Neural Networks as Pseudorandom Number Generators

Mark Goldsmith

A Thesis
in The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy of Computer Science at
Concordia University
Montreal, Quebec, Canada

April 2015

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: **Mark Goldsmith**

Entitled: **Neural Networks as Pseudorandom Number Generators**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Jamal Bentahar	Chair
Dr. Pierre L'Ecuyer	External Examiner
Dr. Pawel Góra	External to Program
Dr. Peter Grogono	Examiner
Dr. Sudhir Mudur	Examiner
Dr. Vašek Chvátal	Thesis Supervisor

Approved by

Chair of Department or Graduate Program Director

April, 2015

Dean of Faculty

Abstract

Neural Networks as Pseudorandom Number Generators
Mark Goldsmith

Concordia University, 2015

This thesis brings two disparate fields of research together; the fields of artificial neural networks and pseudorandom number generation. In it, we answer variations on the following question: can recurrent neural networks generate pseudorandom numbers? In doing so, we provide a new construction of an n-dimensional neural network that has period 2^n, for all n. We also provide a technique for constructing neural networks based on the theory of shift register sequences. The randomness capabilities of these networks is then measured via the theoretical notion of computational indistinguishability and a battery of statistical tests. In particular, we show that neural networks cannot be pseudorandom number generators according to the theoretical definition of computational indistinguishability. We contrast this result by providing some neural networks that pass all of the tests in the SmallCrush battery of tests in the TestU01 testing suite.

Acknowledgments

I would like to start off with a double thank you to my advisor, Vašek Chvátal. It has been an absolute delight to work (and play) under his supervision. His clear, concise, and rigorous mathematical expositions have been and continue to be an inspiration. His good tastes have been contagious over the years, and I am indebted to him for making me more cultured and civilized. His comments and suggestions have been invaluable throughout the writing of this thesis.

Thank you also to my wife, Masha. She initially pushed me in Vašek's direction, and has thus earned the eternal right to gloat. Getting this far would not have been possible without her continual support and advice.

I would also like to thank the many members and visitors to ConCoCo for much lively discussion. In particular, I would like to thank Luc Devroye for his sharp observations on any topic I presented him with; Nithum Thain, whose initial seed has given rise to the state of this thesis; Markéta Vyskočilová, for her enlightening instruction on how neurologists study EEGs; Nan Yang for his thought provoking collaborations; and Yori Zwols for his comments on how else to attack some of the problems I faced. Visits by Peter Gacs and Avi Wigderson were also eye opening and inspiring, and were essential to building my randomness chops. I am grateful as well to Pierre L'Ecuyer for his comments on $\mathbf{F}_2$-generators, which opened up a whole new line of attack for me.

Finally, I thank my parents, my brother, and everyone in my family who I know are absolutely gushing at the thought of this thesis being completed.

Contents

Chapter 1

Introduction

This thesis brings two disparate fields of research together; the fields of artificial neural networks and pseudorandom number generation. In it, we answer variations on the following question: can recurrent neural networks generate pseudorandom numbers? We will see that the answer to this question depends on how we measure randomness, and what kind of restrictions are placed on the neural networks in question.

The artificial neurons that we study are threshold functions based on the model of McCulloch and Pitts [55]. Although these models should not be confused with truly biological neurons, the work in this thesis is loosely inspired by ongoing research on EEG (electroencephalogram) analysis and the study of epileptic seizures.

Motivation: EEG and Chaos

In attempts to study epilepsy, selected patients are monitored continuously for days at a time. During these periods, EEG or ECoG (electrocorticogram) recordings are made. EEG recordings come from placing multiple electrodes on the scalp of the patient. ECoG recordings produce far more accurate data, but they require invasive surgery to place a grid of electrodes directly on the cortex.

Neurologists specialized in epilepsy are trained to read EEG/ECoG recordings, so that mere visual inspection allows them to tell with a reasonable degree of accuracy when a seizure might have occurred. A frequent occurrence is a transition from a 'chaotic', 'random-like' EEG before the seizure (the pre-ictal state) to a more organized sustained rhythm of spikes or sharp waves during the seizure (the ictal state). A few examples of such claims follow:

"Neonatal electrographic seizures consist of paroxysmal events characterized by the sudden appearance of recognizable, repetitive wave forms that evolve in amplitude and frequency, and then wane. The location of the seizure may be focal, confined to one hemisphere, or may spread to the opposite hemisphere. In contrast, the EEG background of the normal newborn is irregular and without clear periodicity." [46]

"The results revealed that epileptic EEG had significant nonlinearity whereas normal EEG behaved similar to Gaussian linear stochastic process." [60]

"To illustrate this concept, we may assume, for simplicity, that at least two states are possible: an interictal one characterized by a normal, apparently random, steady-state of ongoing activity, and another one that is characterized by the paroxysmal occurrence of asynchronous oscillations (seizure)." [47]

"This behavior (of STL_{max}) indicates a gradual preictal reduction in chaoticity, reaching a minimum shortly after seizure onset." [32]

"Abrupt state transitions from highly complex, irregular to less complex, almost periodic dynamics appear to be a characteristic feature of many dynamical disorders including epilepsy..." [43]

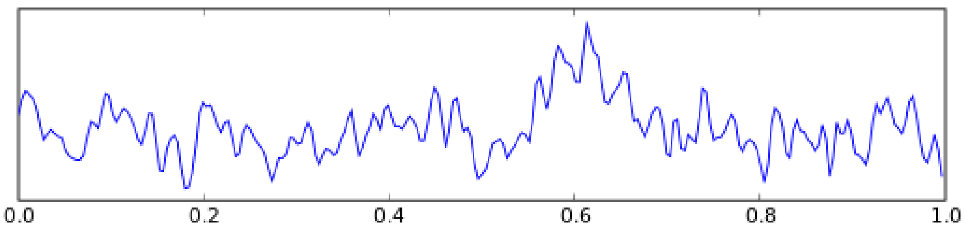


Figure 1.1: A one second sample of a typical EEG reading of a healthy brain.

Of course, 'chaotic' and 'random-like' are two very distinct mathematical ideas. Chaos is a formal, well-defined mathematical property that a dynamical system may or may not possess ([13], [63]). A dynamical system is a *deterministic* mapping from some state space to itself. Chaotic dynamical systems must be sensitive to initial conditions (commonly known as the *butterfly effect*), but this sensitivity is not sufficient for a dynamical system to be chaotic.

A commonly used tool for measuring this sensitivity in a dynamical system is its spectrum of *Lyapunov exponents*; each Lyapunov exponent in the spectrum can be thought of as a measurement of the sensitivity to initial conditions at specific points in the state space of the system, along a certain direction. The work of Takens [70] showed how the dynamics of an underlying dynamical system can be estimated from a time series of data representing a partial trajectory on a state subspace. Wolf et al. [79] used this idea to develop an algorithm for approximating the spectrum of Lyapunov exponents of the underlying dynamical system. This eventually led to a flood of research claiming that Wolf et al.'s algorithm could be used for predicting epileptic seizures:

> "The mean values of L (largest average Lyapunov exponent) for all electrodes in the postictal state are larger than the ones in the preictal state, denoting a more chaotic state postictally." [31]

> "Nonlinear techniques showed that the trajectory of the EGG signals appeared to be more regular and organized before the clinical onset of the seizure than were the ones in the normal state. The results of this work indicate that the EEG becomes progressively less chaotic as seizures advance, with respect to the estimation of the short-term maximum Lyapunov exponents ($\mathrm{STL}_{\max}$), which is a measure of the order or disorder (chaos) of signals." [8] (in reference to [31])

> "The Lyapunov exponent (λ_1) is a measure of the average amount of predictability over time of the EEG signal, which we believe relates to what is characterized by the morphologic regularity ratings we have made manually. λ_1 is a measure used in non-linear dynamic analysis (Ott, 1993). While in theory, a positive λ_1 can be used to identify the existence of chaos (assuming deterministic, non-transient dynamics), our goal is not to determine whether chaos is present but to use λ_1 as a potentially clinical useful tool and to better define seizure therapeutic adequacy by quantifying the degree of morphologic regularity of the seizure." [37]

> "Among the important measures of the dynamics of a linear or nonlinear system are the Lyapunov exponents that measure the average information flow (bits/sec) a system produces along local eigendirections through its movement in its state space. Positive Lyapunov exponents denote generation of information while negative exponents denote destruction of information. A chaotic non-linear system possesses at least one positive Lyapunov exponent, and it is because of this feature that its behavior looks random, even though as a system it is deterministic." [8]

In contrast to chaotic dynamical systems, which map states deterministically over time, a random process evolves (nondeterministically) according to some probabilistic laws. The nondeterminism is the point worth emphasizing here; visiting a state that has already been seen does not necessarily allow us to predict anything about its successor.

When presented with some time series, it may be difficult to distinguish which kind of process is at the heart of the underlying system [76].

Consider the following two time series plots:

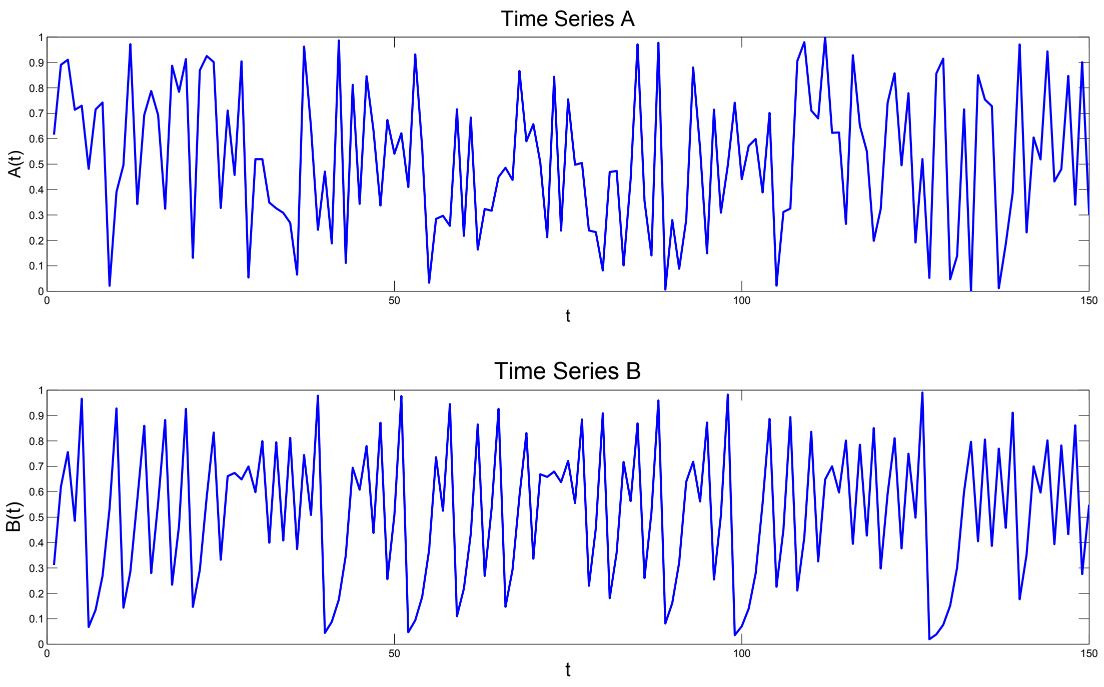
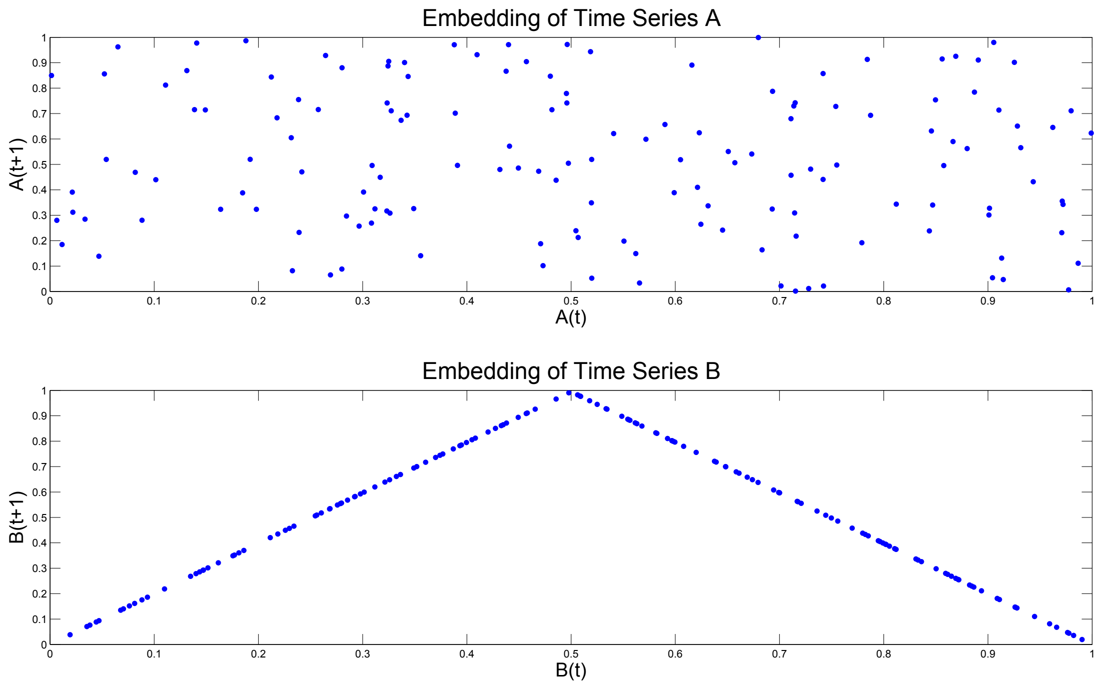


Figure 1.2: Time series A and B. Are they 'random'?

To the mathematically untrained eye, these may both look 'random' or 'chaotic'. How can we deduce whether or not there is any determinism at play? One method is to embed the time series into two dimensional space by plotting successive pairs $(x_0, x_1), (x_1, x_2), (x_2, x_3), \ldots$, where $x_0, x_1, x_2, \ldots$ is the original time series. In the cases above, the determinism of the second system is revealed:



The first time series in Figure 1.2 is ‘truly‘ random data coming from atmospheric noise [1], the latter was generated by the simple but chaotic Tent map, $T : [0, 1] \to [0, 1]$, defined by:

$$T(x) = \begin{cases} 2x & \text{if } x < 0.5; \\ 2(1-x) & \text{if } x \geq 0.5. \end{cases}$$

Things become much more complicated in the case of EEG data, in which nothing is known about the correct embedding dimension or time that should elapse between adjacent samples. However, the main idea remains the same: try to embed the provided time series data into some dimension and extract the determinism; a truly random source will look like noise in all dimensions. Other techniques for finding properties of the possibly deterministic underlying generator of time series data can be found in [68].

The notions of chaotic dynamical systems and random processes meet in the concept of *pseudorandomness*, in which deterministic systems have properties usually observed by a random process.

The question

In July 2009, in a seminar held at Concordia University, Nithum Thain asked whether some initial configuration could cause Conway’s Game of Life [17] to evolve in a way resembling a partial seizure, proceeding from an erratic flutter of apparently unpredictable patterns to sustained rhythmic changes that would begin in a small part of the grid and gradually spread, synchronized, over a larger area before subsiding to give way to the initial erratic mode. In the discussion that followed, a variation emerged: Could a recurrent neural network behave like this?

More concretely, in this thesis we aim to find recurrent neural networks with state trajectories that are *pseudorandom*. Such a network would, by definition, be deterministic, yet still have have random-looking output. We will use a battery of tests in the TestU01 suite ([41], [42]) for testing pseudorandom output. The main reason for this choice is the obvious one: we would like the output to look random, and this is precisely what these tests check for. Another reason for this choice is consistency; other measurements such as Wolf et. al's algorithm for estimating the spectrum of Lyapunov exponents depend on too many parameters that have to be tweaked on a case by case basis [21]. The tests in TestU01 are robust, highly standardized, widely used, and difficult to pass.

The problem of using neural networks to generate pseudorandom output was studied briefly by Elyada and Horn [15]. We will see several constructions of neural networks that improve on their results; namely *maximal* neural networks, which are n-dimensional neural networks whose periods are all of length 2^n, and *shift register neural networks*, which are neural networks based on *shift register sequences*, sometimes referred to as *linear feedback shift registers*. The problem was also discussed in Hughes' Bachelor's thesis [30], in which a relaxation of the classical recurrent neural network model was considered.

Thesis structure

The basic unit of computation in our neural networks, the neuron, is a linear threshold function, sometimes referred to as a *perceptron*. Understanding these functions is essential to understanding the capabilities and limitations of neural networks; in Chapter 2 we survey some useful results on threshold functions that are commonly found in the literature.

Chapter 3 explores neural networks as dynamical systems, with a particular emphasis on neural networks that have trajectories with long periods. The problem of finding long period neural networks can be thought of as the complement of a problem solved by Hopfield, who found conditions under which a neural network has fixed points [28].

Three constructions of *maximal* neural networks are presented; the first comes from Arimoto and Ueda [74], the second is by Orlitsky [67], and the third is new work by

Chvátal and Goldsmith [10]. We will see that the neural networks resulting from these constructions are isomorphic.

We end Chapter 3 by building neural networks based on the theory of shift register sequences. This will allow us to construct another class of neural networks that, although not maximal, still have long enough trajectories for the purpose of pseudorandom number generation.

In Chapter 4 we evaluate neural networks as pseudorandom number generators using two different measures of randomness. The first is that of *computational indistinguishability.* We present a proof by Chvátal, Goldsmith, and Yang, showing that neural networks without hidden neurons cannot be pseudorandom number generators, in a theoretical sense. We contrast this result by pitting some of the neural networks of Chapter 3 against a battery of empirical statistical tests in the *TestU01* package ([41], [42]). Finally, we show how we can modify certain neural networks in order to pass these tests, when some of the neural network is hidden.

The main new results presented in this thesis are: a new construction of maximal neural networks (section 3.1.3); a construction of neural networks based on shift register sequences (section 3.2.1); a proof that neural networks are not pseudorandom number generators in the context of computational indistinguishability (section 4.1.1); and a description of several neural networks that pass all of the statistical tests in the *SmallCrush* battery in TestU01 (section 4.2). Some other new results include a proof showing that the three known maximal neural networks are all isomorphic (section 3.1.4), and a new proof of the lower bound on cross-correlation in maximal neural networks (section 3.1.6).

Chapter 2

Boolean and Threshold Functions

In this chapter, we survey some well-known properties of threshold functions. Most of the results in this section can be found in [58]. Other sources are [74], [4], and [3].

The connection with neural networks, as we shall see at the end of this chapter, is that threshold functions are commonly used to model neurons in artificial neural networks.

2.1 Boolean functions

A boolean function is a function that maps binary strings to bits:

Definition 2.1:
An *n-dimensional boolean function* is a function $f : \{0,1\}^n \to \{0,1\}$.

Alternatively, it will sometimes be more convenient to use $\{-1,1\}$ as our set of symbols instead of $\{0,1\}$, in which case we have

Definition 2.2:
An *n-dimensional boolean function* is a function $f : \{-1,1\}^n \to \{-1,1\}$.

For now, let us consider boolean functions $f : \{0,1\}^n \to \{0,1\}$. We can think of a boolean function as a truth table, or as a labelling of the vertices of the n-dimensional hypercube. The points of $\{0,1\}^n$ that f maps to 1 are the *true points of f*, and the points mapped to 0 are the *false points of f*.

Example 2.1:
Consider the 3-dimensional boolean function $f : \{0,1\}^3 \to \{0,1\}$ defined by the following truth table:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

We can view this boolean function as the 3-dimensional hypercube, where the true points of f are colored black, and the false points colored white.

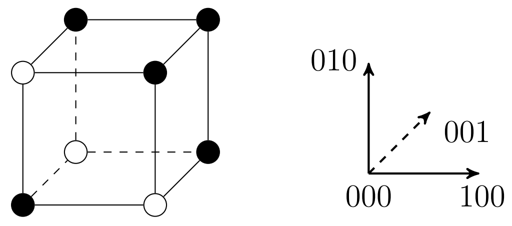


Figure 2.1: The boolean function f and coordinate system.

$\blacksquare$

Notation 2.1:
For $\mathbf{x}$ in $\{0,1\}^n$, we let $\overline{\mathbf{x}}$ be the result of flipping all the bits of $\mathbf{x}$. More precisely, if $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, then $\overline{\mathbf{x}} = (1 - x_1, 1 - x_2, \ldots, 1 - x_n)$.

For an n-dimensional boolean function f, we use $\overline{f}(x_1, \ldots, x_n)$ instead of the more cumbersome

$$\overline{f(x_1, \ldots, x_n)}$$

to denote the result of applying f and then flipping its output. We refer to $\overline{f}$ as the *complement* of f.

Another function of interest first flips its input, and then applies f. We denote this new function by $\underline{f}$, so that

$$\underline{f}(x_1, \ldots, x_n) = f(\overline{x_1}, \ldots, \overline{x_n}).$$

We use $\vee$ to denote logical *disjunction.* The *conjunction* of two boolean values x_1 and x_2 is denoted by x_1x_2; this is equivalent to multiplication with the boolean values viewed as integers and will help minimize the use of brackets to indicate precedence. Finally, the *xor* of x_1 and x_2 is denoted by $x_1 \oplus x_2$. It is defined such that $x_1 \oplus x_2 = (x_1\overline{x_2}) \vee (\overline{x_1}x_2)$.

2.2 Threshold functions

Threshold functions are a particular class of boolean functions.

Definition 2.3:
An n-dimensional boolean function $f : \{0,1\}^n \to \{0,1\}$ is a *linear threshold function* if there exist n real numbers $w_1, w_2, \ldots, w_n$ and another real number θ such that f can be expressed as

$$f(x_1, \ldots, x_n) = \begin{cases} 1 \text{ if } \sum x_i w_i \geq \theta \\ 0 \text{ if } \sum x_i w_i < \theta. \end{cases}$$

The values $w_1, w_2, \ldots, w_n$ are the *weights* and θ is the *threshold* of f.

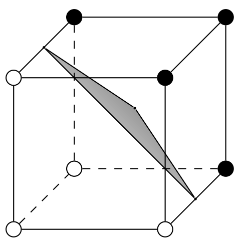
This definition may be generalized to threshold functions that are not necessarily linear. However, since we will only be considering *linear* threshold functions, we will simply refer to them as *threshold functions.*

Definition 2.4:
A threshold function f with weights $w_1, w_2, \ldots, w_n$ and threshold θ is *non-degenerate* if for every $(x_1, x_2, \ldots, x_n)$ in $\{0,1\}^n$ we have $\sum w_i x_i \neq \theta$.

