Relativizations of P=NP and P≠NP in ARNN Model

Raimundo Coelho Leong

November 20, 2010

Abstract

We will provide a formalization of the use of real weights in analog recurrent neural networks as a restricted class of oracles, closely related to physical oracles, and prove that the famous $P = NP$ problem relativizes with respect to deterministic and nonderterministic analog recurrent neural networks with real weights working in polynomial time. As a direct consequence, these devices have restricted computational power.

1 Introduction

Warren McCulloch and Walter Pitts in [11] first presented a model of threshold logic units, forming what is called the analog recurrent neural networks (ARNN for short). These neural networks has been proved by Kleene to be equivalent to abstract devices called finite automata. These threshold units together with a Heaviside function as activation (the McCulloch-Pitts activation function)

$$\sigma_d(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases},$$

can hold binary values only. The system consists on a finite collection of state variables $(x_1(t), \ldots, x_n(t))$ with input lines $u_1(t), \ldots, u_m(t)$. Each of the units X_i holds the value x_i in the correspondent time step and it evolves following the dynamics

$$\begin{cases} x_1(t+1) = \sigma_1(a_{11}(t)x_1(t) + \cdots + a_{1n}(t)x_n(t) + b_{11}(t)u_1(t) + \cdots + b_{1m}(t)u_m(t) + c_1(t)) \\ \vdots \\ x_n(t+1) = \sigma_n(a_{n1}(t)x_1(t) + \cdots + a_{nn}(t)x_n(t) + b_{n1}(t)u_1(t) + \cdots + b_{nm}(t)u_m(t) + c_n(t)) \end{cases}$$

The coefficients, in this case integers, are called weights. This model has been strongly criticized during the Macy's Conferences for its lack of biological and neurophysiological plausibility (see [13]) but they surely satisfy the requirements of a computational device: the inputs are fed in in binary fashion; the output streams are encoded in binary; the system has a finite dimension, corresponding to a finite control.

Neural networks with rational weights and a piecewise linear activation function (the saturated sigmoid)

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

was widely used. As the rational numbers can encode information of arbitrary (but finite) size, for example, in Cantor subsets of the interval $[0, 1]$ ([16], [17]), units holding rational values resemble infinite tapes in Turing machines. In fact, they have been proven by Hava Siegelmann and Eduardo Sontag in [17] to be equivalent and, moreover, that there is a universal architecture of rational neural networks that simulates a given Turing machine in real time.

In further research, neural networks carrying real values were shown to have the same computational power as families of Boolean circuits ([16]). If no time bound is set, they can decide any set! And when a polynomial time bound is imposed, they decide exactly $P/poly$. This idea of "surpassing" the power of a Turing machine has attracted the attention of Jack Copeland ([6]). Martin Davis strongly critized this by recalling that these real weights are in fact oracles, as they were implemented into the network along with their non-computability. These oracles still have to be formalized. It seems that no one noticed that these oracles are related to the $P = NP$ problem, that is, the relativization problems.

Our goal is to define these oracles, we call them Davis's oracles in memory of his criticism, and show that Turing machines equipped with these oracles can be simulated by a neural network with real weights and conversely. It is known that $P = NP$ relativizes for the physical oracles [5]. We will regard them as a particular case of Davis's oracles and show that $P = NP$ relativizes also for this largar class, proving them to be a strict class of oracles.

2 ARNN

2.1 ARNN as a computational model

2.1.1 Synchronization

Many classical computational devices are known to be able to be build in blocks, such as Turing machines. We will show that we can combine neural networks in sequence and in parallel.

Definition 2.1. *A **subsystem** is a neural network where inputs and outputs allowed must not be binary.*

Note that a conventional neural network is a particular kind of subsystem but the converse is not always true.

The following lemma states that we can insert a switch unit that turns another unit off when its value is 1.

