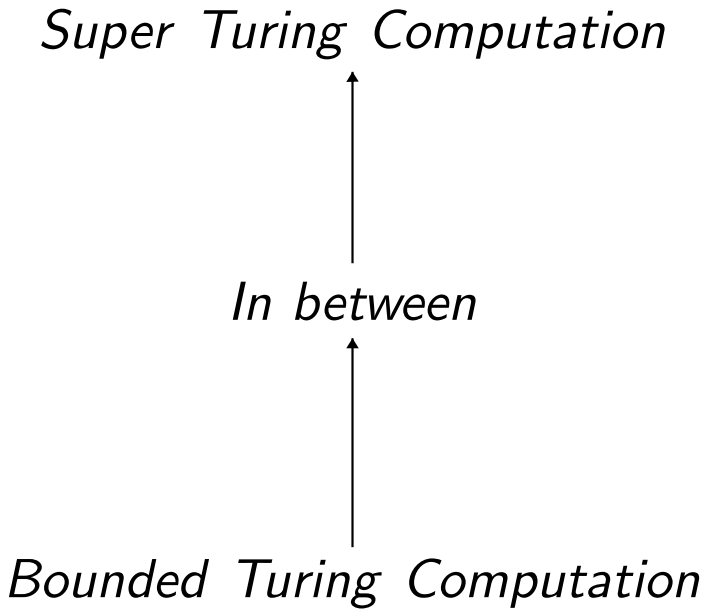
Adaptivity

The Turing Model is rigid.

My contention is that machines can be constructed which will simulate the behaviour of the human mind very closely. They will make mistakes at times, and at times they may make new and very interesting statements. (Turing) [5]

Super Turing

Consider a hierarchy of computational models structured on their ability to adapt to a changing environment.



Definition

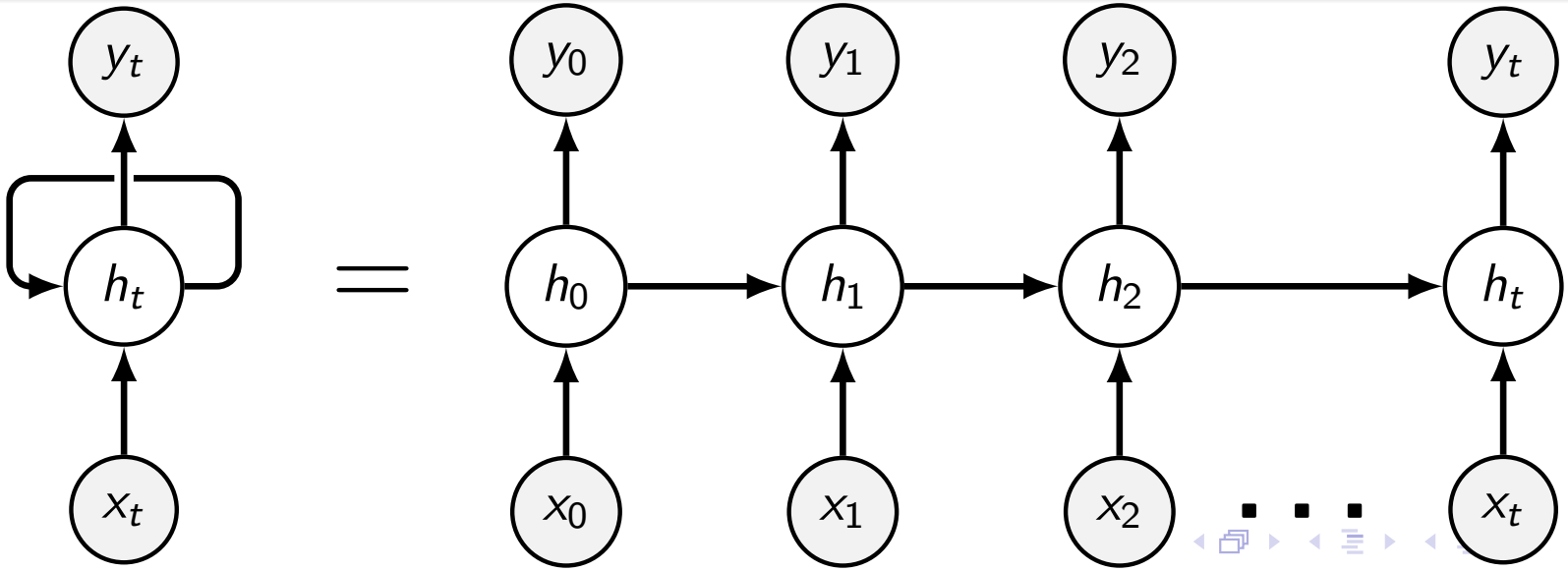
A **Super Turing Machine** can be thought of as a machine that builds as set of Turing Machines each for a specific type of environment, and then chooses which one is best to execute.