Any threshold function can be put into an equivalent non-degenerate form by a slight perturbation of its threshold value. This form will often make our proofs simpler.

The equation $\sum x_i w_i = \theta$ can be thought of as an $(n-1)$-dimensional hyperplane in n-space. Threshold functions can thus be identified with hyperplanes that separate the hypercube into two regions. For this reason, threshold functions are sometimes referred to as *linearly separable functions* in the literature.

Example 2.2:
A 3-dimensional threshold function with hyperplane separating the true and false points.



■

Lemma 2.1:
If f is a threshold function, then so are $\overline{f}$ and $\underline{f}$. Furthermore, $\overline{f}$ and $\underline{f}$ can be realized using the same weights (but may require different threshold values).

Proof:
Without loss of generality, f is non-degenerate with weights $w_1, w_2, \ldots, w_n$ and threshold θ. We flip the sign of these weights to give us new weights $-w_1, -w_2, \ldots, -w_n$. Then $\overline{f}$ can be realized as a threshold function with weights $-w_1, -w_2, \ldots, -w_n$ and threshold $-\theta$. Furthermore, $\underline{f}$ can be realized as a threshold function with weights $-w_1, -w_2, \ldots, -w_n$ and threshold $\theta - \sum w_i$. □

Our next result comes from [58] (page 217).

Theorem 2.1:
Let f_1 and f_2 be $(n-1)$-dimensional threshold functions that can be realized with the same weights $w_1, \ldots, w_{n-1}$. Then the n-dimensional boolean function f, defined as

$$f(x_1, \ldots, x_n) = \overline{x_n} f_1(x_1, \ldots, x_{n-1}) \vee x_n f_2(x_1, \ldots, x_{n-1}) \tag{2.1}$$

is threshold.

Proof:
Without loss of generality, f_1 and f_2 are non-degenerate, with threshold values θ_1 and θ_2, respectively. Consider the n-dimensional threshold function g with weights

$w_1, w_2, \ldots, w_{n-1}, \theta_2 - \theta_1$ and threshold value θ_2. We will show that g is identical to f defined in (2.1).

When the last input is 0, we have

$$g(x_1, \ldots, x_{n-1}, 0) = \begin{cases} 1 & \text{if} \quad \sum_{i=1}^{n-1} w_i x_i > \theta_2; \\ \\ 0 & \text{if} \quad \sum_{i=1}^{n-1} w_i x_i < \theta_2, \end{cases}$$

which implies that $g(x_1, \ldots, x_{n-1}, 0) = f_1(x_1, \ldots, x_{n-1})$ for every $(x_1, \ldots, x_{n-1})$ in $\{0,1\}^{n-1}$. By equation (2.1), we have $f(x_1, \ldots, x_{n-1}, 0) = f_1(x_1, \ldots, x_{n-1})$. We conclude that $f(x_1, \ldots, x_{n-1}, 0) = g(x_1, \ldots, x_{n-1}, 0)$ for every $(x_1, \ldots, x_{n-1})$ in $\{0,1\}^{n-1}$.

Similarly, when the last input is 1, we have

$$g(x_1, \ldots, x_{n-1}, 1) = \begin{cases} 1 & \text{if} \quad \sum_{i=1}^{n-1} w_i x_i + \theta_2 - \theta_1 > \theta_2; \\ \\ 0 & \text{if} \quad \sum_{i=1}^{n-1} w_i x_i + \theta_2 - \theta_1 < \theta_2, \end{cases}$$

which implies that $g(x_1, \ldots, x_{n-1}, 1) = f_2(x_1, \ldots, x_{n-1})$ for every $(x_1, \ldots, x_{n-1})$ in $\{0,1\}^{n-1}$. By equation (2.1), we have $f(x_1, \ldots, x_{n-1}, 1) = f_2(x_1, \ldots, x_{n-1})$. We conclude that $f(x_1, \ldots, x_{n-1}, 1) = g(x_1, \ldots, x_{n-1}, 1)$ for every $(x_1, \ldots, x_{n-1})$ in $\{0,1\}^{n-1}$.$\square$

We will use a special case of the following Corollary later in Section 2.2.3.

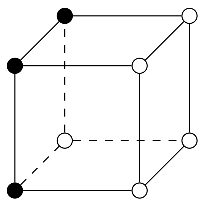
Corollary 2.1:
Let g be an $(n-1)$-dimensional threshold function. Then the n-dimensional function f, defined as

$$f(x_1, \ldots, x_n) = \overline{x_n}\underline{g}(x_1, \ldots, x_{n-1}) \vee x_n\overline{g}(x_1, \ldots, x_{n-1})$$

is threshold.

Proof:
By Lemma 2.1, $\underline{g}$ and $\overline{g}$ are both threshold can be realized with the same weights. The result follows by Theorem 2.1. $\square$

Example 2.3:
The 3-dimensional threshold function g given as

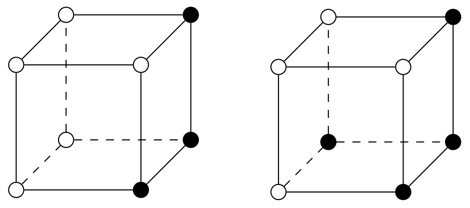


g

yields the 4-dimensional threshold function

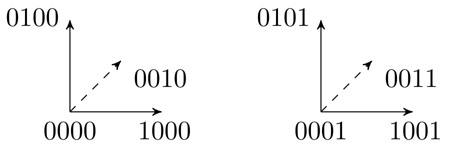
$$f(x_1, x_2, x_3x_4) = \overline{x_4}\underline{g}(x_1, x_2, x_3) \vee x_4\overline{g}(x_1, x_2, x_3),$$

which looks like



$\overline{x_4}\underline{g}(x_1, x_2, x_3)$ $x_4\overline{g}(x_1, x_2, x_3)$

Where the coordinate system is given as



■

2.2.1 Assumability

The notion of *assumability* provides us with a necessary and sufficient condition for a boolean function to be threshold.

Definition 2.5:
A boolean function $f : \{0,1\}^n \to \{0,1\}$ is *assummable* if for every k (not necessarily distinct) true points $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k$ in $\{0,1\}^n$ and k (not necessarily distinct) false points $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^k$ in $\{0,1\}^n$ we have

$$\sum_i^k \mathbf{x}^i \neq \sum_i^k \mathbf{y}^i. \tag{2.2}$$

The next theorem is well known and dates back to [9] and [14]. The proof given here is from [3].

Theorem 2.2:
A boolean function is threshold if and only if it is assumable.

Proof:
Suppose f is a non-degenerate threshold function with weights $w_1, w_2, \ldots, w_n$ and threshold θ. Let $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k$ and $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^k$ be arbitrary true and false points of f, respectively. Then we have

$$\sum_{i=1}^{n} w_i x_i^j > \theta > \sum_{i=1}^{n} w_i y_i^j \quad j = 1, 2 \ldots, k.$$

Summing over all j, we get

$$\sum_{j=1}^{k}\sum_{i=1}^{n} w_i x_i^j > k\theta > \sum_{j=1}^{k}\sum_{i=1}^{n} w_i y_i^j,$$

which gives

$$\sum_{i=1}^{n} w_i \sum_{j=1}^{k} x_i^j > k\theta > \sum_{i=1}^{n} w_i \sum_{j=1}^{k} y_i^j.$$

Therefore,

$$\sum_{i=1}^{n} w_i \left(\sum_{j=1}^{k} x_i^j - \sum_{j=1}^{k} y_i^j \right) > 0,$$

which allows us to conclude the inequality in (2.2).

Conversely, suppose f is not threshold. Let $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^r$ and $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^s$ be the true and false points of f, respectively. We construct an $rs \times n$ matrix A, where the rows of A are the vectors $\mathbf{x}^j - \mathbf{y}^k$, for $j = 1, 2, \ldots, r$ and $k = 1, 2, \ldots, s$. Suppose $A\mathbf{w} > \mathbf{0}$ has a solution. Then for $j = 1, 2, \ldots, r$ and $k = 1, 2, \ldots, s$ we would have

$$\sum_{i=1}^{n} w_i x_i^j > \sum_{i=1}^{n} w_i y_i^k.$$

This contradicts our assumption that f is not a threshold function. Therefore, $A\mathbf{w} > \mathbf{0}$ has no solutions. By Theorem A.1 of the Appendix (page 104), there exists a non-zero vector $\mathbf{z}$ in $\mathbf{R}^{rs}$ with nonnegative entries such that $A^T\mathbf{z} = \mathbf{0}$. Furthermore, since A consists of rational numbers (integers, in fact), we may assume that $\mathbf{z}$ has integer entries. Relabelling the elements of $\mathbf{z}$ according to the ordering of the vectors in A, for $j = 1, 2, \ldots, r$ and $k = 1, 2, \ldots, s$ there are nonnegative integers z_{jk}, at least one of which is nonzero, such that for $i = 1, 2, \ldots, n$ we have

$$\sum_{j,k} z_{jk}(x_i^j - y_i^k) = 0.$$

Thus, we have

$$\sum_{j,k} z_{jk}(\mathbf{x}^j - \mathbf{y}^k) = 0,$$

which shows that f is not assumable. □

Example 2.4:
The n-dimensional *xor* function $f : \{0,1\}^n \to \{0,1\}$, defined as

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} 0 & \text{if } \sum x_i \equiv 0 \pmod 2; \\ 1 & \text{if } \sum x_i \equiv 1 \pmod 2, \end{cases}$$

is not a threshold function for $n \geq 2$. To see why, let

$$\begin{aligned} \mathbf{x} &= (0,0,0,0,\ldots,0), \\ \mathbf{x}' &= (1,1,0,0,\ldots,0), \\ \mathbf{y} &= (1,0,0,0,\ldots,0), \\ \mathbf{y}' &= (0,1,0,0,\ldots,0). \end{aligned}$$

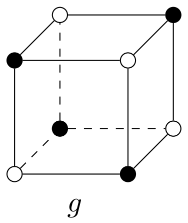


Figure 2.2: The 3-dimensional *xor* function, which cannot be linearly separated.

Then $f(\mathbf{x}) = f(\mathbf{x}') = 0$, and $f(\mathbf{y}) = f(\mathbf{y}') = 1$ and $\mathbf{x} + \mathbf{x}' = \mathbf{y} + \mathbf{y}'$. So f is not assumable and therefore not a threshold function. ■

The inability of a threshold function to compute the *xor* of its inputs was first illustrated by Minsky and Papert in 1969 [56], where they referred to threshold functions as *perceptrons*, a name given by Rosenblatt [64].

Somewhat ironically, the *xor* will play a special role for us; it is a very handy tool when it comes to randomness ([20], [45], [80], [19]) and we will go through some trouble to embed this function into the neural networks constructed in later chapters.

2.2.2 Number of threshold functions

Let $N(n)$ denote the number of threshold functions of dimension n. The numbers of threshold functions have been determined computationally (see [58] and [59]) for $n = 1, 2, \ldots, 9$, and are given here:

n	$N(n)$
1	4
2	14
3	104
4	1882
5	94572
6	15028134
7	8378070864
8	17561539552946
9	144130531453121108

Table 2.1: Number of threshold functions for $n = 1, \ldots, 9$.

Although a precise formula for $N(n)$ is not known, useful lower and upper bounds have been found. A lower bound derived in [58] is

$$N(n) > 2^{\frac{n(n-1)}{2}+32}.$$

We are more concerned with an upper bound, which we will now derive. What follows can be found in the work of Winder [78], Cover [12], and Muroga [58].

Definition 2.6:
Let Y be a subset of $\mathbf{R}^n$. A *dichotomy* of Y is its partition into two disjoint sets. A dichotomy (Y^+, Y^-) of a subset of $\mathbf{R}^n$ is *linearly separable* if there are numbers $x_1, x_2, \ldots, x_{n+1}$ such that

$$\begin{aligned} \sum_{j=1}^{n} x_j y_j > x_{n+1} \quad \text{whenever } (y_1, y_2, \ldots, y_n) \in Y^+, \\ \sum_{j=1}^{n} x_j y_j < x_{n+1} \quad \text{whenever } (y_1, y_2, \ldots, y_n) \in Y^-. \end{aligned} \tag{2.3}$$

Lemma 2.2:
A set of m points in $\mathbf{R}^n$ admits at most $2\sum_{i=0}^{n}\binom{m-1}{i}$ linearly separable dichotomies.

Proof:
Let $D(m,n)$ denote the maximum number of linearly separable dichotomies of a set of m points in $\mathbf{R}^n$ and let $R(m,n)$ denote the maximum number of open regions in $\mathbf{R}^n$ that can be demarcated by m hyperplanes passing through the origin. We claim that

(i) $D(m,n) \le R(m,n+1)$.

To justify this claim, consider any set Y of points $y^1, y^2, \ldots, y^m$ in $\mathbf{R}^n$, let D denote the set of linearly separable dichotomies of Y, and let R denote the set of open regions in $\mathbf{R}^{n+1}$ that are demarcated by the m hyperplanes

$$\textstyle\sum_{j=1}^{n} y_j^i x_j - x_{n+1} = 0 \quad (i = 1, 2, \ldots, m) \tag{2.4}$$

which pass through the origin. We will prove (i) by exhibiting a one-to-one mapping from D to R. (Actually, the mapping that we will exhibit is a one-to-one correspondence between D and R, which implies that (i) holds with the sign of equality; however, the inequality is all we need in proving the lemma.) If a linearly separable dichotomy (Y^+, Y^-) of Y satisfies (2.3), then $(x_1, x_2, \ldots, x_{n+1})$ belongs to one of the open regions that belong to R and this is the region that we assign to (Y^+, Y^-). If this region is also assigned to a linearly separable dichotomy (W^+, W^-) of Y defined by

$$\begin{aligned} \sum_{j=1}^{n} v_j y_j > v_{n+1} \quad & \text{whenever } (y_1, y_2, \ldots, y_n) \in W^+, \\ \sum_{j=1}^{n} v_j y_j < v_{n+1} \quad & \text{whenever } (y_1, y_2, \ldots, y_n) \in W^-, \end{aligned}$$

then

$$\begin{aligned} \sum_{j=1}^{n} y_j^i v_j - v_{n+1} > 0 \quad \text{if and only if} \quad \sum_{j=1}^{n} y_j^i x_j - x_{n+1} > 0, \\ \sum_{j=1}^{n} y_j^i v_j - v_{n+1} < 0 \quad \text{if and only if} \quad \sum_{j=1}^{n} y_j^i x_j - x_{n+1} < 0, \end{aligned}$$

and so $W^+ = Y^+$, $W^- = Y^-$.

Next, we claim that, for all choices of positive integers m and n, we have

(ii) $R(m,n) \le R(m-1,n) + R(m-1,n-1)$.

To justify this claim, consider any m pairwise distinct hyperplanes in $\mathbf{R}^n$ that pass through the origin; call one of these hyperplanes 'new' and call the other $m-1$

hyperplanes 'old'. Since all the old hyperplanes are distinct from the new hyperplane, each of them intersects the new hyperplane in a linear subspace of dimension $n-2$; these at most $m-1$ linear subspaces of dimension $n-2$ (at most $m-1$ since distinct old hyperplanes may intersect the new hyperplane in the same linear subspace) divide the new hyperplane into at most $R(m-1, n-1)$ regions. Since each of these regions in the new hyperplane is a boundary between two regions demarcated by the m hyperplanes, at most $R(m-1, n-1)$ regions of the at most $R(m-1, n)$ regions demarcated by the old hyperplanes are split by the new hyperplane into two.

Claim (ii) implies by induction on m that $R(m,n) \leq 2\sum_{i=0}^{n-1}\binom{m-1}{i}$. The Lemma follows from this inequality combined with (i). $\square$

Corollary 2.2:
The number of n-dimensional threshold functions, $N(n)$, satisfies

$$N(n) \leq 2\sum_{i=0}^{n}\binom{2^n-1}{i}.$$

Proof:
We apply Lemma 2.2 to the 2^n points of $\{0,1\}^n$. $\square$

Corollary 2.3:
The number of n-dimensional threshold functions, $N(n)$, satisfies

$$N(n) \leq 2^{n^2}.$$

Proof:
We follow the proof in [3]. To prove this upper bound, we first show that for $m \geq d$, we have

$$\sum_{i=0}^{d}\binom{m}{i} < \left(\frac{em}{d}\right)^d.$$

We have

$$\begin{aligned}
\sum_{i=0}^{d}\binom{m}{i} &< \left(\frac{m}{d}\right)^{d}\sum_{i=0}^{d}\binom{m}{i}\left(\frac{d}{m}\right)^{d}\\
&\le \left(\frac{m}{d}\right)^{d}\sum_{i=0}^{d}\binom{m}{i}\left(\frac{d}{m}\right)^{i}\\
&\le \left(\frac{m}{d}\right)^{d}\sum_{i=0}^{m}\binom{m}{i}\left(\frac{d}{m}\right)^{i}\\
&= \left(\frac{m}{d}\right)^{d}\left(1+\frac{d}{m}\right)^{m}\\
&< \left(\frac{em}{d}\right)^{d} \qquad \text{since } (1+x/m)^m < e^x \text{ for } x>0.
\end{aligned}$$

Therefore, setting $m = 2^n - 1$ and $d = n$, we get

$$2\sum_{i=0}^{n}\binom{2^n-1}{i} < 2\left(\frac{e(2^n-1)}{n}\right)^{n}.$$

When $n \geq 4$, we have $2\left(\frac{e(2^n-1)}{n}\right)^n < 2(e/4)^4 2^{n^2} < 2^{n^2}$, and the cases of $n = 1, 2, 3$ can easily be verified. $\square$

This Corollary gives us a sense of how few threshold functions there really are when we note that there are 2^{2^n} boolean functions of dimension n.

2.2.3 Self-duality

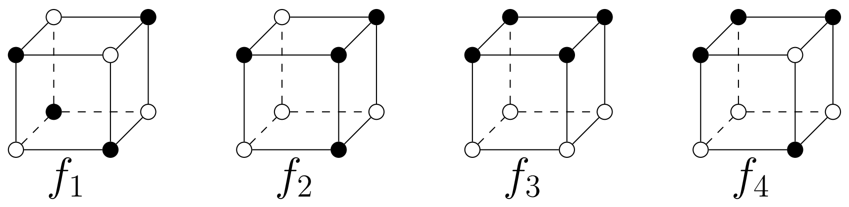
A special class of boolean functions are referred to in the literature as *self-dual*. They have the nice property that antipodes on the hypercube are mapped to complimentary bits.

Definition 2.7:
The *dual* of a boolean function f is the function $\overline{f(\overline{\mathbf{x}})}$, or simply the function $\underline{\overline{f}}$ (recall Notation 2.1 on page 9). Note that $\underline{\overline{f}}$ is well defined since both operations commute: $\overline{(\underline{f})} = \underline{(\overline{f})}$.

A boolean function f is *self-dual* if for every $\mathbf{x}$ in $\{0,1\}^n$ we have

$$f(\mathbf{x}) = \underline{\overline{f}}(\mathbf{x}).$$

Example 2.5:
Below are four 3-dimensional boolean functions:



The first boolean function is self-dual, but not threshold. The second and third, f_2 and f_3, are both self-dual and threshold. The last boolean function is neither self-dual nor threshold. ■

Theorem 2.3:
The n-dimensional threshold function f with weights $w_1, w_2, \ldots, w_n$ and threshold value $\theta = \sum w_i/2$ is self-dual if it is non-degenerate.

Proof:
Suppose f is non-degenerate and $(x_1, x_2, \ldots, x_n)$ in $\{0,1\}^n$ is such that

$$\sum_{i=1}^{n} w_i x_i > \sum_{i=1}^{n} w_i/2.$$

Then

$$\sum_{i=1}^{n} w_i(1-x_i) = -\sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} w_i < -\sum_{i=1}^{n} w_i/2 + \sum_{i=1}^{n} w_i = \sum_{i=1}^{n} w_i/2. \qquad \square$$

The following is from [73], and will be used in Section 3.1.1.

Lemma 2.3:
Let g be an $(n-1)$-dimensional boolean function. For $i = 1, 2, \ldots, n$, the function $u_i : \{0,1\}^n \to \{0,1\}$ defined by

$$u_i(x_1, \ldots, x_n) = x_i \overline{g}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_n) \vee \overline{x_i}\underline{g}(x_i, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \quad (2.5)$$

is a self-dual boolean function. Conversely, if u_i is a self-dual boolean function, then there exists an $(n-1)$-dimensional boolean function g that satisfies (2.5). Furthermore, u_i is a threshold function if and only if g is a threshold function.

Proof:
Fix i and write $G = g(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_n)$, so that

$$u_i(x_1, \ldots, x_n) = x_i \overline{G} \vee \overline{x_i}\underline{G}.$$

First, we will verify that u_i is self-dual. We have

$$\underline{u_i}(x_1, \ldots, x_n) = \overline{x_i}\overline{\underline{G}} \vee x_i G.$$

By using De Morgan's law we get

$$\begin{aligned}
\overline{u_i}(x_1, \ldots, x_n) &= [\overline{x_1} \vee G] \left[x_i \vee \overline{\underline{G}}\right] \\
&= \overline{x_i}\overline{\underline{G}} \vee x_i G \vee G\overline{\underline{G}} \\
&= \overline{x_i}\overline{\underline{G}} \vee x_i G, \qquad\qquad \text{(since } G\overline{\underline{G}} \text{ is redundant)}
\end{aligned}$$

which shows that $\underline{u_i} = \overline{u_i}$.

Next, we will show that every self-dual u_i can be decomposed as in (2.5). To begin, every boolean u_i can be written as

$$u_i(x_1, \ldots, x_n) = x_i\overline{g}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_n) \vee \overline{x_i}h(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_n)$$

for some boolean functions g and h. Write $G = g(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_n)$ as in the preceding paragraph and $H = h(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_n)$. Since

$$u_i(x_1, \ldots, x_n) = x_i\overline{G} \vee \overline{x_i}H,$$

we have

$$\underline{u_i}(x_1, \ldots, x_n) = \overline{x_i}\overline{\underline{G}} \vee x_i\underline{H},$$

and so

$$\begin{aligned}
\overline{\underline{u_i}}(x_1, \ldots, x_n) &= [x_i \vee \underline{G}] \left[\overline{x_i} \vee \overline{\underline{H}}\right] \\
&= x_i\overline{\underline{H}} \vee \overline{x_i}\underline{G} \vee \underline{G}\overline{\underline{H}} \\
&= x_i\overline{\underline{H}} \vee \overline{x_i}\underline{G}.
\end{aligned}$$

If u_i is self-dual, then $u_i = \overline{\underline{u_i}}$, and so $h(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_n) = \underline{g}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_n)$ as desired.