Lemma 2.2. *(Switch Lemma) Let y be a unit with dynamics*

$$y^+ = \sigma\left(\pi\left(\vec{x}\right)\right).$$

Then there exist a substitution of y by $\tilde{y}$ described as

$$\tilde{y}^+ = \sigma\left(\tilde{\pi}\left(\vec{x}, x_{switch}\right)\right)$$

such that if $x_{switch}(t) = 0$ then $\tilde{y}(t+1) = 0$ and if $x_{switch}(t) = 1$ then $\tilde{y}(t+1) = y(t+1)$.

As direct application, we can insert these switch units to control a sequence of subsystems or to combine them in parallel.

Proposition 2.3. *Let Σ_1 and Σ_2 be two subsystems. Then we can connect them to build a new subsystem working as follows. Σ_1 starts its computation while Σ_2 is in rest until the output of Σ_1 is fed into Σ_2. If needed, Σ_1 can be shut down after Σ_2 downloaded its output.*

Proposition 2.4. *Let Σ_1 and Σ_2 be two subsystems with one output unit and Σ with two input lines. Then we can build a subsystem that works as follows. Σ_1 and Σ_2 starts computing at $t = 0$ while Σ is in rest. The output of the first subsystem finishing the computation is saved until the second finishes its computation. When both finish their computations, the outputs are fed into Σ, starting its computation.*

2.1.2 Non-determinism

Definition 2.5. *A **non-deterministic analog recurrent neural net** (NDNN) $\mathcal{N}$ (of dimension m) consists on three input units of **validation of inputs**, **input** and a **guess unit**, receiving streams $v = 01^{|\alpha|}0^\omega$, $u = 0\alpha0^\omega$ and γ, a guess stream, with dynamics defined by*

$$\vec{x}(t+1) = \sigma(A\vec{x}(t) + \vec{b_1}v(t) + \vec{b_2}u(t) + \vec{b_3}\gamma(t) + \vec{c})$$

where $\vec{x}$ is the state vector of dimension m, A an $m \times m$ matrix, $\vec{b_i}$ and $\vec{c}$ vectors of dimension m.

*Two special units are chosen for the **output validation** and the **output**, sending out streams $z = 0^{T_{\mathcal{N}}(\alpha)-1}1^{|\tilde{\phi}(\alpha)|}0^\omega$, $y(t) = 0$ for $t < T_{\mathcal{N}}(\alpha)$ and $y(T_{\mathcal{N}}(\alpha) - 1 + i) = (\tilde{\phi}(\alpha))_i$, where $T_{\mathcal{N}}$ is the **computation time** given α as input and $\tilde{\phi} : \mathbb{N} \longrightarrow \mathbb{N}$ the function computed by $\mathcal{N}$. A word is in $dom(\tilde{\phi})$ if there is a guess stream such that its computation will lead to an output validation $z(t) = 1$ for some t.*

We are interested in a more restricted definition of NDNN, those that computes functions $\phi : \mathbb{N} \longrightarrow \mathbb{N}$, that is, given $\alpha \in \{0,1\}^+$, there is a γ such that $\Phi(\alpha, \gamma_1)$ is defined and, for all such γ this value should be the same.

The notion of acceptance of a language is similar to the one for non-deterministic Turing Machine:

Definition 2.6. *A language L is said to be* ***decided*** *by a non-deterministic net $\mathcal{N}$ if for all $\alpha \in L$, there is a guess γ such that $\Phi(\alpha, \gamma_1) = 1$, that is, $\mathcal{N}$ computes its characteristic function*[1]*.*

Proposition 2.7. *If ϕ is a funcion computed by a NDNN bounded in time by a polynomial, denoted $\phi \in ARNN^{ND}PF$, then $\phi \in NPF$.*

Once proven the other inclusion, we will have the equivalence

Theorem 2.8. $ARNN^{ND}PF = NPF$

The only tool we have is simulation of deterministic Turing machines. The following definition and proposition provide an alternative definition of NPF, which consists on separating a general non-deterministic Turing machine in a guess part attached to a deterministic Turing machine.