Definition

In the RNN each Neuron $i \in N$ updates it's state based on the following equation at timestep t:

$$x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{ij} \cdot x_j(t) + \sum_{j=1}^{M} b_{ij} \cdot u_j(t) + c_i\right), i = 1, ..., N$$

a_{ij} and b_{ij} are the weighted connections, c_i are the weighted bias,
x_j are the activations of the neurons, and u_j are the M new external inputs.



Analog Recurrent Neural Networks

Bounded number of neurons which communicate numbers on the analog [0,1] range as opposed to the standard RNN's σ functions producing the binary $\{0, 1\}$.

Definition

The ARNN **activation function** is defined as the piece-wise function:

$$\sigma(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \leq x \leq 1 \\ 1 & \text{if } x > 1 \end{cases}$$

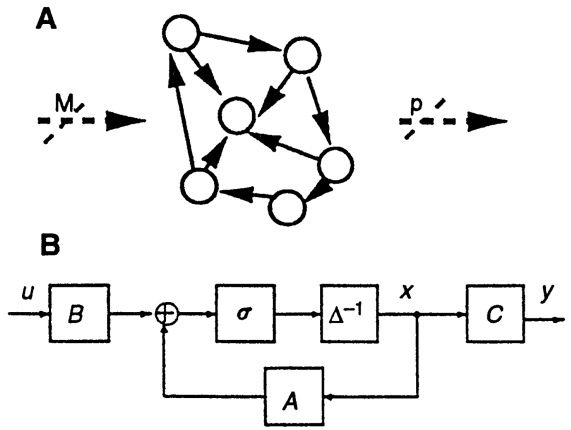


Fig. 1. The analog recurrent neural network (ARNN). (**A**) Graphic view. (**B**) Engineering view.

Figure: [2]

Integer Weight ARNNs

Lemma (1)

If we restrict the weights of the ARNN to be Integers, then the set of languages computed by ARNN-Int is exactly the set of regular languages.

Lemma (2)

The class $\mathcal{L}$ of languages accepted by offline finite automata is exactly the class of regular languages.

Integer Weight ARNNs

Given a a integer network $\mathcal{N}$ containing N neurons that accepts the language L. We define an offline finite automaton $\mathcal{M} = (Q, \{0,1\}, \delta, q_0, F)$ as follows:

1. We identify the input $(D(t), V(t))$ to $\mathcal{N}$ with the values $\{0, 1, \$\}$ using the encoding $f((0,1)) = 0$, $f((1,1)) = 1$, $f((0,0)) = \$$.
2. Set $Q = \{0,1\}^N$ and $q_0 = 0^N$. This identifies each state of $\mathcal{M}$ with the activations of all the neurons.
3. Assume $G(t)$ and $H(t)$ are the first and second neurons in the state enconding. We denote $q[i]$, the i-th coordinate of state q. Then our final states $F = \{q \in Q | q[1] = 1, q[2] = 1\}$.
4. The transition function is the update equation $q^+ = \sigma(Aq + Bu + c)$.

Rational Weight ARNNs

Lemma (1)

If we restrict the weights of the ARNN to be Rational numbers, then the set of languages computed by ARNN-Rat is the set of recursive languages.