Moving on to the threshold part of the statement in the Theorem, suppose u_i is a non-degenerate n-dimensional self-dual threshold with weights $w_1, \ldots, w_n$ and threshold θ. We will show that the $(n-1)$-dimensional threshold function g with weights $-w_1, -w_2, \ldots, -w_{i-1}, -w_{i+1}, \ldots, -w_n$ and threshold $w_i - \theta$ satisfies (2.5). We have

$$\overline{g}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = \begin{cases} 0 & \text{if } \; -\sum_{j \neq i} w_j x_j > w_i - \theta \\ 1 & \text{if } \; -\sum_{j \neq i} w_j x_j < w_i - \theta. \end{cases}$$

Therefore, if $x_i = 1$, then

$$\overline{g}(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = \begin{cases} 0 & \text{if } \sum_{j=1}^n w_j x_j < \theta \\ 1 & \text{if } \sum_{j=1}^n w_j x_j > \theta, \end{cases}$$

which is the same thing as $u_i(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$.

If $x_i = 0$, we have

$$\begin{aligned} u_i(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) &= \overline{u_i(\overline{x_1}, \dots, \overline{x_{i-1}}, 1, \overline{x_{i+1}}, \dots, \overline{x_n})} \quad \text{by self-duality of } u_i \\ &= \overline{\overline{g}(\overline{x_1}, \dots, \overline{x_{i-1}}, \overline{x_{i+1}}, \dots, \overline{x_n})} \quad \text{by the } x_i = 1 \text{ case} \\ &= \underline{g}(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n), \end{aligned}$$

as desired.

Finally, if g is a threshold function then u_i is threshold as well by Corollary 2.1 on page 12. $\square$

2.2.4 Artificial neural networks

In 1943, Warren McCulloch and Walter Pitts proposed a simple model of a biological neuron and defined neural networks in their full generality [55]. Their all-or-none model, in which a neuron either fires completely or not at all (and no options in between), can be described as a threshold function in which all weights are set to +1 or some sufficiently large negative value, the latter type corresponding to their notion of *absolute inhibition*. Independently of this, in 1949 Donald Hebb's *The Organization of Behavior* [25] put forth his proposal for a theory of learning in the brain; the often-quoted 'neurons that fire together, wire together' has its origins therein:

> Let us assume that the persistence or repetition of a reverberatory activity (or 'trace') tends to induce lasting cellular changes that add to its stability. When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

In 1958, Frank Rosenblatt suggested a solution to the problem of how the McCulloch-Pitts model of a neuron could learn. In [64], he proposed ridding the neuron of its absolute inhibition and allowing weights to take on real numbers, in what he called a *perceptron* [64]. This is the model we use in this thesis.

We view neurons as threshold functions as follows. A neuron receives zero-one signals $x_1, x_2, \ldots, x_n$ at its synapses from the axons of other neurons (as well as from its own axon). Each synapse has an associated weight; positive weights correspond to excitatory synapses and negative weights correspond to inhibitory synapses. Hard-coded in the neuron is its threshold value; the neuron *fires* if $\sum w_i x_i \geq \theta$.

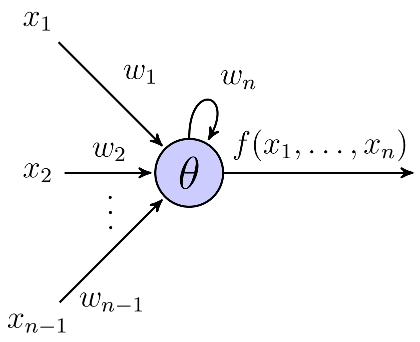


Figure 2.3: An artificial neuron, or *perceptron*, with inputs $x_1, x_2, \ldots, x_n$ and output $f(x_1, \ldots, x_n)$.

Along with this model, Rosenblatt proposed a learning algorithm, which allowed the perceptron to adapt its weights in order to *learn*; when presented a set of input-output pairs, the perceptron changes its weights so that when presented with an input, the corresponding output would be calculated. The details of this famous *perceptron learning algorithm* can be found in Rosenblatt's original paper [64]. Around the same time, Hoff and Woodrow devised their *Adaline* (adaptive linear) learning algorithm [77], which also uses a threshold function as the model of a neuron.

There was one important caveat to these algorithms, which was pointed out by Minsky and Papert in 1969. In order to learn from the input-output pairs, those pairs had to be *learnable*. As we have seen in example 2.4 on page 15, there are some very simple things that a threshold function cannot do. Minsky and Papert's book laid the foundations for the study of problems that can and cannot be computed by neural networks.

In 1986, a new wave of enthusiasm for neural networks was initiated with the publication of [66] by Rumelhart, Hinton, and Williams. They described a *back-propagation* algorithm for *feed-forward* neural networks, and demonstrated how an entire network could learn its weights when given a set of inputs and corresponding desired outputs. They proposed using sigmoid functions to model neurons, which had the advantage of being differentiable. This property was the key ingredient that allowed the back-propagation algorithm to work efficiently. The back-propagation algorithm for neural networks and its offshoots are still widely used in artificial intelligence and pattern recognition ([57] is a very recent and interesting example).

A *feed-forward* neural network consists of an *input* layer, one or more *hidden* layers, and an *output* layer. In this model, we can imagine the ticking of a discrete clock, with information being passed from one layer to the next at each time step.

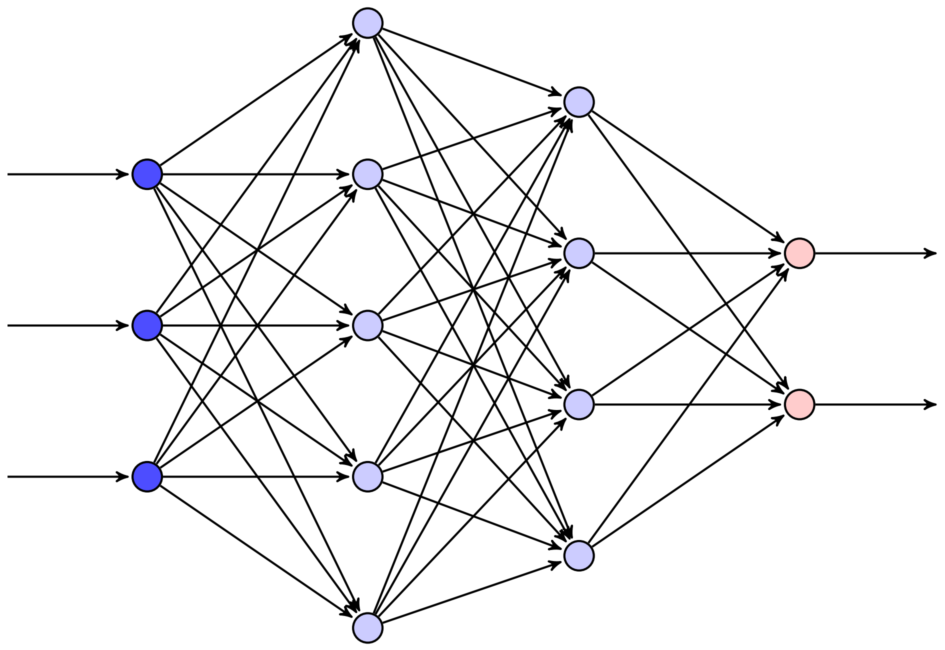


Figure 2.4: A feed-forward neural network of depth 3, consisting of 1 input layer (3 neurons), 2 hidden layers, and 1 output layer (2 neurons).

The *input* layer receives inputs from the external world. The output of the input layer is passed to the first hidden layer; the output of the first hidden layer is then passed to the second hidden layer, etc... until the last hidden layer's output is passed to the output layer, and the calculation is complete. Thus, the architecture of the network can be seen as a directed graph with no cycles.

Meanwhile, in 1982, John Hopfield removed the acyclic constraint to form an alternative model to the feed-forward network [28]. This model is now referred to as a *recurrent* neural network or a *Hopfield* network, although the latter usually implies that certain constraints are satisfied; namely, neurons have no self-connections, and the weight of a connection between two neurons is the same in both directions. Hopfield showed that a neural network will always settle to a fixed point under these conditions. This result is viewed as a possible explanation of how *autoassociative memory* works; the initial state of the neural network is some external pattern, and the stable fixed point of the network is the memory associated with that pattern.

Time could affect the network in Hopfield's model in one of three ways. In *parallel* mode, every neuron is updated at every time step; in *sequential* mode, a single neuron is updated at each time step; in other cases, the network updates in some *hybrid* mode. Hopfield later studied a continuous-time version as well [29]. Further modifications were made to this model by Hinton and Sejnowski a few years later, by allowing for a probabilistic update rule and the use of the sigmoid function, giving rise to the *Boltzman machine* [26] (named after the Boltzman distribution).

In the next chapter, we study recurrent neural networks operating in parallel mode, since they most closely resemble classical pseudorandom number generators that can be iterated over some state space.

Chapter 3

Recurrent Neural Networks and Dynamical Systems

This chapter is divided into two main sections, both of which emphasize the period size of neural networks. In particular, we wish to find neural networks that have long periods. In Section 3.1, we will discuss neural networks that have periods that are as long as possible. In Section 3.2, we describe neural networks that lack this property, but have long periods nonetheless.

Definition 3.1:
Let

$$f_i : \{0,1\}^n \to \{0,1\} \quad (i = 1, 2, \ldots, n)$$

be n threshold functions. The function $\Phi : \{0,1\}^n \to \{0,1\}^n$ defined by

$$\Phi(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x}))$$

is an *n-dimensional recurrent neural network*, or simply a *neural network*.

An n-dimensional neural network $(f_1, f_2, \ldots, f_n)$ may be described by an $n \times n$ weight matrix W and a threshold column vector T of size n. For i and j in $\{1, 2, \ldots, n\}$, let w_{ij} be the the jth weight of threshold function f_i, and let θ_i be the threshold value of

f_i. This network is given in matrix form as

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & & & \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{bmatrix}, \qquad T = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}.$$

Neural networks are dynamical systems; for a neural network Φ and an initial state $\mathbf{x}$, the *trajectory* of $\mathbf{x}$ under Φ is the sequence $\mathbf{x}, \Phi(\mathbf{x}), \Phi^2(\mathbf{x}), \ldots$. We refer to $\Phi^t(\mathbf{x})$ as *the state of* $\mathbf{x}$ *under* Φ *at time* t. Similarly, we let $\Phi_i^t(\mathbf{x})$ denote bit at coordinate i of $\mathbf{x}$ under Φ at time t, with the convention that the superscript t is read before the subscript i. Finally, the *successor* of $\mathbf{x}$ under Φ is $\Phi(\mathbf{x})$, and we sometimes denote this by $\mathbf{x} \to \Phi(\mathbf{x})$.

Example 3.1:
Consider the 3-dimensional neural network with weight matrix and threshold vector given by

$$W = \begin{bmatrix} 1 & 1 & 0 \\ 2 & -2 & 0 \\ 1 & -1 & -2 \end{bmatrix}, \qquad \theta = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}.$$

This network has the following state transitions:

$$\begin{tikzcd}
101 \arrow[dr] & & 001 \arrow[dl] & & \\
& 010 \arrow[d] & 100 \arrow[d] & 111 \arrow[r, bend right] & 110 \arrow[l, bend right] \\
& 000 \arrow[r, bend right] & 011 \arrow[l, bend right] & &
\end{tikzcd}$$

The state transition structure has two connected components. ■

3.1 Maximal period networks

In Chapter 4, our goal will be to describe some neural networks that have random-looking trajectories. Such networks must have trajectories with a long period, otherwise lack of randomness may easily be detected via periodic repetitions.

Definition 3.2:
The *period* of a neural network Φ with initial state $\mathbf{x}$ is the smallest positive integer p such that there exists a nonnegative integer t_0 for which

$$\Phi^{t+p}(\mathbf{x}) = \Phi^t(\mathbf{x}) \quad \text{for} \quad t = t_0, t_0+1, t_0+2, \ldots.$$

Note that different initial states of the same network may have different periods. For an n-dimensional neural network with an initial state, an upper bound on its period is 2^n, since the underlying state space contains 2^n unique states, and state transitions occur deterministically. Furthermore, if an n-dimensional neural network has a period of 2^n for some initial state, then it has a period of 2^n for every initial state.

Definition 3.3:
The period of an n-dimensional neural network is *maximal* if it is 2^n for an initial state (and therefore every initial state).

We are now ready to ask: do neural networks with maximal period exist? Indeed they do; in the next three sections we give three different constructions that yield maximal neural networks. Although the constructions are different, the resulting neural networks turn out to be isomorphic. The construction in Section 3.1.1 was given in [4] and later rewritten in [74]. Section 3.1.2 gives a construction by Orlitsky, which appeared in [67]. Finally, the construction in section 3.1.3 was recently given in [10].

3.1.1 Arimoto's construction

In this section we will survey a proof of the existence of maximal period networks of every dimension. This proof comes from Arimoto ([4], in Japanese), and was later rewritten by Ueda [74].

Theorem 3.1:
Let u be an n-dimensional self-dual threshold function. For $i = 1, 2, \ldots, n$, set $f_i = x_i\overline{u}$. Then for $i = 1, 2, \ldots, n$ we have $u = x_i\overline{f_i} \vee \underline{f_i}$.

Proof:
Without loss of generality, we prove the theorem for $i = 1$. By Lemma 2.3 on page 20, there exists an $(n-1)$-dimensional threshold function g such that

$$u_{(}x_1, \ldots, x_n) = x_1\overline{g}(x_2, \ldots, x_n) \vee \overline{x_1}\underline{g}(x_2, \ldots, x_n).$$

We have

$$f_1 = x_1\overline{u} = x_1\left[\overline{x_1} \vee g\right]\left[x_1 \vee \underline{\overline{g}}\right] = x_1 g$$

by De Morgan's Law. This gives us $x_1\overline{f} = x_1\overline{g}$ since $\overline{f_1} = \overline{x_1} \vee \overline{g}$. Therefore,

$$u = x_1\overline{g} \vee \overline{x_1}\underline{g} = x_1\overline{f_1} \vee \overline{x_1}\underline{g} = x_1\overline{f_1} \vee \underline{f_1}. \tag{3.1}$$

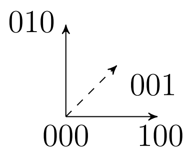
□

Using this, neural networks that consist of self-dual threshold functions can be represented in a special way:

Definition 3.4:
If $\Psi : \{0,1\}^n \to \{0,1\}^n$ is a neural network consisting of self-dual threshold functions $u_1, u_2, \ldots, u_n$, then the *Ueda form* of Ψ is $[x_1\overline{u_1}, x_2\overline{u_2}, \ldots, x_n\overline{u_n}]$.

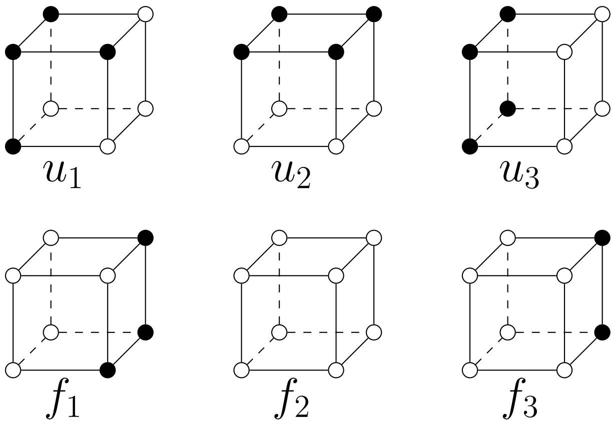
The functions $x_i\overline{u_i}$ in the Ueda form of a self-dual neural network may be much simpler than the underlying self-dual threshold functions u_i, making them easier to analyze. This is its primary advantage for us, since it turns out to be the case in Theorem 3.2 at the end of this section. Other properties of this form can be found in [73]. For example, $\mathbf{x}$ is a fixed point of the neural network with Ueda form $[f_1, f_2, \ldots, f_n]$ if and only if $f_i(\mathbf{x}) = f_i(\overline{\mathbf{x}}) = 0$ for each $i = 1, 2, \ldots, n$.

Example 3.2:
Using the coordinate system



consider the 3-dimensional neural network given by the 3 self-dual threshold functions u_1, u_2, u_3:

This network is given in Ueda form as $[f_1, f_2, f_3]$, where f_1, f_2, f_3 are as follows:



The fixed points of this network are 110 and 001. Note that $f_i(110) = f_i(001) = 0$ for $i = 1, 2, 3$. Furthermore, for every other $\mathbf{x}$, there exists an i such that $f_i(\mathbf{x}) = 1$ or $f_i(\overline{\mathbf{x}}) = 1$. ■

Lemma 3.1:
Let f be an n-dimensional threshold function. For $j = 1, 2, \ldots, n$, the function $x_j f$ is a threshold function.

Proof:
Let f be threshold with weights $w_1, w_2, \ldots, w_n$ and threshold value θ. Fix j and set

$$\begin{aligned}
&r = \sum_{i \neq j} |w_i| - \theta + 1; \\
&v_i = w_i \quad \text{for} \quad i \neq j; \\
&v_j = w_j + r; \\
&\eta = \theta + r.
\end{aligned}$$

We use this to define a new threshold function g with weights $v_1, v_2, \ldots, v_n$ and threshold η. We will now show that $g(\mathbf{x}) = x_j f(\mathbf{x})$ for every $\mathbf{x}$ in $\{0,1\}^n$. First, suppose that $x_j = 1$. Then

$$\sum_i v_i x_i = \sum_{i \neq j} v_i x_i + v_j = \sum_{i \neq j} w_i x_i + w_j + r.$$

Thus, if $f(\mathbf{x}) = 1$ then $\sum v_i x_i \geq \theta + r = \eta$. If $f(\mathbf{x}) = 0$ then $\sum v_i x_i < \theta + r = \eta$. This proves the claim for this case.

If $x_j = 0$ then

$$\sum_i v_i x_i = \sum_{i \neq j} w_i x_i < \sum_{i \neq j} |w_i| + 1 = r + \theta = \eta,$$

so that $g(\mathbf{x}) = 0$. □

Corollary 3.1:
Let f be an n-dimensional threshold function. For each $j = 1, 2, \ldots, n$ the n-dimensional function $\overline{x_j} \vee f$ is a threshold function.

Proof:
Note that $\overline{x_j} \vee f = \overline{x_j \overline{f}}$. By Lemma 3.1, we know that $x_j \overline{f}$ is threshold, and by Lemma 2.1, the complement of a threshold function is also threshold. □

We shall now construct a neural network and proceed to show that it is maximal. For all n, we will define n boolean functions

$$f_i^n : \{0,1\}^n \to \{0,1\} \quad i = 1, 2, \ldots, n,$$

which we will use to describe an n-dimensional neural network Ψ^n in Ueda form. To define $f_i^n(x_1, x_2, \ldots, x_n)$, let us first write

$$c_i^n = \begin{cases} 1 & \text{if } i = 1; \\ x_1 x_2 \ldots x_{i-1} & \text{for } i = 2, 3 \ldots, n \end{cases}$$

and

$$d_i^n = \begin{cases} x_n & \text{if } i = n \\ \\ x_i \overline{d_{i+1}^n} & \text{for } i = 1, 2 \ldots, n-1. \end{cases}$$

We define the functions $f_i^n(x_1, x_2, \ldots, x_n)$ by

$$f_i^n(x_1, x_2, \ldots, x_n) = c_i^n d_i^n \qquad \text{for } i = 1, \ldots, n. \tag{3.2}$$

Lemma 3.2:
The functions $f_i^n(x_1, \ldots, x_n)$ defined in (3.2) can be written explicitly as

$$\begin{aligned}
f_n^n &= x_1x_2\cdots x_n\\
f_{n-1}^n &= x_1x_2\cdots x_{n-1}\overline{x_n}\\
f_{n-2}^n &= x_1x_2\cdots x_{n-2}\left(\overline{x_{n-1}} \vee x_n\right)\\
f_{n-3}^n &= x_1x_2\cdots x_{n-3}\left(\overline{x_{n-2}} \vee x_{n-1}\overline{x_n}\right)\\
f_{n-4}^n &= x_1x_2\cdots x_{n-4}\left(\overline{x_{n-3}} \vee x_{n-2}\left(\overline{x_{n-1}} \vee x_n\right)\right)\\
f_{n-5}^n &= x_1x_2\cdots x_{n-5}\left(\overline{x_{n-4}} \vee x_{n-3}\left(\overline{x_{n-2}} \vee x_{n-1}\overline{x_n}\right)\right)\\
&\vdots\\
f_1^n &= \begin{cases} x_1(\overline{x_2} \vee x_3(\overline{x_4} \vee x_5(\overline{x_6} \vee x_7(\cdots(\overline{x_{n-2}} \vee x_{n-1}\overline{x_n})))\ldots) & \textit{if} \quad n \textit{ is even;}\\ x_1(\overline{x_2} \vee x_3(\overline{x_4} \vee x_5(\overline{x_6} \vee x_7(\cdots \vee x_{n-2}(\overline{x_{n-1}} \vee x_n)))\ldots) & \textit{if} \quad n \textit{ is odd.}\end{cases}
\end{aligned}$$

Proof:
Let n be a fixed positive integer. We will prove the claim by showing that

$$\begin{aligned}
d_n^n &= x_n\\
d_{n-1}^n &= x_{n-1}\overline{x_n}\\
d_{n-2}^n &= x_{n-2}\left(\overline{x_{n-1}} \vee x_n\right)\\
d_{n-3}^n &= x_{n-3}\left(\overline{x_{n-2}} \vee x_{n-1}\overline{x_n}\right)\\
d_{n-4}^n &= x_{n-4}\left(\overline{x_{n-3}} \vee x_{n-2}\left(\overline{x_{n-1}} \vee x_n\right)\right)\\
d_{n-5}^n &= x_{n-5}\left(\overline{x_{n-4}} \vee x_{n-3}\left(\overline{x_{n-2}} \vee x_{n-1}\overline{x_n}\right)\right)\\
&\vdots\\
d_{n-i}^n &= \begin{cases} x_{n-i}(\overline{x_{n-i+1}} \vee x_{n-i+2}(\overline{x_{n-i+3}} \vee x_{n-i+4}(\cdots(\overline{x_{n-2}} \vee x_{n-1}\overline{x_n})))\ldots) & \text{if} \quad i \text{ is even;}\\ x_{n-i}(\overline{x_{n-i+1}} \vee x_{n-i+2}(\overline{x_{n-i+3}} \vee x_{n-i+4}(\cdots \vee x_{n-2}(\overline{x_{n-1}} \vee x_n)))\ldots) & \text{if} \quad i \text{ is odd,}\end{cases}
\end{aligned}$$

and we do this by induction on i. Our base case is when $i = n$, which is confirmed by the definition of d_n^n. Supposing the claim holds for $d_{n-(i-1)}^n$, we will show that it holds for d_{n-i}^n as well. By the definitions of d_{n-i}^n and d_{n-i+1}^n, and the application of De Morgan's Law, we have

$$d_{n-i}^n = x_{n-i}\overline{d_{n-(i-1)}^n} = x_{n-i}\overline{\left(x_{n-i+1}\overline{d_{n-i+1}}\right)} = x_{n-i}\left(\overline{x_{n-i+1}} \vee d_{n-i+1}\right).$$

Therefore, by the induction hypothesis,

$$d^n_{n-i} = \begin{cases} x_{n-i}\left(\overline{x_{n-i+1}} \vee x_{n-i+1}\left(\overline{x_{n-i+2}} \vee x_{n-i+3}(\cdots x_n)\right)\right) & \text{if } i \text{ is even} \\ x_{n-i}\left(\overline{x_{n-i+1}} \vee x_{n-i+1}\left(\overline{x_{n-i+2}} \vee x_{n-i+3}(\cdots \overline{x_n})\right)\right) & \text{if } i \text{ is odd.} \end{cases} \qquad \square$$

Note that by Lemma 3.1 and Corollary 3.1, the functions $f^n_1, f^n_2, \ldots, f^n_n$ are threshold functions, so $\Psi^n = [f^n_1, f^n_2, \ldots, f^n_n]$ is indeed a neural network consisting of self-dual threshold functions. We are now ready to show that it is maximal.