Definition 2.9. *The class $\exists PF$ consists on functions $\phi : \mathbb{N}^n \longrightarrow \mathbb{N}$ such that there exist a function $\Phi : \mathbb{N}^{n+1} \longrightarrow \mathbb{N}$ in PF and a polynomial p with the properties:*

(a) $\langle x_1, \ldots, x_n \rangle \in dom(\phi)$ if and only if there exists k such that $|k| \leq p\left(\sum_{i=1}^{n} |x_i|\right)$ and $\langle x_1, \ldots, x_n, k \rangle \in dom(\Phi)$;

(b) $\phi(x_1, \ldots, x_n)$ is defined and its value is y if and only if there exists a k such that $|k| \leq p\left(\sum_{i=1}^{n} |x_i|\right)$, $\langle x_1, \ldots, x_n, k \rangle \in dom(\Phi)$, $\Phi(x_1, \ldots, x_n, k)$ is defined and, for all k in these conditions, $\Phi(x_1, \ldots, x_n, k) = y$.

Proposition 2.10. $NPF = \exists PF$

We prove that $ARNN^{NP}[\mathbb{Q}] = \exists PF$.

Hence, the class of languages accepted by these nets is exactly NP. Joining this with the fact that deterministic neural nets with polynomial time bound decides exactly P, one is induced naturally to the positive relativization of the Hypothesis $P = NP$

3 Relativization

3.1 Davis's Oracles

In the proof of the simulation of neural networks with real weights by families of circuits, Siegelmann and Sontag provided one important result:

Proposition 3.1. *The output of an ARNN after t steps is only affected by the first $O(t)$ digits in the expansion of the weights and the states in each computation time. More preciscely, $Trunc_t = \left\lceil log\left(\frac{1}{8}(LW)^{(t-1)}\right)\right\rceil$ digits of precision will suffice, where L is one plus the dimension of the net and the number of input lines, W is one plus the largest absolute value of its weights.*

[1] The characteristic function of a set A is given by $\chi_A(\alpha) = 1$ if $\alpha \in A$, else $\chi_A(\alpha) = \perp$ (undefined).

Hence, given an ARNN with real weights, we can simulate it by a Turing machine M, a combination of M_1 and M_2, where M_1 extracts $Trunc_t$ bits of the real weights and M_2 a Turing machine that simulates a rational ARNN with the truncated weights. M_1 is in fact a Turing machine equipped with an oracle as we will see.

Definition 3.2. *A **Davis's oracle** O_r is a subset of $Prefix(r)$ for some $r \in \Gamma^\omega$, where Γ is the alphabet used by the working tapes.*[2] *The first n digits of r will be denoted $r_{|n}$.*

We will also call Davis's oracle sets of the kind

$$O_r^f = \{\langle 0^n, \tilde{r}\rangle : n \in \mathbb{N}, \tilde{r} \text{ is a prefix of } r_{|f(n)}\}$$

for some $f : \mathbb{N} \longrightarrow \mathbb{N}$ time constructible. These two definitions are equivalent with a possible delay in time f when implemented in a Turing machine.

Theorem 3.3. *(a) A system Σ with weights in $\mathbb{Q} \cup \{r\}$, $r \in [0,1]$, working in time t can be simulated by a Turing machine with a Davis's oracle O_r in time $O(p(t))$, p polynomial; (b) A Turing machine M with a Davis's oracle O_r [O_r^f] working in time t can be simulated by a neural net Σ weights in $\mathbb{Q} \cup \{r\}$, $r \in [0,1]$ working in time $O(p(t))$ [$O(f + p(t))$].*

By induction, we can conclude

Corollary 3.4. $ARNN^P[\mathbb{Q}](r_1)(r_2)\dots(r_n) = P(O_{r_1})(O_{r_2})\dots(O_{r_n})$.

Theorem 3.5. *These classes of sets coincides:*

1. $P/poly$
2. $\bigcup_{S sparse} P(S)$
3. $\bigcup_{T tally} P(T)$
4. $\bigcup_{r\in[0,1]} ARNN^P[\mathbb{Q}](r)$
5. $ARNN^P[\mathbb{R}]$
6. $\bigcup_{O Davis's\ Oracle} P(O)$

This is also true for the analogous non-deterministic classes.