Lemma (2)

The class $\mathcal{L}$ of languages accepted by p-stack Turing machines is exactly the class of recursive languages.

Let $\alpha = \alpha_1\alpha_2...$ be the stack from top to bottom.

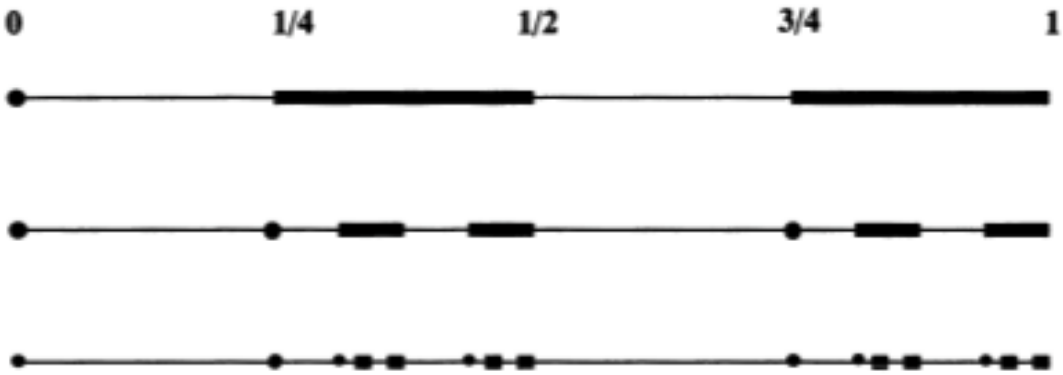


Figure: 4-Cantor Set

TM with Advice VS Oracle Turing Machines

Definition (Turing Machines with Advice)

Turing Machines that take **Advice** have an *advice string* α_n for each input size n. This α_n is a secondary input to the TM.

Definition (**P**/**poly**)

A language $\mathcal{L} \in \mathbf{P}/\mathbf{poly}$ if there exists a polynomial time Turing machine M and a set of advice strings $\{a_0, a_1, ...\}$ such that $|a_n| \leq n^{O(1)}$ such that for every string x, $x \in \mathcal{L}$ iff $M(x, a_{|x|})$ returns Accept.

A Unary Problem is of the form: $\mathcal{L} = \{1^n | \ ...\}$. We then clearly have some $M(1^n, a_{|n|})$ which decides $\mathcal{L}$.

Definition (Oracle Turing Machines)

An *oracle* Turing machine is a TM M that has a special read-write tape we call the *Oracle Tape* and three special states q_{query}. q_{yes}, q_{no}. To run machine M, we specify some language $O \subseteq 0, 1^*$ to act as the oracle for M.

ARNNs are Super Turing

Lemma

If we allow the weights of the ARNN to be Reals, then ARNNs are able to recognize the complexity class P/poly.

Note that although the real weights are defined with unbounded precision, it is possible to show that "linear precision suffices".

Conclusion

1. Super Turing Computation is a model of computation which allows for environmental Adaptivity and learning.
2. Super Turing Computation can be represented by a series of standard Turing Machines.
3. Analog Neural Networks are just one of many models of Super Turing Computation.
4. There exist many interesting connections between well studied complexity classes and ARNNs.

Works Cited

[1] Cabessa, J., and Siegelmann, H. T.
The computational power of interactive recurrent neural networks.
Neural Computation 24, 4 (2012), 996–1019.

[2] Siegelmann, H. T.
Computation beyond the turing limit.
Science 268, 5210 (1995), 545–548.
00374.

[3] Siegelmann, H. T.
Turing on super-turing and adaptivity.
Progress in biophysics and molecular biology 113, 1 (2013), 117–126.

[4] Siegelmann, H. T., and Sontag, E. D.
Analog computation via neural networks.
Theoretical Computer Science 131, 2 (1994), 331–360.

[5] Turing, A. M.
Intelligent machinery, a heretical theory.
The Turing test: Verbal behavior as the hallmark of intelligence 105 (1948).