Theorem 3.2:
For every positive integer n, the neural network Ψ^n given in Ueda form as $\Psi^n = [f^n_1, f^n_2, \ldots, f^n_n]$ is maximal.

Proof:
To begin, we claim that for all $i = 1, 2, \ldots, n-1$ we have

$$f^n_i(x_1, x_2, \ldots, x_n) = \begin{cases} \overline{f^{n-1}_i(x_1, x_2, \ldots, x_{n-1})} & \text{if } x_1 = x_2 = \ldots = x_n = 1; \\ \\ f^{n-1}_i(x_1, x_2, \ldots, x_{n-1}) & \text{otherwise.} \end{cases} \tag{3.3}$$

More precisely, when a bit string $\mathbf{x}$ does not consist entirely of ones and j is the first index such that $x_j = 0$, we have

$$f^{n-1}_i(\mathbf{x}) = f^n_i(\mathbf{x}) = \begin{cases} 0 \\ \text{ if } j \le i; \\ \\ x_1 \cdot x_2 \cdot \cdots \cdot x_i\left(\overline{x_{i+1}} \vee x_{i+2} \cdot \left(\overline{x_{i+3}} \cdots \vee x_{j-1}\right) \cdots\right) \\ \text{ if } j > i \;\;\&\;\; j - i \equiv 1 \pmod 2; \\ \\ x_1 \cdot x_2 \cdot \cdots \cdot x_i\left(\overline{x_{i+1}} \vee x_{i+2} \cdot \left(\overline{x_{i+3}} \cdots \vee x_{j-2} \cdot \overline{x_{j-1}}\right) \cdots\right) \\ \text{ if } j > i \;\;\&\;\; j - i \equiv 0 \pmod 2. \end{cases}$$

and, with 1^n standing for the bit string that consists of n ones, we have

$$f^n_i(1^n) = \begin{cases} 0 & \text{if } n - i \equiv 0 \pmod n; \\ 1 & \text{if } n - i \equiv 1 \pmod n, \end{cases}$$

$$f_i^{n-1}(1^{n-1}) = \begin{cases} 0 & \text{if } n-1-i \equiv 0 \pmod{n-1}; \\ 1 & \text{if } n-1-i \equiv 1 \pmod{n-1}; \end{cases}$$

These observations constitute the proof of claim (3.3).

We will use induction on n to prove that the neural network Ψ^n is maximal. As for the induction basis, the Ueda form of Ψ^1 is $[f_1^1]$, where $f_1^1(x_1) = x_1$ and the 1-dimensional self-dual function u such that $f_1 = x_1\overline{u}$ is $u = \overline{x_1}$. Thus, the trajectory of 0 under Ψ^1 is $0 \to 1 \to 0 \to \dots$ as desired.

As for the induction step, suppose that the network Ψ^{n-1} is maximal: the trajectory of $\mathbf{1}^{n-1}$ under Ψ^{n-1} is

$$\mathbf{x}^1 \to \mathbf{x}^2 \to \dots \to \mathbf{x}^{2^{n-1}} \to \mathbf{x}^1 \to \dots,$$

where $\mathbf{x}^1 = 1^{n-1}$ and $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^{2^{n-1}}$ are pairwise distinct. By Theorem 3.1, the self-dual function u_n^n such that $f_n^n = x_n\overline{u_n^n}$ satisfies $u_n^n = x_n\overline{f_n^n} \vee \underline{f_n^n}$. Therefore,

$$u_n^n = x_n\overline{(x_1x_2\cdots x_n)} \vee \left(\overline{x_1}\,\overline{x_2}\cdots\overline{x_n}\right) = \begin{cases} \overline{x_n} & \text{if } x_1 = x_2 = \cdots = x_n = 1 \quad (3.4) \\ x_n & \text{otherwise.} \quad (3.5) \end{cases}$$

We conclude that Ψ^n has the following state transitions:

$$\begin{array}{rclll}
\mathbf{x}^1 1 & \to & \overline{\mathbf{x}^2}0 & & \text{by (3.3) and (3.4)} \\
\overline{\mathbf{x}^1}0 & \to & \mathbf{x}^2 1 & & \text{by (3.3), (3.4), and self-duality} \\
\mathbf{x}^i 1 & \to & \mathbf{x}^{i+1}1 & \text{for } i = 2, 3, \dots, 2^{n-1}-1 & \text{by (3.3) and (3.5)} \\
\overline{\mathbf{x}^i}0 & \to & \overline{\mathbf{x}^{i+1}}0 & \text{for } i = 2, 3, \dots, 2^{n-1}-1 & \text{by (3.3) and (3.5)} \\
\mathbf{x}^{2^{n-1}}1 & \to & \mathbf{x}^1 1 & & \text{by (3.3) and (3.5)} \\
\overline{\mathbf{x}^{2^{n-1}}}0 & \to & \overline{\mathbf{x}^1}0 & & \text{by (3.3), (3.5) and self-duality.}
\end{array}$$

Thus, Ψ^n yields the following period trajectory:

$$\mathbf{x}^1 1 \to \overline{\mathbf{x}^2}0 \to \overline{\mathbf{x}^3}0 \to \cdots \to \overline{\mathbf{x}^{2^{n-1}}}0 \to \overline{\mathbf{x}^1}0 \to \mathbf{x}^2 1 \to \mathbf{x}^3 1 \to \cdots \to \mathbf{x}^{2^{n-1}}1 \to \mathbf{x}^1 1, \tag{3.6}$$

which has a period of 2^n □

3.1.2 Orlitsky's construction

In this section we survey another proof of the existence of maximal n-dimensional neural networks for all n. The original proof comes from an unpublished manuscript by Orlitsky; we follow the proof in its published form, found in [67], where the motivation was related to the *transient period* of a neural network, which is the number of iterations required before a state is revisited. Recall that Hopfield established criteria under which recurrent neural networks were guaranteed to converge to a fixed point [28]. In [23], Goles and Martinez showed that even under these criteria, it is possible that the number of iterations required before reaching a fixed point could be exponential in the network size. The presentation of Orlitsky's contruction in [67] emphasizes that transient periods in the general setting, where no fixed point is guaranteed, can also be exponential in the size of the network.

In Section 3.1.4, we will show that the network constructed here is isomorphic to the one constructed in the previous section, and that of the next section. For this construction, we switch over to the $\{-1, 1\}$ symbol set. We will only be considering neural networks in which

1. all thresholds are set to 0;
2. the underlying threshold functions are non-degenerate.

Let $\Phi : \{-1, 1\}^n \to \{-1, 1\}^n$ be a neural network, where $w_{i,j}$ is the jth weight of neuron i. If Φ satisfies 1 and 2 above, then its state transitions are defined by

$$\Phi_i^t(\mathbf{x}) = \operatorname{sgn}\left(\sum_{j=1}^{n} w_{i,j}\Phi_j^{t-1}(\mathbf{x})\right),$$

where sgn is the *sign* function, defined by

$$\operatorname{sgn}(z) = \begin{cases} 1 \text{ if } z > 0; \\ 0 \text{ if } z = 0; \\ -1 \text{ if } z < 0. \end{cases}$$

Lemma 3.3:
Let $\Phi : \{-1, 1\}^n \to \{-1, 1\}^n$ be a neural network such that all thresholds are 0. If there exist $\mathbf{x}$ and $s > 0$ and t such that $\Phi^t(\mathbf{x}) = -\Phi^{t+s}(\mathbf{x})$, then for all $k \geq 0$ we have

$$\Phi^{t+k}(\mathbf{x}) = -\Phi^{t+s+k}(\mathbf{x}).$$

Proof:
The proof is by induction on k. The case when $k = 0$ holds by the hypothesis of the Lemma. Suppose the claim holds for all values up to and including $k - 1$, and let $w_{i,j}$

be the jth weight of the ith threshold function. Then

$$\Phi^{t+k-1}(\mathbf{x}) = -\Phi^{t+s+k-1}(\mathbf{x})$$

implies that for $i = 1, \ldots, n$ we have

$$\sum_{j=1}^{n} w_{i,j}\Phi_j^{t+k-1}(\mathbf{x}) = -\sum_{j=1}^{n} w_{i,j}\Phi_j^{t+s+k-1}(\mathbf{x}).$$

Therefore,

$$\Phi^{t+k}(\mathbf{x}) = \operatorname{sgn}\left(\sum_{j=1}^{n} w_{i,j}\Phi_i^{t+k-1}(\mathbf{x})\right) = -\operatorname{sgn}\left(\sum_{j=1}^{n} w_{i,j}\Phi_i^{t+s+k-1}(\mathbf{x})\right) = -\Phi^{t+s+k}(\mathbf{x}).$$

$\square$

We will now describe Orlitsky's neural network. For every positive integer n, we define an n-dimensional neural network Ω_n as follows. For $1 \leq i, j \leq n$ we define the weight

$$w_{n,i,j} = \begin{cases} -\frac{3}{2^{j-1}} & 1 \leq i < j \leq n; \\ \\ j - 1 - \frac{1}{2^{j-1}} & 1 \leq i = j \leq n; \\ \\ (-1)^{i-j-1} & 1 \leq j < i \leq n; \end{cases}$$

all thresholds are set to 0.

When $n = 1$, we have the weight matrix

$$\begin{pmatrix} -1 \end{pmatrix},$$

and state transitions

$$+1 \to -1 \to +1 \to \cdots .$$

When $n = 2$, the weight matrix is

$$\begin{pmatrix} -1 & -\frac{3}{2} \\ 1 & \frac{1}{2} \end{pmatrix},$$

and the state transitions are

$$(+1, +1) \to (-1, +1) \to (-1, -1) \to (+1, -1) \to (+1, +1) \to \cdots .$$

When $n = 3$, the weight matrix is

$$\begin{pmatrix} -1 & -\frac{3}{2} & -\frac{3}{4} \\ 1 & \frac{1}{2} & -\frac{3}{4} \\ -1 & 1 & 1\frac{3}{4} \end{pmatrix},$$

and the state transitions are

$$\begin{aligned}&(+1,+1,+1) \to (-1,+1,+1) \to (-1,-1,+1) \to (+1,-1,+1) \to \\ &(-1,-1,-1) \to (+1,-1,-1) \to (+1,+1,-1) \to (-1,+1,-1) \to \\ &(+1,+1,+1) \to \cdots .\end{aligned}$$

When $n = 4$, the weight matrix is

$$\begin{pmatrix} -1 & -\frac{3}{2} & -\frac{3}{4} & -\frac{3}{8} \\ 1 & \frac{1}{2} & -\frac{3}{4} & -\frac{3}{8} \\ -1 & 1 & 1\frac{3}{4} & -\frac{3}{8} \\ 1 & -1 & 1 & 2\frac{7}{8} \end{pmatrix},$$

and the state transitions are

$$\begin{aligned}&(+1,+1,+1,+1) \to (-1,+1,+1,+1) \to (-1,-1,+1,+1) \to (+1,-1,+1,+1) \to \\ &(-1,-1,-1,+1) \to (+1,-1,-1,+1) \to (+1,+1,-1,+1) \to (-1,+1,-1,+1) \to \\ &(-1,-1,-1,-1) \to (+1,-1,-1,-1) \to (+1,+1,-1,-1) \to (-1,+1,-1,-1) \to \\ &(+1,+1,+1,-1) \to (-1,+1,+1,-1) \to (-1,-1,+1,-1) \to (+1,-1,+1,-1) \to \\ &(+1,+1,+1,+1) \to \cdots .\end{aligned}$$

For the rest of this section, the initial seed for Ω_n will be the n-dimensional vector of ones, $\mathbf{x} = 1^n$, and we will write $y_n(t) = \Omega_n^t(\mathbf{x})$, so $y_n(t)$ denotes the state of the vector of all ones under Ω_n at time t. We also write $y_{n,i}(t)$ to denote the ith coordinate of $y_n(t)$. Furthermore, we write

$$P_{n,i}(t) = \sum_{j=1}^{n} w_{n,i,j} y_{n,j}(t),$$

so that $P_{n,i}(t)$ denotes the the *potential* at neuron i at time t, hence $y_n(t+1) = \mathrm{sgn}(P_{n,i}(t))$.

We prove the maximality of Ω_n by proving the following:

Theorem 3.3:
For all positive integers n, the following hold:

$$y_n(0), y_n(1), \ldots, y_n(2^n-1) \text{ are pairwise distinct;} \tag{3.7}$$

and,

$$y_{n,i}(2^n-1) = (-1)^{n-i-1} \qquad i = 1, \ldots, n; \tag{3.8}$$

Furthermore, for $i = 1, \ldots, n$ we have

$$P_{n,i}(t) \le \frac{-1}{2^{n-1}} \quad \text{or} \quad P_{n,i}(t) \ge \frac{3}{2^{n-1}} \qquad t = 1, \ldots, 2^n - 2 \tag{3.9}$$

and

$$P_{n,i}(2^n-1) = \frac{1}{2^{n-1}}. \tag{3.10}$$

Proof:
The proof is by induction on n. When $n = 1$ there is one weight, and it's value is -1. This yields the trajectory $+1, -1, +1, \ldots$. Furthermore, we have $P^1_{1,1} = (-1)(-1) = 1$. We proceed by assuming that (3.7), (3.8), (3.9), and (3.10) hold for all dimensions up to and including n, and will prove them true for $n+1$. We will do this by proving the following two claims:

$$y_{n+1}(t) = y_n(t)\,1 \qquad \text{for} \quad 0 \le t \le 2^n - 1; \tag{3.11}$$

and

$$y_{n+1}(2^n) = (-1, -1, \ldots, -1). \tag{3.12}$$

We first prove (3.11) by induction on t. This claim holds for $t = 0$, since the seed is the vector of ones. Suppose it holds up to and including t, for $t < 2^n - 1$. For $i = 1, \ldots, n$ we have

$$\begin{aligned}
P_{n+1,i}(t) &= \sum_{j=1}^{n+1} w_{n+1,i,j} y_{n+1,j}(t) \\
&= \sum_{j=1}^{n} w_{n+1,i,j} y_{n+1,j}(t) + w_{n+1,i,n+1} y_{n+1,n+1}(t) \\
&= \sum_{j=1}^{n} w_{n,i,j} y_{n+1,j}(t) + w_{n+1,i,n+1} y_{n+1,n+1}(t) && \text{by how the weights are defined} \\
&= \sum_{j=1}^{n} w_{n,i,j} y_{n,j}(t) + w_{n+1,i,n+1} y_{n+1,n+1}(t) && \text{by the induction hypothesis on } t \\
&= P_{n,i}(t) - \left(\frac{3}{2^n}\right) \cdot 1.
\end{aligned}$$

By (3.9), we have $P_{n,i}(t) \geq 3/2^{n-1}$ or $P_{n,i}(t) \leq -1/2^{n-1}$. In the former case, we have

$$P_{n+1,i}(t) \geq \frac{3}{2^{n-1}} - \frac{3}{2^n} = \frac{3}{2^n}; \tag{3.13}$$

and in the latter

$$P_{n+1,i}(t) \leq -\frac{1}{2^{n-1}} - \frac{3}{2^n} \leq -\frac{5}{2^n}. \tag{3.14}$$

In either case, we get

$$y_{n+1,i}(t+1) = y_{n,i}(t+1) \qquad (i = 1, \ldots, n).$$

Similarly, for the last coordinate we have

$$\begin{aligned} P_{n+1,n+1}(t) &= \sum_{j=1}^{n+1} w_{n+1,n+1,j} y_{n+1,j}(t) \\ &= \sum_{j=1}^{n} (-1)^{n-j} y_{n,j}(t) + \left(n - \frac{1}{2^n}\right) \cdot 1 \\ &\geq 2 - \frac{1}{2^n} \qquad \text{by (3.7)} \\ &\geq \frac{3}{2^n}. \end{aligned} \tag{3.15}$$

Therefore, $y_{n+1,n+1}(t+1) = 1$, and (3.11) is confirmed.

Next, we prove claim (3.12).

For $i = 1, \ldots, n$, we have

$$\begin{aligned} P_{n+1,i}(2^n - 1) &= \sum_{j=1}^{n+1} w_{n+1,i,j} y_{n+1,j}(2^n - 1) \\ &= \sum_{j=1}^{n} w_{n+1,i,j} y_{n+1,j}(2^n - 1) + w_{n+1,i,n+1} y_{n+1,n+1}(2^n - 1) \\ &= \sum_{j=1}^{n} w_{n,i,j} y_{n,j}(2^n - 1) - \frac{3}{2^n} \cdot 1 \quad \text{by (3.11) and weight definitions} \\ &= P_{n,i}(2^n - 1) - \frac{3}{2^n} \cdot 1 \qquad \text{by (3.11)} \\ &= \frac{1}{2^{n-1}} - \frac{3}{2^n} \qquad \text{by the induction hypothesis and (3.10)} \\ &= -\frac{1}{2^n}. \end{aligned}$$

Similarly, for the last neuron, we have

$$\begin{aligned}
P_{n+1,n+1}(2^n-1) &= \sum_{j=1}^{n+1} w_{n+1,n+1,j}\, y_{n+1,j}(2^n-1)\\
&= \sum_{j=1}^{n} w_{n,n+1,j}\, y_{n,j}(2^n-1) + w_{n+1,n+1,n+1}\, y_{n+1,n+1}(2^n-1)\\
&= \sum_{i=1}^{n} (-1)^{n-j}(-1)^{n-j-1} + \left(n-\frac{1}{2^n}\right)\\
&= -\frac{1}{2^n}.
\end{aligned}$$

This proves (3.12).

We now use (3.11) and (3.12) to complete the induction step on n and the proof of the Theorem. By the induction hypothesis and (3.11), we get that

$$y_{n+1}(0),\ y_{n+1}(1),\ \ldots,\ y_{n+1}(2^n-1)$$

are pairwise distinct and that the last coordinate of each state is $+1$. By (3.12), we have $y_{n+1}(2^n) = (-1,-1,\ldots,-1) = -y_{n+1}(0)$. This allows us to apply Lemma 3.3, which yields

$$y_{n+1}(t) = -y_{n+1}(t-2^n) \qquad t = 2^n,\ldots,2^{n+1}-1.$$

Therefore, $y_{n+1}(2n), y_{n+1}(2n+1),\ldots,y_{n+1}(2^{n+1}-1)$ are pairwise distinct and the last coordinate of each state is -1. This establishes (3.7); the first 2^n states are pairwise distinct. For the last state, (3.11) and the induction hypothesis yield

$$y_{n+1,i}(2^n-1) = y_{n,i}(2^n-1) = (-1)^{n-i-1} \qquad i = 1,\ldots,n$$

and

$$y_{n+1,n+1}(2^n-1) = 1 = (-1)^{n-(n+1)-1}.$$

Therefore, Lemma 3.3 implies

$$y_{n+1,i}(2^{n+1}-1) = -y_{n+1,i}(2^n-1) = (-1)^{(n+1)-i-1} \qquad i = 1,\ldots,n+1,$$

and (3.8) is established.

From (3.12) and Lemma 3.3, we conclude that (3.9) holds.

Finally, (3.10) holds when $t = 1,\ldots,2^n-2$ by inequalities (3.13), (3.15), (3.15); when $t = 2^n,\ldots,2^{n+1}-1$ by those same inequalities and Lemma 3.3; and when $t = 2^n-1$ by (3.12). $\square$

Corollary 3.2:
The neural network Ω_n is maximal for all n.

Proof:
This follows immediately from (3.7) and (3.8) of the Theorem. $\square$

3.1.3 Alternative construction

We shall now present a third, new proof of the existence of maximal neural networks given in [10]. This construction will yield a maximal neural network similar to that of Orlitsky's, but using the $\{0, 1\}$ symbol set. Another difference is in the proof technique; Orlitsky began by defining a weight matrix and then showed that the resulting trajectory had a special structure. The proof below defines a trajectory with a similar special structure and then proceeds to show that weights and thresholds exist that realize it.