3.2 Davis's Oracles vs. Physical Oracles

The working time of a standard oracle is constant by definition. This means that when it is called, no matter how long a query word is, the time taken to answer the query is the same, which is unrealistic when a physical experiment is incorporated as an oracle. For that we implement a clock into an oracle. We will be dealing with Davis's oracles with polynomial access time and exponential access time.

[2] r can be regarded as a real number in $[0,1]$.

Definition 3.6. *A function f is* ***s-approximable in time*** *T if there is a Turing machine that computes $f_{|s(n)}$ in time $T(n)$ given input of total size n. When the function s is not explicitly mentioned, we are considering $s(n) = n$.*

Definition 3.7. *A* ***generalized Davis's oracle*** *is a Davis's oracle $O_{f,r}$, a subset of $Prefix(f(r))$, where $r \in [0,1]$ and f is a function approximable in time T. The time protocol of this oracle is T.*

They can be seen as a blackbox with the following two components:

- A Davis's oracle O_r;
- A Turing machine M_f that approximates f in time T.

This oracle can be easily implemented in an ARNN, done as follows:

- A BAM subsystem to extract the bits of r;
- A subsystem simulating M_f in real time.

It takes $O(n + T(n))$ to compute the first n bits of r. Languages decided by these nets will be denoted by $ARNN[\mathbb{Q}](f, r)$.

Clearly they do not fit in the definition of a standard oracle. As stated before, an internal clock is *naturally* implemented into them, so we can control the time they take to answer a given query. By choosing a function f approximable in polynomial time and exponential time, we obtain respectively a *Davis's oracle with polynomial* and *exponential access time* and also their simulations in ARNN.

In [5], the scatter machine experiments and the collider machine experiments (abbreviated $SME(r)$ and $CME(r)$) has been studied by the authors, finding them to be oracles where the $P = NP$ problem relativizes. Both consists in extracting the bits of a real number r through the bisection method in polynomial time in $SME(r)$ and exponential time in $CME(r)$.

They are in fact particular cases of Davis's oracles with polynomial and exponential time access. As in the last referred article relativization results have been proven for those physical oracles, we will extend these proofs to Davis's oracles.

3.3 Relativization results

3.3.1 Polynomial access time

Proposition 3.8. *Let A be a set decided by a non-deterministic Turing machine M clocked in polynomial time p, equipped with a Davis's oracle O of polynomial time access. Then, there is a non-deterministic Turing machine $\tilde{M}$ working in polynomial time querying the same oracle a polynomial number of times during a computation. In fact, $|Q(\tilde{M}, z, O)| \leq p(|z|) + 1$.*

From this we can prove that

Theorem 3.9. *(Positive relativization) $P = NP$ if and only if, for any Davis's oracle O with polynomial access time,*

$$P(O) = NP(O).$$

By real time simulation of Turing machines with Davis's oracles by ARNN with one real weight, we have

Theorem 3.10. *$P = NP$ if and only if, for any $r \in [0, 1]$,*

$$ARNN^P[\mathbb{Q}](r) = ARNN^{NP}[\mathbb{Q}](r).$$

3.3.2 Exponential access time

Proposition 3.11. *Let A be a set decided by a non-deterministic Turing machine M clocked in polynomial time with an exponential Davis's oracle implemented in it. Then, there is a non-deterministic Turing machine $\tilde{M}$ also working in polynomial time querying the same oracle only a logarithmic number of times, where the total number of query words in all branches is logarithmic.*

Denote by $P_\ell(O)$ the class of languages decided by Turing machines working in polynomial time equipped with an oracle such that, for inputs of size n, only a logarithmic number on n of queries are admitted to be made. The last proposition showed that

Corollary 3.12. *For all Davis's oracles O_r, $P(O_r) = P_\ell(O_r)$.*

In particular,

Corollary 3.13. *For all Davis's oracles O_r and f approximable in exponential time, $P(O_{f,r}) = P_\ell(O_{f,r})$.*

Definition 3.14. *We denote by $\mathcal{F}*$ the set of prefix functions with size limited by a funtion in $\mathcal{F}$. That is, if $f \in \mathcal{F}*$, then for all n, $f(n)$ is a prefix of $f(n+1)$ and $|f(n)| \leq g(n)$ for some function $g \in \mathcal{F}$.*

Lemma 3.15. *For all Davis's oracle O with exponential access time, $P(O) \subseteq P/log* \subseteq P/log$.*

For this class of oracles we have also the negative relativization result, which is still an open problem for the polynomial ones.