For every positive integer n, we define a mapping $\Phi_n : \{0, 1\}^n \to \{0, 1\}^n$ by

$$\Phi_n(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_n)$$

where, with m the largest subscript such that $(x_1, x_2, \ldots, x_m)$ is an alternating vector, $(0, 1, 0, 1, \ldots)$ or $(1, 0, 1, 0, \ldots)$,

$$y_i = \begin{cases} \overline{x}_m & \text{when } 1 \leq i \leq m, \\ x_i & \text{when } m < i \leq n. \end{cases} \tag{3.16}$$

For instance,

$$\begin{aligned}
\Phi_4(0,0,0,0) &= (1,0,0,0), & \Phi_4(0,0,0,1) &= (1,0,0,1),\\
\Phi_4(0,0,1,0) &= (1,0,1,0), & \Phi_4(0,0,1,1) &= (1,0,1,1),\\
\Phi_4(0,1,0,0) &= (1,1,1,0), & \Phi_4(0,1,0,1) &= (0,0,0,0),\\
\Phi_4(0,1,1,0) &= (0,0,1,0), & \Phi_4(0,1,1,1) &= (0,0,1,1),\\
\Phi_4(1,0,0,0) &= (1,1,0,0), & \Phi_4(1,0,0,1) &= (1,1,0,1),\\
\Phi_4(1,0,1,0) &= (1,1,1,1), & \Phi_4(1,0,1,1) &= (0,0,0,1),\\
\Phi_4(1,1,0,0) &= (0,1,0,0), & \Phi_4(1,1,0,1) &= (0,1,0,1),\\
\Phi_4(1,1,1,0) &= (0,1,1,0), & \Phi_4(1,1,1,1) &= (0,1,1,1).
\end{aligned}$$

Note that the definition of Φ_n implies that Φ_n is self-dual and that

$$\Phi_1(0) = 1 \tag{3.17}$$

and that, when $n \geq 2$,

$$\Phi_n(x_1, x_2, \ldots, x_{n-1}, 0) = \begin{cases} (1, 1, \ldots, 1, 1) \text{ if } (x_1, x_2, \ldots, x_n) \text{ is} \\ \quad \text{the alternating vector } (\ldots, 1, 0, 1, 0), \\ (\Phi_{n-1}(x_1, x_2, \ldots, x_{n-1}), 0) \text{ otherwise.} \end{cases} \tag{3.18}$$

Lemma 3.4:
For every positive integer n there are threshold functions

$$f_{n,i} : \{0,1\}^n \to \{0,1\} \qquad (i = 1, 2, \ldots, n)$$

such that

$$\Phi_n(\mathbf{x}) = (f_{n,1}(\mathbf{x}), f_{n,2}(\mathbf{x}), \ldots, f_{n,n}(\mathbf{x})) \textit{ for all } \mathbf{x} \textit{ in } \{0,1\}^n. \tag{3.19}$$

Proof:
For every positive integer n and for all $i = 1, 2, \ldots, n$, we will construct weights $w_{n,i,\,j}$ $(j = 1, 2, \ldots, n)$ and threshold values $\theta_{n,i}$. Then we will define

$$f_{n,i}(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } \sum_{j=1}^n w_{n,i,\,j} x_j \geq \theta_{n,i} \\ 0 & \text{otherwise} \end{cases}$$

and prove that (3.19) is satisfied.

Our construction of $w_{n,i,\,j}$ and $\theta_{n,i}$ is recursive. To begin, we set

$$w_{1,1,1} = -1,\ \theta_{1,1} = 0;$$

for all integers n greater than 1, we set

$$w_{n,n,\,j} = \begin{cases} 1 & \text{if } j \not\equiv n \bmod 2, \\ -1 & \text{if } j \equiv n \bmod 2 \text{ and } j < n, \\ n-2 & \text{if } j = n, \end{cases}$$
$$\theta_{n,n} = \lfloor n/2 \rfloor,$$

$$w_{n,n-1,\,j} = \begin{cases} w_{n-1,n-1,\,j} & \text{if } j \leq n-2, \\ w_{n-1,n-1,\,j} + 1 & \text{if } j = n-1, \\ -1 & \text{if } j = n, \end{cases}$$
$$\theta_{n,n-1} = \theta_{n-1,n-1},$$

and, when $i = 1, 2, \ldots, n-2$,

$$w_{n,i,\,j} = \begin{cases} w_{n-1,i,\,j} + w_{n-2,i,\,j} & \text{if } j \le n-2, \\ w_{n-1,i,\,j} & \text{if } j = n-1, \\ -1 & \text{if } j = n, \end{cases}$$
$$\theta_{n,i} = \theta_{n-1,i} + \theta_{n-2,i} - 1.$$

Since the sequence $\Phi_1, \Phi_2, \Phi_3, \ldots$ is completely determined by its self-duality and properties (3.17), (3.18), proving that (3.19) is satisfied reduces to proving that

(i) $f_{n,i}(\overline{\mathbf{x}}) = \overline{f_{n,i}(\mathbf{x})}$ for all $i = 1, \ldots n$ and all $\mathbf{x}$ in $\{0,1\}^n$,

observing that $f_{1,1}(0) = 1$, and proving that

(ii) if $\mathbf{x}$ in $\{0,1\}^n$ is the alternating vector $(\ldots, 1, 0, 1, 0)$,
then $f_{n,i}(\mathbf{x}) = 1$ for all $i = 1, \ldots n$,

(iii) if $(x_1, \ldots, x_{n-1}, 0)$ in $\{0,1\}^n$ is not the alternating vector $(\ldots, 1, 0, 1, 0)$,
then $f_{n,n}(x_1, \ldots, x_{n-1}, 0) = 0$,

(iv) if $(x_1, \ldots, x_{n-1}, 0)$ in $\{0,1\}^n$ is not the alternating vector $(\ldots, 1, 0, 1, 0)$,
then $f_{n,i}(x_1, \ldots, x_{n-1}, 0) = f_{n-1,i}(x_1, \ldots, x_{n-1})$ for all $i = 1, \ldots n-1$.

Straightforward, if a little tedious, induction on n shows that

$$\sum_{j \not\equiv n \bmod 2} w_{n,i,\,j} = \theta_{n,i} \qquad \text{for all } i = 1, \ldots n, \tag{3.20}$$
$$\sum_{j \equiv n \bmod 2} w_{n,i,\,j} = \theta_{n,i} - 1 \qquad \text{for all } i = 1, \ldots n. \tag{3.21}$$

Summing up each pair of these equations, we conclude that

$$\textstyle\sum_{j=1}^{n} w_{n,i,\,j} = 2\theta_{n,i} - 1 \text{ for all } i = 1, \ldots n,$$

which is easily seen to imply (i); equations (3.20) alone imply directly (ii); proposition (iii) follows from the definitions of $w_{n,n,\,j}$ and $\theta_{n,n}$.

In proving (iv), we will treat $i = n-1$ separately from $i \le n-2$.

To prove that $f_{n,n-1}(x_1, x_2, \ldots, x_{n-1}, 0) = f_{n-1,n-1}(x_1, x_2 \ldots, x_{n-1})$ for all $(x_1, \ldots, x_{n-1})$ in $\{0,1\}^{n-1}$ other than the alternating vector $(\ldots, 0, 1, 0, 1)$, recall that

$$\textstyle\sum_{j=1}^{n-1} w_{n,n-1,\,j} x_j = \sum_{j=1}^{n-1} w_{n-1,n-1,\,j} x_j + x_{n-1} \text{ and } \theta_{n,n-1} = \theta_{n-1,n-1}, .$$

It follows that

$$f_{n,n-1}(x_1,\dots,x_{n-1},0) \neq f_{n-1,n-1}(x_1,\dots,x_{n-1})$$

if and only if $x_{n-1}=1$ and

$$\sum_{j=1}^{n-1} w_{n-1,n-1,\,j}x_j + 1 = \theta_{n-1,n-1}.$$

Since $w_{n-1,n-1,\,n-1}=n-3$ and $\theta_{n-1,n-1}=\lfloor (n-1)/2\rfloor$, this means

$$\sum_{j=1}^{n-2} w_{n-1,n-1,\,j}x_j = -\lceil (n-3)/2\rceil;$$

since

$$w_{n-1,n-1,\,j} = \begin{cases} 1 & \text{if } j \not\equiv n-1 \bmod 2, \\ -1 & \text{if } j \equiv n-1 \bmod 2 \text{ and } j < n-1, \end{cases}$$

this means further that $(x_1,\dots,x_{n-1})$ is the alternating vector $(\dots,0,1,0,1)$.

To prove that we have $f_{n,i}(x_1,\dots,x_{n-1},0) = f_{n-1,i}(x_1,\dots,x_{n-1})$ for all $i=1,\dots,n-2$ and for all $(x_1,\dots,x_{n-1})$ in $\{0,1\}^{n-1}$ other than the alternating vector $(\dots,0,1,0,1)$, we shall use induction on n. In the induction step, we distinguish between two cases.

Case 1: $(x_1,\dots,x_{n-1})$ *is the alternating vector* $(\dots,1,0,1,0)$.
In this case, we do not use the induction hypothesis. Equations (3.21) show that

$$\textstyle\sum_{j=1}^{n-1} w_{n,i,\,j}x_j + w_{n,i,\,n} = \theta_{n,i} - 1 \qquad \text{for all } i=1,\dots n;$$

since $w_{n,i,\,n}=-1$ for all $i=1,\dots n-1$, it follows that

$$\textstyle\sum_{j=1}^{n-1} w_{n,i,\,j}x_j = \theta_{n,i} \qquad \text{for all } i=1,\dots n-1,$$

and so $f_{n,i}(x_1,\dots,x_{n-1},0)=1$ for all $i=1,\dots,n-1$. By (ii) with $n-1$ in place of n, we have $f_{n-1,i}(x_1,\dots,x_{n-1})=1$ for all $i=1,\dots,n-1$.

Case 2: $(x_1,\dots,x_{n-1})$ *is not the alternating vector* $(\dots,1,0,1,0)$.
In this case, consider an arbitrary $(x_1,\dots,x_{n-1},0)$ in $\{0,1\}^n$ other than the alternating vector $(\dots,1,0,1,0)$. The induction hypothesis guarantees (alone if $x_{n-1}=0$ and combined with (i) if $x_{n-1}=1$) that $f_{n-1,i}(x_1,\dots,x_{n-1}) = f_{n-2,i}(x_1,\dots,x_{n-2})$ for all $i=1,\dots n-2$.

If $i \le n-2$ and $f_{n-1,i}(x_1,\ldots,x_{n-1}) = 1$, then $f_{n-2,i}(x_1,\ldots,x_{n-2}) = 1$, and so

$$\textstyle\sum_{j=1}^{n-1} w_{n,i,\,j}x_j = \sum_{j=1}^{n-1} w_{n-1,i,\,j}x_j + \sum_{j=1}^{n-2} w_{n-2,i,\,j}x_j \ge \theta_{n-1,i} + \theta_{n-2,i} > \theta_{n,i},$$

which implies $f_{n,i}(x_1,\ldots,x_{n-1},0) = 1$.

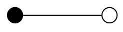
If $i \le n-2$ and $f_{n-1,i}(x_1,\ldots,x_{n-1}) = 0$, then $f_{n-2,i}(x_1,\ldots,x_{n-2}) = 0$, and so

$$\textstyle\sum_{j=1}^{n-1} w_{n,i,\,j}x_j = \sum_{j=1}^{n-1} w_{n-1,i,\,j}x_j + \sum_{j=1}^{n-2} w_{n-2,i,\,j}x_j \le (\theta_{n-1,i}-1) + (\theta_{n-2,i}-1) < \theta_{n,i},$$

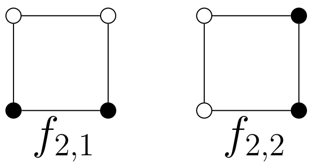
which implies $f_{n,i}(x_1,\ldots,x_{n-1},0) = 0$. $\square$

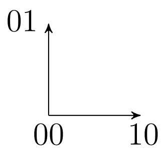
The first 4 networks are shown below:

$n = 1:$

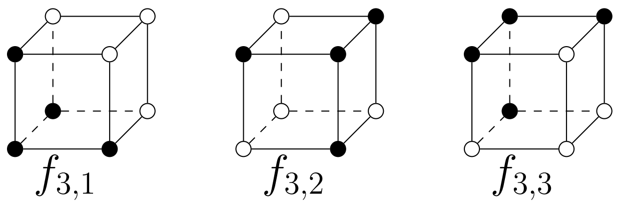


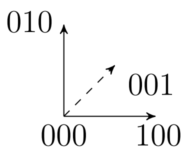
$n = 2:$



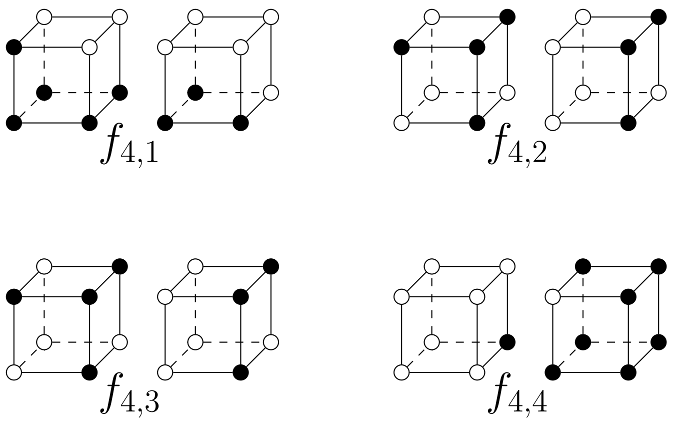


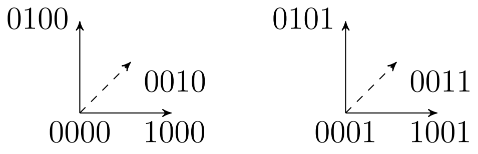
$n = 3:$





$n = 4:$





Theorem 3.4:
The neural network Φ_n is maximal.

Proof:
Straightforward induction on n, using self-duality and properties (3.17) and (3.18), proves a finer statement: The 2^n vectors $\Phi_n^t(0, 0, \ldots, 0)$ with $t = 0, 1, \ldots, 2^n - 1$ are pairwise distinct and $\Phi_n^t(0, 0, \ldots, 0)$ with $t = 2^n - 1$ is the alternating vector $(\ldots, 0, 1, 0, 1)$. □

Implicit in our proof of Theorem 3.4 is a simple way of transforming each trajectory

$$(0, 0, \ldots, 0) \to \Phi_n(0, 0, \ldots, 0) \to \cdots \to \Phi_n^{N-1}(0, 0, \ldots, 0) \tag{3.22}$$

with $N = 2^n$ into the trajectory

$$(0, 0, \ldots, 0) \to \Phi_{n+1}(0, 0, \ldots, 0) \to \cdots \to \Phi_{n+1}^{2N-1}(0, 0, \ldots, 0) : \tag{3.23}$$

First append 0 as the last bit to each point of the trajectory (3.22) and let T denote the resulting sequence of 2^n vectors $(x_1, \ldots, x_n, 0)$ in $\{0, 1\}^{n+1}$; then flip every bit $(0 \leftrightarrow 1)$ of every vector in T and let $\overline{T}$ denote the resulting sequence of 2^n vectors $(x_1, \ldots, x_n, 1)$ in $\{0, 1\}^{n+1}$; the trajectory (3.23) is the concatenation $T\overline{T}$. For instance, if $n = 3$, then (3.22) is

$$(0, 0, 0) \to (1, 0, 0) \to (1, 1, 0) \to (0, 1, 0) \to$$
$$(1, 1, 1) \to (0, 1, 1) \to (0, 0, 1) \to (1, 0, 1),$$

T is

$$\begin{aligned}&(0,0,0,0) \to (1,0,0,0) \to (1,1,0,0) \to (0,1,0,0) \to \\ &(1,1,1,0) \to (0,1,1,0) \to (0,0,1,0) \to (1,0,1,0),\end{aligned}$$

$\overline{T}$ is

$$\begin{aligned}&(1,1,1,1) \to (0,1,1,1) \to (0,0,1,1) \to (1,0,1,1) \to \\ &(0,0,0,1) \to (1,0,0,1) \to (1,1,0,1) \to (0,1,0,1),\end{aligned}$$

and (3.23) is

$$\begin{aligned}&(0,0,0,0) \to (1,0,0,0) \to (1,1,0,0) \to (0,1,0,0) \to \\ &(1,1,1,0) \to (0,1,1,0) \to (0,0,1,0) \to (1,0,1,0) \to \\ &(1,1,1,1) \to (0,1,1,1) \to (0,0,1,1) \to (1,0,1,1) \to \\ &(0,0,0,1) \to (1,0,0,1) \to (1,1,0,1) \to (0,1,0,1).\end{aligned}$$

Thus, if the n-dimensional network yields

$$\mathbf{y}^1 \to \mathbf{y}^2 \to \cdots \to \mathbf{y}^{2^{n-1}},$$

then the $(n+1)$-dimensional network yields

$$\mathbf{y}^1 0 \to \mathbf{y}^2 0 \to \cdots \to \mathbf{y}^{2^{n-1}} 0 \to \overline{\mathbf{y}^1} 1 \to \overline{\mathbf{y}^2} 1 \to \cdots \to \overline{\mathbf{y}^{2^{n-1}}} 1. \tag{3.24}$$

It may be interesting to note that Φ_n can be specified in yet another way. Every one-to-one mapping $r : \{0,1\}^n \to \{0,1,\ldots 2^n-1\}$ generates a mapping $\Phi : \{0,1\}^n \to \{0,1\}^n$ through the formula

$$\Phi(\mathbf{x}) = r^{-1}(r(\mathbf{x}) + 1 \bmod 2^n).$$

We have, for $\mathbf{s} = r^{-1}(\mathbf{0})$ and for all $t = 0,1,\ldots 2^n-1$,

$$\mathbf{x} = \Phi^t(\mathbf{s}) \Leftrightarrow t = r(\mathbf{x})$$

(this can be checked by straightforward induction on t); it follows that Φ has period 2^n. Our Φ_n is generated by the mapping $r_n : \{0,1\}^n \to \{0,1,\ldots 2^n-1\}$ defined by $r_n(x_1, x_2, \ldots x_n) = \sum_{j=1}^n c_j 2^{j-1}$ with

$$c_j = \begin{cases} |x_j - x_{j+1}| & \text{when } 1 \le j < n, \\ x_n & \text{when } j = n. \end{cases}$$

For instance,

$$\begin{aligned}
&r_4(0,0,0,0)=0,\ r_4(0,0,0,1)=12,\ r_4(0,0,1,0)=6,\ r_4(0,0,1,1)=10,\\
&r_4(0,1,0,0)=3,\ r_4(0,1,0,1)=15,\ r_4(0,1,1,0)=5,\ r_4(0,1,1,1)=9,\\
&r_4(1,0,0,0)=1,\ r_4(1,0,0,1)=13,\ r_4(1,0,1,0)=7,\ r_4(1,0,1,1)=11,\\
&r_4(1,1,0,0)=2,\ r_4(1,1,0,1)=14,\ r_4(1,1,1,0)=4,\ r_4(1,1,1,1)=8
\end{aligned}$$

This mapping r_n is one-to-one for every n: from the integer $r_n(x_1, x_2, \dots x_n)$, we can recover its binary encoding $(c_1, c_2, \dots c_n)$, from which we can recover first x_n, then x_{n-1}, and so on until x_1.

To see that $r_n(\Phi_n(\mathbf{x})) = r_n(\mathbf{x}) + 1 \bmod 2^n$, observe that (i) if x is the alternating vector $(\dots, 0, 1, 0, 1)$, then $r_n(\mathbf{x}) = 2^n - 1$ and (ii) for all other vectors $(x_1, x_2, \dots, x_n)$ in $\{0,1\}^n$, the largest subscript m such that $(x_1, x_2, \dots, x_m)$ is an alternating vector equals the smallest subscript m such that $c_m = 0$, in which case $r_n(x_1, x_2, \dots x_n) + 1 = \sum_{j=1}^{n} d_j 2^{j-1}$ with

$$d_j = \begin{cases} 0 & \text{when } 1 \le j < m, \\ 1 & \text{when } j = m, \\ c_j & \text{when } m < j \le n \end{cases}$$

and, with $(y_1, \dots y_n)$ defined by (3.16), we have $r_n(y_1, \dots y_n) = \sum_{j=1}^{n} d_j 2^{j-1}$.

Finally, we point out that even though the construction of weights and thresholds given here is recursive, the initial definition of the mapping Φ_n given in (3.16) provides a simple iterative method for calculating the successor of a state.

3.1.4 Maximal neural network isomorphisms

The three maximal neural networks described in the previous sections were all found independently of each other, and have not been studied together in a single work. In this section, we show that these three maximal neural are all isomorphic. We begin by defining isomorphism for maximal neural networks by considering mappings that permute coordinates and flip bits. A permutation is a one-to-one function $\pi : \{1, 2, \dots, n\} \to \{1, 2, \dots, n\}$, but we will abuse this notation by also allowing $\pi : \{0,1\}^n \to \{0,1\}^n$ to permute bit strings by writing $\pi(x_1, x_2, \dots, x_n) = \big(x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(n)}\big)$.

Definition 3.5:
Two maximal n-dimensional neural networks $\Phi : \{0,1\}^n \to \{0,1\}^n$ and $\Psi : \{0,1\}^n \to \{0,1\}^n$ are *isomorphic* if there exist

1. a time shift value s;
2. a permutation $\pi : \{0,1\}^n \to \{0,1\}^n$;
3. a bit flipping index set $S \subseteq \{1,2,\ldots,n\}$ and a function $f_S : \{0,1\}^n \to \{0,1\}^n$ of the form

$$\text{bit } i \text{ of } f_S(x_1,\ldots,x_n) \text{ is } \begin{cases} \overline{x_i} \text{ if } i \in S; \\ x_i \text{ if } i \notin S, \end{cases}$$

such that for all non-negative integers t and some initial seed $\mathbf{x}$ we have

$$\Phi^t(\mathbf{x}) = f_S(\pi(\Psi^{t+s}(\mathbf{x}))).$$

Theorem 3.5:
For every positive integer n, the maximal neural networks Φ_n and Ψ_n (described in Sections 3.1.1 and 3.1.3, respectively) are isomorphic.

Proof:
We produce an isomorphism using the identity permutation, a time shift of 1, and bit flipping set

$$S_n = \begin{cases} \{1,3,5,7,\ldots,n-1\} & \text{if } n \text{ is odd;} \\ \\ \{2,4,6,\ldots,n\} & \text{if } n \text{ is even.} \end{cases}$$

We proceed by induction on n, using an initial seed of 0^n for each n. Let $\Phi^t_{n,i}$ be the ith bit of 0^n under Φ_n at time t, and let $\Psi^t_{n,i}$ be the ith bit of 0^n under Ψ_n at time t. Note that the seed 0^n is implied in this notation. We will prove that for all t and $i = 1,2,\ldots,n$ we have

$$\Phi^t_{n,i} = \begin{cases} \Psi^{t+1}_{n,i} & \text{if } n-i \text{ is odd;} \\ \\ \overline{\Psi^{t+1}_{n,i}} & \text{if } n-i \text{ is even.} \end{cases}$$

Suppose n is even.