Theorem 3.16. *(Positive relativization) $P = NP$ if and only if, for all Davis's oracles O with exponential time access,*

$$P(O) = NP(O).$$

Theorem 3.17. *(Negative relativization) $P \neq NP$ if and only if, for all Davis's oracles O with exponential time access,*

$$P(O) \neq NP(O).$$

The demonstration of the positive relativization result is similar to the case of polynomial access time. For the proof of the latter, we will need the following lemma well known in Structural Complexity theory [3]:

Lemma 3.18. *If $SAT \in P/log$, then $P = NP$.*

As particular cases of these results, we have:

Theorem 3.19. *$P = NP$ if and only if, for all $r \in [0,1]$,*

$$P(CME_e(r)) = NP(CME_e(r)).$$

Theorem 3.20. *$P \neq NP$ if and only if, for all $r \in [0,1]$,*

$$P(CME_e(r)) \neq NP(CME_e(r)).$$

Theorem 3.21. *$P = NP$ if and only if, for all $r \in [0,1]$ and f approximable in exponential time,*

$$P(O_{f,r}) = NP(O_{f,r}).$$

Theorem 3.22. *$P \neq NP$ if and only if, for all $r \in [0,1]$ and f approximable in exponential time,*

$$P(O_{f,r}) \neq NP(O_{f,r}).$$

Transposing these results to neural nets,

Theorem 3.23. *$P = NP$ if and only if, for all $r \in [0,1]$ and f approximable in exponential time,*

$$ARNN^P[\mathbb{Q}](f,r) = ARNN^{NP}[\mathbb{Q}](f,r).$$

Theorem 3.24. *$P \neq NP$ if and only if, for all $r \in [0,1]$ and f approximable in exponential time,*

$$ARNN^P[\mathbb{Q}](f,r) \neq ARNN^{NP}[\mathbb{Q}](f,r).$$

4 Conclusion

We provided a formalization of the idea of a real weight internalized in a rational ARNN regarded as a Davis's oracle, concretizing the criticism of Martin Davis in a constructive point of view. The simulation of a rational neural net with one real weight by a Turing machine with a Davis's oracle defined by that real weight shows that, not only the whole class of $ARNN^P[\mathbb{R}]$ coincide with $P/poly$, but there is also an inductive correspondence between the subclasses by incorporating the real weights one by one. This creates a hierarchy in $P/poly$, where each class can be represented by a finite set of real weights used, that is, sets of the form

$$\{r_1, r_2, \ldots, r_n : r_i \in \mathbb{R}, \forall 1 \leq i \leq n\}.$$

Taking into account time protocols of oracles, we proved that SME and CME are particular cases of Davis's oracles with polynomial and exponential access time, respectively. Furthermore, the famous problem $P = NP$ positively relativizes for Davis's oracles with both access times. We immediately conclude that Davis's oracles constitute a restricted class of oracles. In respect to real neural nets working in polynomial time, we conclude that they are in fact a restricted computational model. In resume, we have the following classification table:

Set of weights	Time restriction	Protocol	Computational power
$\mathbb{Z}$	none	none	Regular languages
$\mathbb{Q}$	none	none	Recursive languages
$\mathbb{Q}$	t	none	DTIME(t)
$\mathbb{Q} \cup \{r_1\}$	polynomial	exponential	$P_\ell(O_{r_1})$
$\vdots$			$\vdots$
$\mathbb{Q} \cup \{r_1, \ldots, r_n\}$	polynomial	exponential	$P_\ell(O_{r_1}) \ldots (O_{r_n})$
$\vdots$			$\vdots$
$\mathbb{R}$	polynomial	exponential	$P/log*$
$\mathbb{Q} \cup \{r_1\}$	polynomial	polynomial	$P(O_{r_1})$
$\vdots$			$\vdots$
$\mathbb{Q} \cup \{r_1, \ldots, r_n\}$	polynomial	polynomial	$P(O_{r_1}) \ldots (O_{r_n})$
$\vdots$			$\vdots$
$\mathbb{R}$	polynomial	polynomial	$P/poly$
$\mathbb{R}$	none	none	All languages

Table 1: A more refined classification by computational power of ARNN.

For the case of exponential access time, we also have the relativization of $P \neq NP$. This result is still an open problem for the polynomial case. Once proven (or disproved) for SME, its proof might be adapted to the case of Davis's oracles of polynomial access time and, in particular, real ARNN working in polynomial time.

Returning to more "grounded" issues, there are still simple questions that seem not simple to be answered. For instance, a very rewarding work would be a good classification of functional properties of neural nets. Within the logic system described by McCulloch and Pitts in [11], it is possible to realize a given logical function. But giving a description of *how* one neural network performs its computation is not trivial. Most of the known work refers to layered neural nets (for example, BAM).

Computational approach did not show to be the best single way to understand our brain and to explain how it works. But it turned out to be a relatively good tool to functionally map our brain and, at the same time, provided a very

interesting biologically inspired computational model that uniformly classifies complexity classes in a surprising way.

References

[1] James A. Anderson. *An Introduction to Neural Networks*. Massachusetts Institute of Technology, 1995.

[2] José Luis Balcázar, Josep Días, and Joaquim Gabarró. *Structural Complexity I*. Springer-Verlag, 2nd edition, 1988, 1995.

[3] José Luis Balcázar, Josep Días, and Joaquim Gabarró. *Structural Complexity II*. Springer-Verlag, 1990.

[4] Edwin Beggs, José Félix Costa, Bruno Loff, and John V. Tucker. Oracles and advice as measurements. In Cristian S. Calude, José Félix Costa, Rudolf Freund, Marion Oswald, and Grzegorz Rozenberg, editors, *Unconventional Computation (UC 2008)*, volume 5204 of *Lecture Notes in Computer Science*, pages 33–50. Springer-Verlag, 2008.

[5] Edwin Beggs, José Félix Costa, and John V. Tucker. Comparing complexity classes relative to physical oracles. 2009. Technical Report.

[6] Jack Copeland and Diane Proudfoot. Alan Turing's forgotten ideas in Computer Science. *Scientific American*, 280:99–103, 1999.

[7] Martin Davis. The myth of hypercomputation. In Christof Teuscher, editor, *Alan Turing: the life and legacy of a great thinker*, pages 195–212. Springer, 2006.

[8] Robin Gandy. The confluence of ideas in 1936. In Rolf Herken, editor, *The universal Turing machine: a half-century survey*, pages 55–111. Oxford University Press, 1988.

[9] S. Haykin. *Neural Networks: A Comprehensive Foundation*. MacMillan College Publishing, 1994.

[10] Hava T. Siegelmann João Pedro Neto and José Félix Costa. Symbolic Processing in Neural Networks. *Journal of the Brazilian Computer Society*, 8(3):58–70, 2003.

[11] Warren McCulloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Journal of Mathematical Analysis and Applications*, 5:115–133, 1943.

[12] M. L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Engelwood Cliffs, NJ, 1967.

[13] James Nyce. Nature's machine: mimesis, the analog computer and the rhetoric of technology. In Ray Paton, editor, *Computing with Biological Metaphors*. Chapman & Hall, 1994.

[14] Mark Schlatter and Ken Aizawa. Walter Pitts and "A Logical Calculus". *Synthese*, 162:235–250, 2008.

[15] Hava T. Siegelmann and Eduardo D. Sontag. On the computational power of neural networks. pages 331–360, 1992.

[16] Hava T. Siegelmann and Eduardo D. Sontag. Analog computation via neural networks. *Theoretical Computer Science*, 131(2):331–360, 1994.

[17] Hava T. Siegelmann and Eduardo D. Sontag. On the computational power of neural networks. *J. Comp. Syst. Sciences*, 50(1):132–150, 1995.

[18] Eduardo D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Springer-Verlag, 1990.