If $0 \le t < 2^{n-1}$ then we consider two cases according to the parity of i:

If i is odd then

$$\Phi^t_{n,i} = \Phi^t_{n-1,i} = \overline{\Psi^{t+1}_{n-1,i}} = \Psi^{t+1}_{n,i}.$$

The first and third equalities hold by how the trajectories (3.6) and (3.24) (on pages 34 and 47, respectively) are structured; the middle equality holds by the induction hypothesis.

Similarly, if i is even and $i \neq n$ then

$$\Phi^t_{n,i} = \Phi^t_{n-1,i} = \Psi^{t+1}_{n-1,i} = \overline{\Psi^{t+1}_{n,i}}.$$

If $2^{n-1} \leq t \leq 2^n$ then our two cases are:

If i is odd then

$$\Phi^t_{n,i} = \overline{\Phi^{t-2^{n-1}}_{n,i}} = \overline{\Phi^{t-2^{n-1}}_{n-1,i}} = \Psi^{t-2^{n-1}+1}_{n-1,i} = \overline{\Psi^{t-2^{n-1}+1}_{n,i}} = \Psi^{t+1}_{n,i}.$$

If i is even and $i \neq n$ then

$$\Phi^t_{n,i} = \overline{\Phi^{t-2^{n-1}}_{n,i}} = \overline{\Phi^{t-2^{n-1}}_{n-1,i}} = \overline{\Psi^{t-2^{n-1}+1}_{n-1,i}} = \Psi^{t-2^{n-1}+1}_{n,i} = \overline{\Psi^{t+1}_{n,i}}.$$

Once again, the middle equalities hold by induction, the rest hold by the structures of the trajectories (3.6) and (3.24).

Finally, trajectories (3.6) and (3.24) show that if $i = n$ then

$$\Phi^t_{n,n} = \overline{\Psi^{t+1}_{n,n}}.$$

This completes the proof when n is even. The case when n is odd is essentially the same. □

Although we have only defined isomorphism of maximal neural networks over the $\{0, 1\}$ symbol set, it is easy to see that Ω_n, defined in Section 3.1.2, and Φ_n are isomorphic. We can see this by replacing 1 with -1, and 0 with 1 in the trajectory given in (3.24); we get the same trajectory given by 3.11 and 3.12. Thus, the maximal neural networks Ψ_n, Ω_n, and Φ_n are all isomorphic.

In light of this result, it is natural to ask if *all* n-dimensional maximal neural networks are isomorphic. We will show that this is not the case. In order to do so, we will first show how Hamming distances can be used for certifying non-isomorphism between pairs of maximal neural networks.

Definition 3.6:
Let Φ be a neural network and let $h_t(\Phi, x)$ denote the Hamming distance between

$\Phi^t(\mathbf{x})$ and $\Phi^{t-1}(\mathbf{x})$, for $t = 1, 2, \ldots$. The *Hamming sequence* of a neural network Φ with seed $\mathbf{x}$ is

$$h_1(\Phi, \mathbf{x}), h_2(\Phi, \mathbf{x}), \ldots.$$

Lemma 3.5:
Let Φ and Ψ be isomorphic maximal neural networks. Then there exists a time shift value δ such that the Hamming sequences

$$h_1(\Phi, \mathbf{x}), h_2(\Phi, \mathbf{x}), h_3(\Phi, \mathbf{x}), \ldots \tag{3.25}$$

and

$$h_\delta(\Psi, \mathbf{x}), h_{\delta+1}(\Psi, \mathbf{x}), h_{\delta+2}(\Psi, \mathbf{x}), \ldots \tag{3.26}$$

are identical.

Proof:
Let Φ and Ψ be isomorphic maximal neural networks, with isomorphism given by s, π, S, as in Definition 3.5. For all t, the Hamming distance between $f_S(\pi(\Psi^{t+s}(\mathbf{x})))$ and $f_S(\pi(\Psi^{t+s-1}(\mathbf{x})))$ is equal to the Hamming distance between $\Psi^{t+s}(\mathbf{x})$ and $\Psi^{t+s-1}(\mathbf{x})$. Therefore, setting $\delta = s$ we get that (3.25) and (3.26) are the same. $\square$

Example 3.3:
The neural network with weight matrix

$$\begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & -1 \\ 1 & 1 & 1 \end{bmatrix}$$

and threshold vector

$$\begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

is maximal; it yields

$$000 \to 100 \to 010 \to 110 \to 111 \to 011 \to 101 \to 001 \to 000 \to \cdots.$$

Another 3-dimensional maximal neural network has weight matrix

$$\begin{bmatrix} -1 & -1 & 1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{bmatrix}$$

and threshold vector

$$\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix};$$

it yields

$$000 \to 100 \to 010 \to 001 \to 111 \to 011 \to 101 \to 110 \to 000 \to \cdots .$$

The Hamming sequence of the first trajectory is $1, 2, 1, 1, 1, 2, 1, 1, \ldots$ and that of the second is $1, 2, 2, 2, 1, 2, 2, 2, \ldots$. Thus, these networks are not isomorphic by Lemma 3.5.$\blacksquare$

3.1.5 Properties of maximal period networks

Now that we have affirmed the existence of maximal period neural networks, in this section we give some properties that all such networks must exhibit. In particular, the threshold functions in the maximal neural networks that we have constructed were all self-dual. The following shows that this is a necessary condition for a neural network to be maximal.

Lemma 3.6:
Let $\Phi = (f_1, f_2, \ldots, f_n)$ be a neural network with maximal period. Then the threshold functions $f_1, f_2, \ldots, f_n$ are self-dual.

Proof:
Fix j in $\{1, 2, \ldots, n\}$. Suppose that threshold function f_j has weights $w_1, w_2, \ldots, w_n$ and threshold θ, and suppose $(x_1^*, \ldots, x_n^*)$ is a point in $\{0,1\}^n$ such that

$$f_j(x_1^*, x_2^*, \ldots, x_n^*) = f_j(\overline{x_1^*}, \overline{x_2^*}, \ldots, \overline{x_n^*}) = 1.$$

Then we have

$$\sum_{i=1}^{n} w_i \geq 2\theta.$$

Therefore, for every $(x_1, x_2, \ldots, x_n)$ in $\{0,1\}^n$ we have

$$f_j(x_1, x_2, \ldots, x_n) = 1$$

or

$$f_j(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}) = 1,$$

or both. Thus, we have at least $2^{n-1} + 1$ points in $\{0,1\}^n$ for which f_j fires, but in order to traverse the entire space $\{0,1\}^n$, each threshold function f_j must have 2^{n-1} inputs for which it is 0 and 2^{n-1} inputs for which it is 1.

Now suppose that

$$f_j(x_1^*, x_2^*, \ldots, x_n^*) = f_j(\overline{x_1^*}, \overline{x_2^*}, \ldots, \overline{x_n^*}) = 0.$$

Then

$$\sum_{i=1}^{n} w_i < 2\theta.$$

Therefore, for every $(x_1, x_2, \ldots, x_n)$ in $\{0,1\}^n$ we must have either

$$f_j(x_1, x_2, \ldots, x_n) = 0$$

or

$$f_j(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}) = 0.$$

We now have $2^{n-1} + 1$ non-firing vertices, which is impossible for a maximal period sequence. □

Corollary 3.3:
Let $\Phi = (f_1, f_2, \ldots, f_n)$ be a neural network with maximal period and let $y(t)$ be the state of the network after t steps, so that $y(t) = \Phi^t(0, 0, \ldots, 0)$. Then for every non-negative integer t we have

$$\overline{y(t)} \to \overline{y(t+1)}.$$

Proof:
Fix i in $\{1, 2, \ldots, n\}$. Let $y_i(t)$ denote the ith bit of $y(t)$. Then

$$y_i(t+1) = f_i\left(y_1(t), y_2(t), \ldots, y_n(t)\right),$$

and thus by Lemma 3.6 we have

$$y_i(t+1) = \overline{f_i\left(\overline{y_1(t)}, \overline{y_2(t)}, \ldots, \overline{y_n(t)}\right)}$$

so that

$$\overline{y_i(t+1)} = f_i\left(\overline{y_1(t)}, \overline{y_2(t)}, \ldots, \overline{y_n(t)}\right).$$

This means the same thing as

$$\overline{y_i(t)} \to \overline{y_i(t+1)}$$

and the proof is complete since

$$\overline{y(t)} = (\overline{y_1(t)}, \overline{y_2(t)}, \ldots, \overline{y_n(t)}). \qquad \square$$

Corollary 3.4:
Let $\Phi = (f_1, f_2, \ldots, f_n)$ be a neural network with maximal period and let $y(t)$ be the state of the network after t steps. Then for every non-negative integer t we have

$$y(t + 2^{n-1}) = \overline{y(t)}.$$

Proof:
The proof is by induction on t. When $t = 0$ we argue as follows. Consider the segment

$$y(2^{n-1}) \to y(2^{n-1} + 1) \to \cdots \to y(2^n - 3) \to y(2^n - 2) \to y(2^n - 1) \to y(0).$$

By Corollary 3.3 we know that the following segment is also part of the trajectory:

$$\overline{y(2^{n-1})} \to \overline{y(2^{n-1} + 1)} \to \cdots \to \overline{y(2^n - 3)} \to \overline{y(2^n - 2)} \to \overline{y(2^n - 1)} \to \overline{y(0)}.$$

Both of these segments contain $2^{n-1} + 1$ distinct elements since we are assuming that the network is maximal. Therefore, by the pigeonhole principle there exist an i and j in $\{0, 1, \ldots, 2^{n-1}\}$ such that

$$y(2^{n-1} + i) = \overline{y(2^{n-1} + j)}. \tag{3.27}$$

We cannot have $i = j$ since an element cannot equal its complement. Furthermore, we may assume $i < j$ by taking complements of equation (3.27) if necessary. Equation (3.27) also yields

$$\overline{y(2^{n-1} + j)} \to y(2^{n-1} + i + 1),$$

and by Corollary 3.3 we get

$$y(2^{n-1} + j) \to \overline{y(2^{n-1} + i + 1)}.$$

We now have a cycle:

$$\overline{y(2^{n-1} + j)} \to y(2^{n-1} + i + 1) \to y(2^{n-1} + i + 2) \to \cdots \to y(2^{n-1} + j)$$

$$y(2^{n-1} + j) \to \overline{y(2^{n-1} + i + 1)} \to \overline{y(2^{n-1} + i + 2)} \to \cdots \to \overline{y(2^{n-1} + j)}.$$

Since the period is maximal we must have $j = 2^{n-1}$ and $i = 0$. By equation (3.27) we get

$$y(2^{n-1}) = \overline{y(2^{n-1}+2^{n-1})} = \overline{y(2^n)} = \overline{y(0)}.$$

This completes the proof for the base case. Suppose the claim holds up to and including time $t-1$. By the induction hypothesis we have

$$y(2^{n-1}+t-1) = \overline{y(t-1)}.$$

By definition,

$$y(2^{n-1}+t-1) \to y(2^{n-1}+t).$$

By Corollary 3.3 we have

$$\overline{y(t-1)} \to \overline{y(t)}.$$

Therefore,

$$y(2^{n-1}+t) = \overline{y(t)}. \qquad \square$$

Corollaries 3.3 and 3.4 show that the trajectories of maximal neural networks contain a lot of structure. This does not bode well for our goal of finding random-like neural network trajectories. In the next section we will see one more 'bad' property of maximal neural networks: there is always some relatively high correlation between some bits.

3.1.6 Cross-correlation

We will end this section on maximal neural networks by observing that the cross-correlation between some neurons in a maximal neural network is high. The idea of using cross-correlation as a device for showing that maximal neural networks are bad pseudorandom generators was proposed by Yori Zwols, and much of the work in this section was in collaboration with him. To make our proofs more concise, we switch to using $\{-1, 1\}$ as our symbol set. The main results of this section are Theorem 3.6 and Theorem 3.7.

Let $f : \{-1,1\}^n \to \{-1,1\}$ be a boolean function and consider the edges of the n-dimensional hypercube. These edges are classified into n *parallel classes:* an edge belongs to the j-th parallel class if its two endpoints differ in the j-th coordinate, and we call such an edge a j-edge. We call the edges of the hypercube *f-increasing* or *f-decreasing* or *f-constant:* an edge with endpoints u^-, u^+ such that

$$\begin{aligned} u^- &= (a_1, \ldots, a_{j-1}, -1, a_{j+1}, \ldots a_n) \\ u^+ &= (a_1, \ldots, a_{j-1}, +1, a_{j+1}, \ldots a_n) \end{aligned}$$

is called f-increasing if $f(u^-) < f(u^+)$, it is called f-decreasing if $f(u^-) > f(u^+)$, and it is called f-constant if $f(u^-) = f(u^+)$. When a single function f is under consideration, we will drop the $f-$ prefix and simply refer to the edges as *increasing*, *decreasing*, or *constant*.

Lemma 3.7:
Let f be an n-dimensional threshold function with weights $w_1, w_2, \ldots, w_n$. Then for $j = 1, 2 \ldots, n$ we have

$$w_j = 0 \implies \text{all edges in the } j\text{-th parallel class are constant;} \tag{3.28}$$
$$w_j > 0 \implies \text{all edges in the } j\text{-th parallel class are non-decreasing;} \tag{3.29}$$
$$w_j < 0 \implies \text{all edges in the } j\text{-th parallel class are non-increasing.} \tag{3.30}$$

Proof:
Let $g(x) = \sum_{i=1}^{n} w_i x_i$ and let $u = (u^-, u^+)$ be a j-edge. Then $g(u^+) = g(u^-) + 2w_j$, which proves claim (3.28). If $w_j > 0$ then $g(u^+) > g(u^-)$, which implies $f(u^+) \geq f(u^-)$. This proves claim (3.29). If $w_j < 0$ then $g(u^-) < g(u^+)$, which implies $f(u^-) \leq f(u^+)$. This proves claim (3.30). $\square$

For our next Lemma, we use the following notation:

Definition 3.7:
For a fixed n and $1 \leq k \leq n$, we define $r_k : \{-1, 1\}^n \to \{-1, 1\}^n$ as $r_k(\mathbf{x}) = (x_1, x_2, \ldots, x_{k-1}, \overline{x_k}, x_{k+1}, \ldots, x_n)$.

Lemma 3.8:
Let f be an n-dimensional threshold function with weights $w_1, w_2, \ldots, w_n$ and threshold θ. If $|w_j| \leq |w_k|$ and (u^-, u^+) is a non-constant j-edge, then

$$f\left(r_k(u^-)\right) = f\left(r_k(u^+)\right).$$

Proof:
Without loss of generality, we may assume that the j-th parallel class is increasing, and that (u^-, u^+) is an increasing j-edge. As in the previous proof, let $g(x) = \sum_{i=1}^{n} w_i x_i$. For $i = 1, 2, \ldots, n$ and $i \neq j$, let $x_i = u_i$. We have two cases to consider.

If $\operatorname{sgn}(w_k) = \operatorname{sgn}(x_k)$ then

$$\begin{aligned} g(r_k(u^+)) &= w_j - w_k x_k + \sum_{i \neq j,k} w_i x_i = 2w_j - 2w_k x_k + \sum_{i \neq j,k} w_k x_k \\ &= 2w_j - 2w_k x_k + g(u^-) < 2(w_j - w_k x_k) + \theta < \theta. \end{aligned}$$

Therefore, $-1 = f(r_k(u^+)) = f(r_k(u^-))$ since (u^-, u^+) is increasing in the j-th parallel class.

If $\operatorname{sgn}(w_k) \neq \operatorname{sgn}(x_k)$ then

$$g(r_k(u^-)) = -w_j - w_k x_k + \sum_{i \neq j,k} w_i x_i = -2(w_j + w_k x_k) + g(u^+) \geq \theta.$$

Therefore, $1 = f(r_k(u^-)) = f(r_k(u^+))$ since (u^-, u^+) is increasing in the j-th parallel class. □

Turning to the case of self-dual threshold functions, we observe that we may assume a zero threshold:

Lemma 3.9:
Let f be an n-dimensional self-dual threshold function with weights $w_1, w_2, \ldots, w_n$ and threshold θ, and let g be the n-dimensional threshold function with weights $w_1, w_2, \ldots, w_n$ and threshold 0. *Then $g(x) = f(x)$ for all x in $\{-1,1\}^n$.*

Proof:
Since f is self-dual, we have $\sum_{i=1}^n w_i x_i \neq \theta$ for all x in $\{-1,1\}^n$. Furthermore,

$$\begin{aligned}
\sum_{i=1}^n w_i x_i > \theta \quad &\Longrightarrow \quad -\sum_{i=1}^n w_i x_i < \theta \quad \text{by self-duality} \\
&\Longrightarrow \quad \sum_{i=1}^n w_i x_i > -\theta \\
&\Longrightarrow \quad 2\sum_{i=1}^n w_i x_i > 0 \\
&\Longrightarrow \quad \sum_{i=1}^n w_i x_i > 0.
\end{aligned}$$

Similarly,

$$\begin{aligned}
\sum_{i=1}^n w_i x_i < \theta \quad &\Longrightarrow \quad -\sum_{i=1}^n w_i x_i > \theta \quad \text{by self-duality} \\
&\Longrightarrow \quad \sum_{i=1}^n w_i x_i < -\theta \\
&\Longrightarrow \quad 2\sum_{i=1}^n w_i x_i < 0 \\
&\Longrightarrow \quad \sum_{i=1}^n w_i x_i < 0.
\end{aligned}$$

□

A common notion in the analysis of boolean functions [61] is that of influence:

Definition 3.8:
For a boolean function $f : \{-1,1\}^n \to \{-1,1\}$, the *influence* of the ith coordinate, $I_i(f)$, is defined as the number of points x in $\{-1,1\}^n$ such that $f(x) \neq f(r_i(x))$.

An asymptotic version of the following Lemma has been proved using discrete Fourier analysis in [24]. Here we prove the result from first principles.

Lemma 3.10:
If f is an n-dimensional self-dual threshold function, then there exists a coordinate i such that

$$I_i(f) \geq \binom{n-1}{\lfloor \frac{n-1}{2} \rfloor}.$$

Proof:
Without loss of generality, suppose $\max_i\{|w_i|\} = |w_n|$ and $w_n > 0$. By Lemma 3.8, every non-constant j-edge yields a non-constant k-edge. Therefore, for all j and k in $\{1, 2, \ldots, n\}$, if $|w_j| \leq |w_k|$ then $I_j(f) \leq I_k(f)$. Thus, the coordinate with largest influence is coordinate n. Furthermore, $I_n(f)$ is equal to the number of x in $\{-1,1\}^{n-1}$ such that

$$\sum_{i=1}^{n-1} w_i x_i + w_n > 0$$

and

$$\sum_{i=1}^{n-1} w_i x_i - w_n < 0,$$

which is the number of x in $\{-1,1\}^{n-1}$ such that

$$-w_n < \sum_{i=1}^{n-1} w_i x_i < w_n.$$

Another way to count this is to set $v_i = w_i/w_n$ for $i = 1, \ldots, n-1$, and to count the number of x in $\{-1,1\}^{n-1}$ such that

$$-1 < \sum_{i=1}^{n-1} v_i x_i < 1,$$

where $|v_i| \leq 1$. An unpublished result by Sárközy and Szemerédi [5] says that the number of such x is at least $\binom{n-1}{\lfloor \frac{n-1}{2} \rfloor}$. $\square$

Definition 3.9:
Let $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(p), \mathbf{x}(1), \ldots$ be a sequence of n-dimensional bit vectors with period p. We define *the cross-correlation (with time lag 1) between x_i and x_j* to be $C(x_i, x_j) = \frac{1}{p}\sum_{t=1}^{p} x_j(t)x_i(t+1)$.

Lemma 3.11:
Let $f_1, \ldots, f_n$ be an n-dimensioanl maximal neural network producing the trajectory $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(2^n), \mathbf{x}(1), \ldots$. *For every pair of integers i and j in $\{1, \ldots, n\}$ we have*

$$C(x_i, x_j) = \frac{\pm I_j(f_i)}{2^n}.$$

Proof:
By definition,

$$C(x_i, x_j) = \frac{1}{2^n}\sum_{t=1}^{2^n} x_j(t)x_i(t+1).$$

Since the network cycles through all 2^n states, the right hand side can be written as

$$\frac{1}{2^n}\sum_{\mathbf{x}\in\{-1,1\}^n} x_j f_i(\mathbf{x}).$$

Instead of summing over all $\mathbf{x}$ in $\{-1,1\}^n$, we can sum over all j-edges u, so that

$$C(x_i, x_j) = \frac{1}{2^n}\sum_{u}\left[-f_i(u^-) + f_i(u^+)\right].$$

The quantity $-f_i(u^-) + f_i(u^+)$ is non-zero if and only if u is a j-edge that is not f-constant; it is 2 if the j-th parallel class is f-increasing, and -2 if it is f-decreasing.□

Theorem 3.6:
The output $\mathbf{x}(1), \mathbf{x}(2), \ldots$ *of a maximal neural network always contains two coordinates i and j such that*

$$|C(x_i, x_j)| \geq \binom{n-1}{\lfloor\frac{n-1}{2}\rfloor}/2^n. \tag{3.31}$$

Proof:
This follows from Lemma 3.10 and Lemma 3.11. □

In order to get a sense of the implications of this result, we consider the 'truly' random case:

Theorem 3.7:
Let $X = (X_1, \dots, X_{2^n})$ and $Y = (Y_1, \dots, Y_{2^n})$, where each X_i and Y_i are independent random variables taking on -1 and $+1$, each with probability $1/2$. Then

$$\mathbf{P}\left(\left|C(X,Y)\right| \geq \frac{\binom{n-1}{\lfloor \frac{n-1}{2} \rfloor}}{2^n} \right) \leq \frac{1}{2^{2^n - n^2 - n}}.$$

Proof:
We wish to evaluate $(1/2^n)|\sum_{i=1}^{2^n} X_i Y_i|$. We do this by writing $Z_i = X_i Y_i$. Now Z_i is also random variable taking on -1 and $+1$, each with probability $1/2$, and $|\sum_{i=1}^{2^n} Z_i|$ can be seen as the distance from the origin after 2^n steps of a random walk starting at 0. Letting $m = 2^{n-1}$, the expected value of this distance can be computed as follows:

$$\begin{aligned}
\mathbf{E}\left|\sum_{i=1}^{2m} Z_i\right| &= 2m\frac{\binom{2m}{0}}{2^{2m}} + (2m-2)\frac{\binom{2m}{1}}{2^{2m}} + \cdots + 0\frac{\binom{2m}{m}}{2^{2m}} + \cdots + (2m-2)\frac{\binom{2m}{2m-1}}{2^{2m}} + 2m\frac{\binom{2m}{2m}}{2^{2m}} \\
&= \frac{2}{2^{2m}} \sum_{k=0}^{2m} |m-k| \binom{2m}{k} \\
&= \frac{2}{2^{2m}}\, 2 \sum_{k=0}^{m} (m-k) \binom{2m}{k} \\
&= \frac{2}{2^{2m}} \left(2m \sum_{k=0}^{m} \binom{2m}{k} - 2 \sum_{k=0}^{m} k \binom{2m}{k} \right) \\
&= \frac{2}{2^{2m}} \left(m \left(2^{2m} + \binom{2m}{m} \right) - 2\, m\, 2^{2m-1} \right) \\
&= \frac{2}{2^{2m}}\, m \binom{2m}{m} \\
&= \frac{2^n}{2^{2^n}} \binom{2^n}{2^{n-1}}.
\end{aligned}$$

Therefore,

$$\mathbf{E}|C(X,Y)| = \frac{1}{2^n}\mathbf{E}\left|\sum_{i=1}^{2m} Z_i\right| = \frac{\binom{2^n}{2^{n-1}}}{2^{2^n}} \leq \frac{2^n!}{2^{2^n}} \leq \frac{(2^n)^n}{2^{2^n}} = \frac{2^{n^2}}{2^{2^n}}.$$

Therefore, by Markov's inequality we get

$$\mathbf{P}\left(\left|C(X,Y)\right| \geq \frac{\binom{n-1}{\lfloor\frac{n-1}{2}\rfloor}}{2^n}\right) \leq \frac{2^n\mathbf{E}|C(X,Y)|}{\binom{n-1}{\lfloor\frac{n-1}{2}\rfloor}} \leq 2^n\mathbf{E}|C(X,Y)| \leq \frac{1}{2^{2^n-n^2-n}}. \quad \square$$

Even with this loose upper bound, we see that the probability of getting such a large cross-correlation is extremely rare in a random case.

3.2 Long period networks

The properties given in Sections 3.1.5 and 3.1.6 show that neural networks with maximal period may contain too much structure to be "random-like". For this reason, we will now describe some neural networks that have very long, but not maximal, periods.

3.2.1 Shift register networks

In this section we will borrow from the theory of *shift register sequences*, and prove a new result stating that they can be implemented on a neural network. Shift register sequences are sometimes referred to as *linear feedback shift register sequences* or *Tausworthe sequences* [71]. Suffice it here to give a brief outline of the theory. In Appendix B the theory is presented in greater detail.

The idea of implementing shift registers sequences via a neural network was inspired by Pierre L'Ecuyer. In a private communication, he suggested that neural networks may be capable of implementing $\mathbf{F}_2$-generators [40], a class of generators that does indeed include the shift register generators described next.

Definition 3.10:
Let n be a positive integer and let $a_1, a_2, \ldots, a_{n-1}$ be in $\{0, 1\}$. A sequence $x_0, x_1, x_2, \ldots$ satisfying

$$x_{t+n} = a_{n-1}x_{t+n-1} + a_{n-2}x_{t+n-2} + \cdots + a_1x_{t-1} + x_t \pmod 2 \qquad (3.32)$$

is an *n-th order linear recurrence sequence* (mod 2). The values $x_0, x_1, \ldots, x_{n-1}$ are the *seeds* of the sequence.

Example 3.4:
The linear recurrence $x_{t+3} = x_{t+2} + x_t \pmod 2$ with seed $x_0 = x_1 = x_2 = 1$ yields the sequence 11101001110100111010 0..., which has period 7. ■

For every n-th order linear recurrence sequence obeying

$$x_{t+n} = a_{n-1}x_{t+n-1} + a_{n-2}x_{t+n-2} + \cdots + a_1 x_{t-1} + x_t \pmod 2,$$

we associate an $n \times n$ matrix of bits defined by

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \cdots & 0 \\ & & & \vdots & & & \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & a_1 & a_2 & \cdots & a_{n-3} & a_{n-2} & a_{n-1} \end{bmatrix}.$$

We shall refer to A as the *recurrence matrix* of the associated linear recurrence sequence. Note that the *state vectors* $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ defined by

$$\mathbf{x}_t = \begin{pmatrix} x_t \\ x_{t+1} \\ \cdots \\ x_{t+n-1} \end{pmatrix},$$

obey

$$\mathbf{x}_{t+1} = A\mathbf{x}_t,$$

for $t = 0, 1, \ldots$.

Example 3.5:
The recurrence matrix of the linear recurrence $x_{t+3} = x_{t+2} + x_t \pmod 2$ from Example 3.4 is

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}.$$

Starting with state vector $\mathbf{x}_0 = 111$, the state vector trajectory is

$$\begin{array}{c}
1\,1\,1 \\
1\,1\,0 \\
1\,0\,1 \\
0\,1\,0 \\
1\,0\,0 \\
0\,0\,1 \\
0\,1\,1 \\
1\,1\,1 \\
1\,1\,0 \\
1\,0\,1 \\
0\,1\,0 \\
1\,0\,0 \\
0\,0\,1 \\
0\,1\,1 \\
\vdots\ .
\end{array}$$

■

The previous example can be used to illustrate where the term *shift register sequence* comes from. Imagine an n-bit register filled with the initial seed $x_0, x_1, \ldots, x_{n-1}$. To get the next state, the contents of the register are shifted left; the previous first bit is lost; the new last bit is the *xor* of the bits defined by the recurrence relation (the first and last bit in the example above).

The characteristic polynomial of a square matrix M is defined by $\chi_M(z) = \det(zI - M)$. For a recurrence matrix

$$A = \begin{bmatrix}
0 & 1 & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & \cdots & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & \cdots & 0 \\
 & & & \vdots & & & \\
0 & 0 & 0 & \cdots & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & \cdots & 0 & 1 \\
1 & a_1 & a_2 & \cdots & a_{n-3} & a_{n-2} & a_{n-1}
\end{bmatrix},$$

its characteristic polynomial χ_A is given by $\chi_A(z) = z^n - a_{n-1}z^{n-1} - a_{n-2}z^{n-2} - \cdots - a_1 - 1$. Thus, for an n-th order linear recurrence relation $x_{t+n} = a_{n-1}x_{t+n-1} + a_{n-2}x_{t+n-2} + \cdots + a_1x_{t-1} + x_t \pmod 2$, we refer to $z^n - a_{n-1}z^{n-1} - a_{n-2}z^{n-2} - \cdots - a_1 - 1$ as the characteristic polynomial of the recurrence.

Since the zero state is always mapped to itself in a shift register sequence, an upper bound on the period of an n-order linear recurrence is $2^n - 1$, and indeed this upper bound is tight. When does a shift register sequence with register size n have period 2^n-1? This is answered in Theorem B.1 of Appendix B: if the characteristic polynomial $z^n - a_{n-1}z^{n-1} - a_{n-2}z^{n-2} - \cdots - a_1 - 1$ is *primitive* over $\mathbf{F}_2$, then the n-th order linear recurrence relation $x_{t+n} = a_{n-1}x_{t+n-1} + a_{n-2}x_{t+n-2} + \cdots + a_1 x_{t-1} + x_t \pmod 2$ has period $2^n - 1$ for all seeds except the one consisting of all zeros.

We are almost ready to build neural networks based on shift register sequences. In doing so, we shall focus on primitive *trinomials* with the form $z^n + z^m + 1$, where $1 < m < n$. Focusing on primitive polynomials that are trinomials when implementing pseudorandom number generators is often done for speed; the lack of extra non-zero coefficients allows for more efficient implementations (see [40] and [39], for example). However, we are not concerned with efficiency here. The advantage of trinomials for us is that they will yield neural networks with fewer neurons and connections, thus making them easier to describe; the disadvantage is that they have some known statistical defects ([52], [54], [40]). Tables of primitive trinomials have been computed in [81].

Before moving on to the main theorem, we will warm up with some observations. The main hurdle in implementing a linear recurrence modulo 2 is computing the *xor* function. As we saw in example 2.4 on page 15, this function is not threshold. However, since $x_1 \oplus x_2 = (x_1 \vee x_2)(\overline{x_1 x_2})$, a neural network can compute $x_1 \oplus x_2$ in two steps:

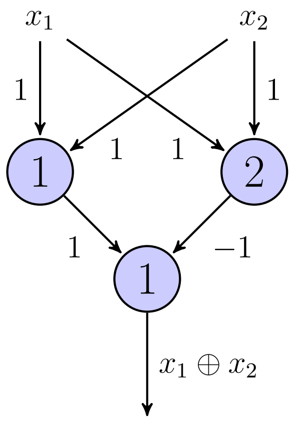


Figure 3.1: A neural network that computes $x_1 \oplus x_2$ after two time steps.

The trick to avoiding the two step time delay is to take advantage of the simple shifting structure of the linear recurrence we are generating. We do this by sampling the bits two steps earlier than we really need them for the $\oplus$ function. We will see this more precisely in the proof of Theorem 3.8.

Let us now describe how to construct a neural network for the primitive trinomial

$z^n + z^m + 1$. We shall construct an $(n+2)$-dimensional neural network Ψ consisting of neurons $f_1, f_2, \ldots, f_n, f_{n+1}, f_{n+2}$.

The first n neurons will mimic the underlying shift register, and the last two will function as auxiliary neurons, which will be used to calculate the *xor* for the special nth bit. Only the last two neurons will depend on the parameter m.

We construct our network in four different sections; we first construct neurons $f_1, f_2, \ldots, f_{n-1}$, then the construction of f_n, f_{n+1}, and f_{n+2} are handled separately.

We define the neurons $f_1, f_2, \ldots, f_{n-1}$ by

$$w_{i,j} = \begin{cases} 1 & \text{for } i = 1, \ldots, n-1, \text{ and } j = i+1; \\ 0 & \text{for } i = 1, \ldots, n-1, \text{ and } j \neq i+1 \end{cases}$$

and

$$\theta_i = 1 \qquad \text{for } i = 1, \ldots, n-1.$$

Neuron f_n is defined by

$$w_{n,j} = \begin{cases} 1 & \text{if } j = n+2; \\ 0 & \text{otherwise ;} \end{cases}$$

and

$$\theta_n = 1.$$

Neuron f_{n+1} is defined by

$$w_{n+1,j} = \begin{cases} 1 & \text{if } j = 3 \text{ or } j = m+2; \\ 0 & \text{otherwise ;} \end{cases}$$

and

$$\theta_{n+1} = 2.$$

Finally, neuron f_{n+2} is defined by

$$w_{n+2,j} = \begin{cases} 1 & \text{if } j = 2 \text{ or } j = m+1; \\ -2 & \text{if } j = n+1; \\ 0 & \text{otherwise ;} \end{cases}$$

and

$$\theta_{n+2} = 2.$$

To summarize, the weight matrix of the neural network is

$$\begin{array}{c|cccccccccccccc|}
 & 1 & 2 & 3 & 4 & \ldots & m+1 & m+2 & m+3 & \cdots & & n & n+1 & n+2 \\
\hline
1 & 0 & 1 & 0 & 0 & & & & & & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 0 & & & & & & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 1 & & & & & & 0 & 0 & 0 & 0 \\
\vdots & & & & & \ddots & & & & & & & & \\
m+1 & 0 & 0 & 0 & 0 & & & 1 & & & & & & \\
m+2 & 0 & 0 & 0 & 0 & & & & 1 & & & & & \\
\vdots & & & & & & & & & \ddots & & & & \\
n-1 & 0 & 0 & 0 & 0 & & & & & & 0 & 1 & 0 & 0 \\
n & 0 & 0 & 0 & 0 & & 0 & 0 & & & 0 & 0 & 0 & 1 \\
n+1 & 0 & 0 & 1 & 0 & & 0 & 1 & & & 0 & 0 & 0 & 0 \\
n+2 & 0 & 1 & 0 & 0 & & 1 & 0 & & & 0 & 0 & -2 & 0
\end{array},$$

and the $(n+2)$-dimensional threshold vector is

$$\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 2 \\ 1 \end{bmatrix}.$$

For a seed $\mathbf{x}$ in $\{0,1\}^{n+2}$, we have

$$\Psi_i^t(\mathbf{x}) = \Psi_{i+1}^{t-1}(\mathbf{x}) \qquad \text{for } i = 1, 2, \ldots, n-1; \tag{3.33}$$

$$\Psi_n^t(\mathbf{x}) = \Psi_{n+2}^{t-1}(\mathbf{x}); \tag{3.34}$$

$$\Psi_{n+1}^t(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \Psi_3^{t-1}(\mathbf{x}) + \Psi_{m+2}^{t-1}(\mathbf{x}) \geq 2; \\ 0 & \text{otherwise,} \end{cases} \tag{3.35}$$

and

$$\Psi_{n+2}^t(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \Psi_2^{t-1}(\mathbf{x}) + \Psi_{m+1}^{t-1}(\mathbf{x}) - 2\Psi_{n+1}^{t-1}(\mathbf{x}) \geq 1; \\ 0 & \text{otherwise.} \end{cases} \tag{3.36}$$

The stage is now set and we can prove that Ψ behaves as a shift register sequence when the auxiliary neurons are initialized in a particular manner.

Theorem 3.8:
Let $z^n + z^m + 1$ be a primitive trinomial and let Ψ be the $(n+2)$-dimensional neural network described above. Then there are at least $2^n - 1$ seeds that have period $2^n - 1$ under Ψ.

Proof:
The trinomial $z^n + z^m + 1$ is the characteristic polynomial of the n-th order linear recurrence

$$x_{t+n} = x_{t+m} + x_t \pmod 2. \tag{3.37}$$

Let $\mathbf{x}^0 = (x_1, x_2, \ldots, x_n)$ be a non-zero n-dimensional seed for the linear recurrence (3.37), and let the state vectors of the recurrence be $\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \ldots$. We augment the seed $\mathbf{x}^0$ to an $(n+2)$-dimensional seed $\mathbf{s} = (s_1, s_2, \ldots, s_{n+2})$ for our neural network by setting

$$s_i = \begin{cases} x_i & \text{for} \quad i = 1, 2, \ldots, n; \\ x_2 x_{m+1} & \text{if} \quad i = n+1; \\ x_1 \oplus x_m & \text{if} \quad i = n+2. \end{cases} \tag{3.38}$$

By induction on t, we will show that the following hold:

$$\Phi_i^t(\mathbf{s}) = \mathbf{x}_i^t \qquad 1 \leq i \leq n; \tag{3.39}$$

$$\Phi_{n+1}^t(\mathbf{s}) = \mathbf{x}_2^t \mathbf{x}_{m+1}^t; \tag{3.40}$$

$$\Phi_{n+2}^t(\mathbf{s}) = \mathbf{x}_1^t \oplus \mathbf{x}_m^t. \tag{3.41}$$

When $t = 0$, this claim follows from how the seed $\mathbf{s}$ was defined in (3.38). Suppose this claim holds with $t-1$ in place of t. We will first show that (3.39) holds. When $1 \leq i \leq n-1$, we have $\Phi_i^t(\mathbf{s}) = \Phi_{i+1}^{t-1}(\mathbf{s})$ by (3.33), and $\Phi_{i+1}^{t-1}(\mathbf{s}) = \mathbf{x}_{i+1}^{t-1}$ by the induction hypothesis. By the linear recurrence (3.37), we have $\mathbf{x}_{i+1}^{t-1} = \mathbf{x}_i^t$. For the nth bit, we have $\Phi_n^t(\mathbf{s}) = \Phi_{n+2}^{t-1}(\mathbf{s})$ by (3.34). By the induction hypothesis and linear recurrence (3.37), we get $\Phi_{n+2}^{t-1}(\mathbf{s}) = \mathbf{x}_1^{t-1} \oplus \mathbf{x}_m^{t-1} = \mathbf{x}_n^t$.

To see that (3.40) holds, we have

$$\begin{aligned}
\Phi_{n+1}^{t}(\mathbf{s}) &= \Phi_{3}^{t-1}(\mathbf{s})\Phi_{m+2}^{t-1}(\mathbf{s}) && \text{(by (3.35))}\\
&= \mathbf{x}_{3}^{t-1}\mathbf{x}_{m+2}^{t-1} && \text{(by the induction hypothesis)}\\
&= \mathbf{x}_{2}^{t}\mathbf{x}_{m+1}^{t}. && \text{(by linear recurrence (3.37))}
\end{aligned}$$

Finally, (3.41) holds because

$$\begin{aligned}
\Phi_{n+2}^{t}(\mathbf{s}) &= \left(\Phi_{2}^{t-1}(\mathbf{s}) \vee \Phi_{m+1}^{t-1}(\mathbf{s})\right)\left(\overline{\Phi_{n+1}^{t-1}}\right) && \text{(by (3.36))}\\
&= \left(\mathbf{x}_{2}^{t-1} \vee \mathbf{x}_{m+1}^{t-1}\right)\left(\overline{\mathbf{x}_{2}^{t-1}\mathbf{x}_{m+1}^{t-1}}\right) && \text{(by the induction hypothesis)}\\
&= \left(\mathbf{x}_{1}^{t} \vee \mathbf{x}_{m}^{t}\right)\left(\overline{\mathbf{x}_{1}^{t}\mathbf{x}_{m}^{t}}\right) && \text{(by linear recurrence (3.37))}\\
&= \mathbf{x}_{1}^{t} \oplus \mathbf{x}_{m}^{t}. && \square
\end{aligned}$$

Example 3.6:
The primitive trinomial $x^4 + x + 1$ is the characteristic polynomial of the linear recurrence

$$x_{t+4} = x_{t+1} + x_t \pmod 2.$$

The seed $\mathbf{x}^0 = 1111$ yields the following sequence:

$$\begin{array}{c} 1\,1\,1\,1 \\ 1\,1\,1\,0 \\ 1\,1\,0\,0 \\ 1\,0\,0\,0 \\ 0\,0\,0\,1 \\ 0\,0\,1\,0 \\ 0\,1\,0\,0 \\ 1\,0\,0\,1 \\ 0\,0\,1\,1 \\ 0\,1\,1\,0 \\ 1\,1\,0\,1 \\ 1\,0\,1\,0 \\ 0\,1\,0\,1 \\ 1\,0\,1\,1 \\ 0\,1\,1\,1 \\ 1\,1\,1\,1 \\ \vdots \end{array},$$

which has period $2^4 - 1$. The corresponding 6-dimensional neural network has weight matrix

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & -2 & 0 \end{bmatrix}$$

and threshold vector

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 2 \\ 1 \end{bmatrix}.$$

The network looks like:

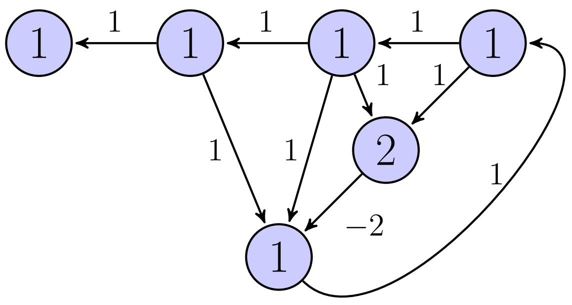


Figure 3.2: The neural network associated with $x^4 + x + 1$.

Beginning with the seed 1111, we augment it to a 6-bit seed as follows: the fifth bit is the logical *and* of the second and third bit; the sixth bit is the *xor* of first and second bit. Thus, the augmented seed is 111110, and the neural network produces

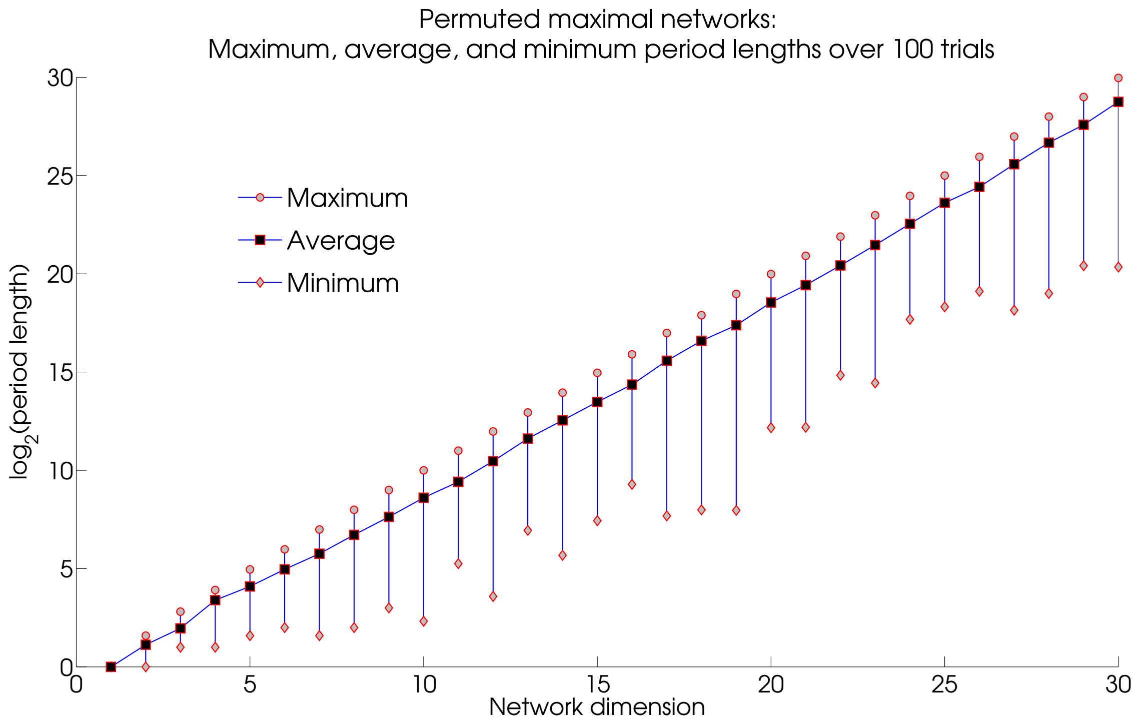
$$\begin{array}{c}
1\,1\,1\,1\,1\,0\\
1\,1\,1\,0\,1\,0\\
1\,1\,0\,0\,0\,0\\
1\,0\,0\,0\,0\,1\\
0\,0\,0\,1\,0\,0\\
0\,0\,1\,0\,0\,0\\
0\,1\,0\,0\,0\,1\\
1\,0\,0\,1\,0\,1\\
0\,0\,1\,1\,0\,0\\
0\,1\,1\,0\,1\,1\\
1\,1\,0\,1\,0\,0\\
1\,0\,1\,0\,0\,1\\
0\,1\,0\,1\,0\,1\\
1\,0\,1\,1\,0\,1\\
0\,1\,1\,1\,1\,1\\
1\,1\,1\,1\,1\,0\\
\vdots
\end{array},$$

which also has period $2^4 - 1$. ■

3.2.2 Random permutations of maximal period networks

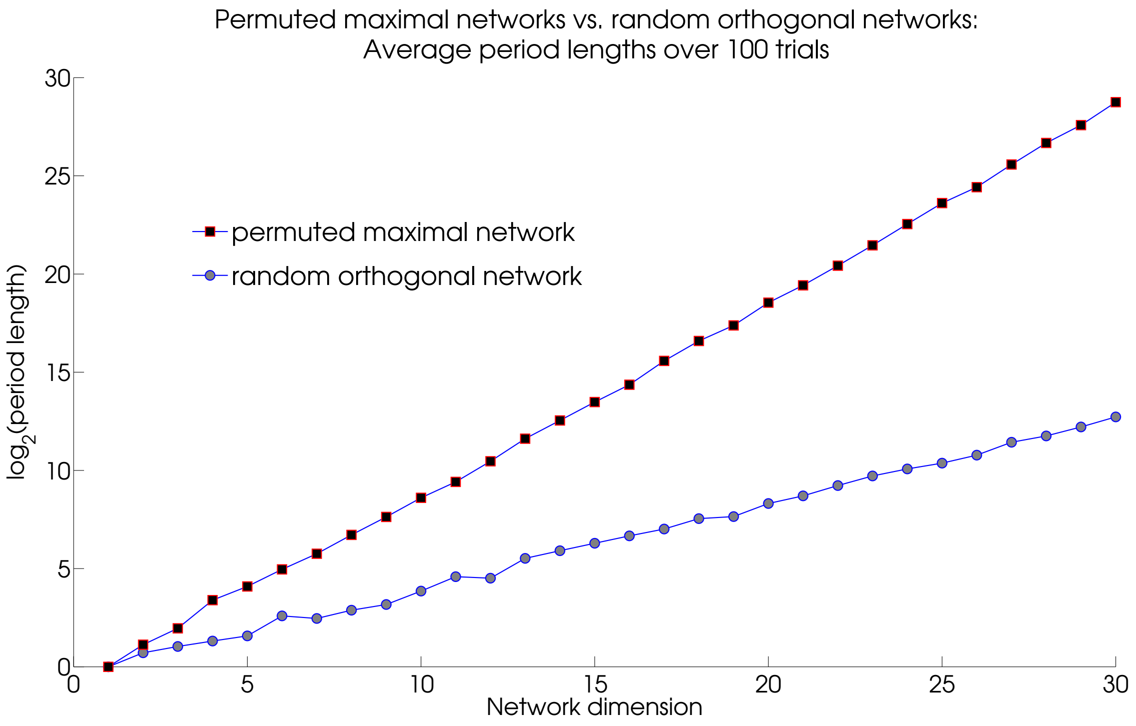
The maximal neural networks of Section 3.1.3 can be altered as follows to produce neural networks with some long, but not necessarily maximal, trajectories. Consider the n-dimensional maximal neural network described in Section 3.1.3 with weight matrix W_n and threshold vector θ_n. Let P_n be an $n \times n$ permutation matrix (an $n \times n$ matrix with entries in $\{0, 1\}$ where every row and every column has precisely one 1 entry). This yields a new neural network with weight matrix $P_n W_n$ and threshold vector $P_n \theta_n$. We are essentially permuting the order of the n threshold functions of the original maximal neural network. It turns out that these permutations yield trajectories that are often still very long, as the following experiment demonstrates.

For a fixed dimension n, we generate an $n \times n$ permutation matrix P_n by drawing uniformly from the $n!$ permutation matrices. This gives us a new neural network with weight matrix $P_n W_n$ and threshold vector $P_n \theta_n$. Running this network with a randomly generated seed, we compute the period of the trajectory. This was done 100 times for each n in $\{1, 2, \ldots, 30\}$. The results are summarised below:



3.2.3 Random orthogonal weight matrices

In [15], Elyada and Horn conducted statistical experiments and proposed the following neural networks for pseudorandom number generation: for a fixed dimension n, let W be an $n \times n$ matrix, where each entry is drawn randomly from the standard normal distribution. This matrix W is then orthogonalized to W' using the Gram–Schmidt process. The threshold vector θ is set so that its ith entry is the sum of the elements of the ith row of W', divided by 2. This is done to self-dualize the threshold functions (see Theorem 2.3 on page 20). We reproduced some of their experiments, along the same lines of Section 3.2.2. A comparison of the two experiments is shown below:



Chapter 4

Randomness and Pseudorandomness

In this chapter we measure the ability of neural networks to behave randomly. The chapter is divided into two sections; the first is about theoretical pseudorandom number generation, and the second will describe the results of a battery of statistical tests applied to some of the neural networks described in the previous chapter.

4.1 Computational indistinguishability

The notion of computational indistinguishability stems from ideas in computational complexity theory. It was first found in the work of Yao [80] and that of Goldwasser and Micali [22]. The main idea is that two objects are considered to be similar if no efficient algorithm can tell them apart. In the context of pseudorandomness, a pseudorandom number generator is a deterministic function that, with high probability, cannot be distinguished from a uniform random variable by any efficient algorithm. We follow the treatment in [18]. Another good reference is [33].

Definition 4.1:
Let I be a countable index set. A *probability ensemble indexed by I* is a sequence of random variables indexed by I.

The notion of a *computational distinguisher* uses an efficient algorithm D, typically taking pseudorandom candidates as input and having boolean values as output, according to whether or not D accepts the candidate. More precisely,

Definition 4.2:
Two probability ensembles $X = X_1, X_2, \ldots$ and $Y = Y_1, Y_2, \ldots$ are *computationally indistinguishable* if for every probabilistic polynomial-time algorithm D, every positive integer k, and all sufficiently large n we have

$$\left| P\Big(D(X_n, 1^n) = 1\Big) - P\Big(D(Y_n, 1^n) = 1\Big) \right| < \frac{1}{n^k}.$$

Typically, for all i the random variables X_i and Y_i are multi-variate random variables of dimension i. The 1^n input to D is a technical safety precaution to ensure that we are considering all algorithms D with running time polynomial in n, even in the atypical case where $|X_n| = O(poly(n))$ but $poly(n) \neq O(|X_n|)$, where $|X_n|$ denotes the dimension of the multi-variate random variable X_n.

Probability ensembles that are of particular interest are the *uniform ensembles*:

Definition 4.3:
Let $l : \mathbf{N} \to \mathbf{N}$. A *uniform ensemble* is a probability ensemble $U = U_1, U_2, \ldots$, where U_i is a random variable uniformly distributed over the set of bit strings of length $l(i)$. In the special case where $l(i) = i$ for all i, we refer to U as the *standard uniform ensemble.*

Example 4.1:
Suppose $X = X_1, X_2, \ldots$, where X_i is a random variable producing bit strings of length i following the distribution

$$P(X_i = (x_1, x_2, \ldots, x_i)) = \begin{cases} 0 & \text{if } x_1 = 0 \\ \frac{1}{2^{i-1}} & \text{if } x_1 = 1. \end{cases}$$

We can distinguish this ensemble from the standard uniform ensemble U by using the algorithm D that simply outputs the first bit of its input. Then

$$\Big| P\big(D(X_n, 1^n) = 1\big) - P\big(D(U_n, 1^n) = 1\big) \Big| = |1 - 1/2| = 1/2,$$

so X is not computationally indistinguishable from U. ■

Definition 4.4:
A *pseudorandom generator* is a deterministic polynomial-time algorithm G satisfying the following two conditions:

1. Expansion: There exists a function $l : \mathbf{N} \to \mathbf{N}$ such that $l(n) > n$ for all $n \in \mathbf{N}$, and $l(|s|) = |G(s)|$ for all $s \in \{0,1\}^*$;
2. Pseudorandomness: With $U_1, U_2, U_3, \ldots,$ being the standard uniform ensemble, the probability ensemble
$$G(U_1), G(U_2), G(U_3), \ldots$$
is computationally indistinguishable from some uniform ensemble
$$U'_1, U'_2, U'_3, \ldots.$$

With these definitions in place, we are ready to prove one of the main new results of this thesis: neural networks are not pseudorandom generators.

4.1.1 Distinguishing neural networks

Let $\Phi : \{0,1\}^n \to \{0,1\}^n$ be a neural network and let t be an integer greater than n. We consider the mapping
$$\Phi_t : \{0,1\}^n \to \{0,1\}^t$$
defined by letting $\Phi_t(x)$ denote the sequence of the first t bits in the concatenation of
$$x, \Phi(x), \Phi^2(x), \ldots.$$

We will show that such mappings cannot provide pseudorandom generators unless t is small relative to n:

Theorem 4.1:
There is a polynomial-time deterministic algorithm that, given a positive integer n and a sequence y of t bits, returns either the message `McCulloch- Pitts` *or the message* `random` *in such a way that*

(i) *if $y = \Phi_t(x)$ for some neural network $\Phi : \{0,1\}^n \to \{0,1\}^n$ and some x in $\{0,1\}^n$, then the algorithm returns* `McCulloch-Pitts`,

(ii) *if y is chosen uniformly from $\{0,1\}^t$ and if $t \geq (2+\varepsilon)n^2$ for some positive constant ε, then the algorithm returns* `random` *with probability at least $1-e^{-\delta n}$, where δ is a positive constant depending only on ε.*

We do not know whether or not Theorem 4.1 can be strengthened by reducing the lower bound $(2+\varepsilon)n^2$ on the length of y even just to $2n^2$. Nevertheless, this lower bound can be reduced all the way to $n+1$ if we are allowed to sample not just one, but multiple sequences $\Phi_t(x)$.

Theorem 4.2:
There is a polynomial-time deterministic algorithm that, given m sequences $y^1, \ldots, y^m$ of $n+1$ bits, returns either the message `McCulloch-Pitts` *or the message* `random` *in such a way that*

(i) *if $y^1 = \Phi_{n+1}(x^1), \ldots, y^m = \Phi_{n+1}(x^m)$ for some neural network $\Phi : \{0,1\}^n \to \{0,1\}^n$ and some $x^1, \ldots, x^m$ in $\{0,1\}^n$, then the algorithm returns* `McCulloch-Pitts`,

(ii) *if $m \geq (2+\varepsilon)n$ for some positive constant ε and if $y^1, \ldots, y^m$ are chosen independently and uniformly from $\{0,1\}^{n+1}$, then the algorithm returns* `random` *with probability at least $1-e^{-\delta n}$, where δ is a positive constant depending only on ε.*

We will use the following corollary of Lemma 2.2 on page 16:

Lemma 4.1:
For every positive ε there is a positive γ with the following property: If Y is a subset of $\mathbf{R}^n$ such that $|Y| \geq (2+\varepsilon)n$, then a dichotomy chosen uniformly from all dichotomies of Y is linearly separable with probability at most $e^{-\gamma n}$.

Proof:
It is enough to derive the conclusion under the additional assumption that $\varepsilon \leq 1$. Under this assumption, let m denote $|Y|$ and let p denote the probability that a dichotomy chosen uniformly from all dichotomies of Y is linearly separable. Since there are precisely 2^m dichotomies of Y and, by Lemma 2.2, at most $2\sum_{i=0}^{n}\binom{m-1}{i}$ of them are linearly separable, we have

$$p \leq 2^{-m+1}\sum_{i=0}^{n}\binom{m-1}{i} \leq 2^{-m+1}\sum_{i=0}^{n}\binom{m}{i}.$$

Since $m \geq (2+\varepsilon)n$ and $\varepsilon \leq 1$, we have $n \leq (0.5-\varepsilon/6)m$; a special case of the well-known bound on the tail of the binomial distribution (see, for instance, [27, Theorem

1]) guarantees that for every positive α smaller than 0.5 there is a positive β such that

$$\sum_{i \le (0.5-\alpha)m} \binom{m}{i} \le 2^m e^{-\beta m};$$

setting $\alpha = \varepsilon/6$, we conclude that $p \le 2e^{-\beta m}$, which proves the lemma. □

We will also use the following:

Lemma 4.2:
If $y^1, y^2, \dots, y^m$ are chosen independently and uniformly from a set of size N and if $m = O(N^{1/3})$, then $y^1, y^2, \dots, y^m$ are pairwise distinct with probability $1 - O(N^{-1/3})$.

Proof:
Note that $y^1, y^2, \dots, y^m$ are pairwise distinct with probability

$$\frac{N(N-1)\cdots(N-m+1)}{N^m}$$

and that

$$\frac{N(N-1)\cdots(N-m+1)}{N^m} \ge \left(\frac{N-m}{N}\right)^m = \left(1 - \frac{m}{N}\right)^m \ge 1 - \frac{m^2}{N}.$$ □

Proof of Theorem 4.1. The algorithm goes as follows: Let n be a positive integer and let y be a sequence of t bits. We write $m = \lfloor (t-1)/n \rfloor$ and define

$$Y^+ = \{(y_{(i-1)n+1}, y_{(i-1)n+2}, \dots, y_{in}) : 1 \le i \le m,\ y_{in+1} = 1\},$$
$$Y^- = \{(y_{(i-1)n+1}, y_{(i-1)n+2}, \dots, y_{in}) : 1 \le i \le m,\ y_{in+1} = 0\}.$$

If this dichotomy is linearly separable, then return `McCulloch-Pitts`; else return `random`.

To see that this algorithm runs in polynomial time, observe that testing whether a finite dichotomy is linearly separable amounts to solving a linear programming problem; the epoch-making result of Khachiyan [34] guarantees that this can be done in polynomial time.

To prove (i), let us assume that $y = \Phi_t(x)$ for some neural network $\Phi : \{0,1\}^n \to \{0,1\}^n$ defined by $\Phi(x) = (f_1(x), f_2(x), \dots, f_n(x))$ and for some x in $\{0,1\}^n$. Now

$$y_{in+1} = f_1(y_{(i-1)n+1}, y_{(i-1)n+2} \cdots, y_{in})$$

for all $i = 1, 2, \ldots, m$, which means that f_1 takes value 1 on all points of Y^+ and value 0 on all points of Y^-; since f_1 is a threshold function, the dichotomy (Y^+, Y^-) is linearly separable, and so the algorithm returns `McCulloch-Pitts`.

To prove (ii), let us assume that y is chosen uniformly from $\{0,1\}^t$ and that $t \geq (2+\varepsilon)n^2$ for some positive constant ε. Since the probability that the algorithm returns `random` increases as t increases, we may replace the assumption that $t \geq (2+\varepsilon)n^2$ by the assumption that $t = \lceil (2+\varepsilon)n^2 \rceil$. Write $Y = Y^+ \cup Y^-$. Since y is chosen uniformly from $\{0,1\}^t$, the points of Y are chosen independently and uniformly from $\{0,1\}^n$, and so Lemma 4.2 with $N = 2^n$ guarantees that $|Y| = m$ with probability $1 - O(2^{-n/3})$. When $|Y| = m$, the assumption that y is chosen uniformly from $\{0,1\}^t$ implies that the dichotomy (Y^+, Y^-) of Y is chosen uniformly from all dichotomies of Y, in which case Lemma 4.1 guarantees that (Y^+, Y^-) is linearly separable with probability at most $e^{-\gamma n}$. We conclude that the algorithm returns `random` with probability at least $1 - O(2^{-n/3}) - e^{-\gamma n}$, which is at least $1 - e^{-\delta n}$ for some positive δ. □

Proof of Theorem 4.2. The algorithm goes as follows: Let $y^1, \ldots, y^t$ be sequences of $n+1$ bits. Write $y^i = (y^i_1, \ldots, y^i_{n+1})$ for all i and define

$$Y^+ = \{(y^i_1, y^i_2, \ldots, y^i_n) : 1 \leq i \leq m,\ y^i_{n+1} = 1\},$$
$$Y^- = \{(y^i_1, y^i_2, \ldots, y^i_n) : 1 \leq i \leq m,\ y^i_{n+1} = 0\}.$$

If this dichotomy is linearly separable, then return `McCulloch-Pitts`; else return `random`.

Analysis of this algorithm is just like the analysis in the proof of Theorem 4.1.

□

4.1.2 Indistinguishability of neural networks with hidden neurons

We will now consider a relaxation of the problem by allowing our neural network to contain *hidden* neurons, that is, neurons that are hidden from the distinguisher.

Definition 4.5:
Consider an N-dimensional neural network Φ with neurons $\nu_1, \ldots, \nu_N$. We partition the network into 2 types of neurons: n *visible* neurons and $N - n$ *hidden* neurons. We adopt the convention that the visible neurons are listed first, so that

the visible neurons are $\nu_1, \ldots, \nu_n$. When the network is initialized with a seed $\mathbf{x}$ in $\{0,1\}^N$, we denote the visible state of the network at time t by $\Phi_v^t(\mathbf{x})$, where $\Phi_v^t(\mathbf{x}) = (\Phi_1^t(\mathbf{x}), \ldots, \Phi_n^t(\mathbf{x}))$.

We will show that a neural network with hidden neurons can produce any trajectory in its visible part. First, we define boolean networks.

Definition 4.6:
Let

$$g_i : \{0,1\}^n \to \{0,1\} \quad (i = 1, 2, \ldots, n)$$

be n boolean functions. The function $G : \{0,1\}^n \to \{0,1\}^n$ defined by

$$G(\mathbf{x}) = (g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_n(\mathbf{x}))$$

is an *n-dimensional boolean network.*

Thus, neural networks are simply boolean networks where each underlying boolean function is threshold. Boolean networks are also dynamical systems that produce trajectories $\mathbf{x}, G(\mathbf{x}), G^2(\mathbf{x}), \ldots$ for a seed $\mathbf{x}$. Clearly, any dynamical system on $\{0,1\}^n$ can be seen as a boolean network of dimension n. Next, we provide a trivial method for converting boolean networks into neural networks.

Theorem 4.3:
For every boolean network $G : \{0,1\}^n \to \{0,1\}^n$ there exists a neural network $\Phi : \{0,1\}^N \to \{0,1\}^N$, where $N = n + 2^n$, with the following property: For every $\mathbf{x}$ in $\{0,1\}^n$ there exists an $\mathbf{x}'$ in $\{0,1\}^N$ such that, for all nonnegative integers t, the restriction of $\Phi^t(\mathbf{x}')$ on its first n components equals $G^t(\mathbf{x})$.

Proof:
Let $g_1, \ldots, g_n$ be the boolean functions of the boolean network G. For each $i = 1, \ldots, n$, we associate a visible neuron ν_i with g_i; we create 2^n hidden neurons $\nu_{\mathbf{y}}$ with $\mathbf{y}$ ranging over $\{0,1\}^n$.

At each time $t+1$, a visible neuron ν_i will fire if and only if a hidden neuron $\nu_{\mathbf{y}}$ such that $g_i(\mathbf{y}) = 1$ fired at time t. This can be accomplished by letting all hidden neurons

$\nu_{\mathbf{y}}$ such that $g_i(\mathbf{y}) = 1$ feed into ν_i, setting the weight on every link into ν_i at 1, and setting the threshold value of ν_i at 1.

At each time $t+1$, precisely one of the hidden neurons will fire: this is the hidden neuron $\nu_{\mathbf{y}}$ such that the visible neurons ν_i with $y_i = 1$ fired at time t and the visible neurons ν_i with $y_i = 0$ did not fire at time t. This can be accomplished by letting all visible neurons feed into all hidden neurons, setting the weight on the link from visible neuron ν_i to hidden neuron $\nu_{\mathbf{y}}$ at $2y_i - 1$, and setting the threshold value of $\nu_{\mathbf{y}}$ at $\sum_i y_i$.

Finally, given an $\mathbf{x}$ in $\{0,1\}^n$, we define an $\mathbf{x}'$ in $\{0,1\}^N$ as follows: the restriction of $\mathbf{x}'$ on its first n components equals $\mathbf{x}$, the remainining 2^n components of $\mathbf{x}'$ are indexed by vectors in $\{0,1\}^n$, and

$$\mathbf{x}'_{\mathbf{y}} = \begin{cases} 1 & \text{if } \mathbf{y} = \mathbf{x}, \\ 0 & \text{otherwise.} \end{cases}$$

Induction on t shows that for all nonnegative integers t, the restriction of $\Phi^t(\mathbf{x}')$ on its first n components equals $G^t(\mathbf{x})$. □

The consequence of Theorem 4.3 is that neural networks with no restrictions on the number of hidden neurons *can* be pseudorandom number generators, *if pseudorandom number generators exist.* However, it is known that the existence of pseudorandom number generators implies $P \neq NP$ [18]. Thus, attempts at proving the existence of pseudorandom number generators in the form of neural networks with hidden neurons will not be pursued here. Theorem 4.3 also illustrates the need for a more refined version of the problem that places bounds on the number of hidden neurons.

4.2 Statistical tests

In this section we will depart from the theoretical notions of Section 4.1, and consider the more practical and commonly used statistical tests for evaluating pseudorandom number generators. We shall describe some neural networks that produce pseudorandom-like output, in the sense that they pass many statistical tests. This does not necessarily contradict the results on theoretical pseudorandom generators of the previous section. The famous linear congruential generators may shed some light on this: on the one hand, most runtime libraries of the C programming language use a linear congruential generator to produce the output of their *rand*() function. On the other hand, it has been shown that linear congruential generators are not pseudorandom generators in the theoretical sense ([6], [16], [69], [36]).

Several test suites exist for testing pseudorandom sequences, including L'Ecuyer and Simard's *TestU01* ([41], [42]), Marsaglia's *Diehard tests* [49], and the *NIST* (National Institute of Standards and Technology) test suite [65]. We will use *TestU01* since it is the most stringent, is the most up-to-date, and also contains versions of the *Diehard* and *NIST* test suites. We will use the smallest battery *SmallCrush*, consisting of 10 tests, since our goal is to find random-*looking* trajectories; the larger batteries in *TestU01* are more appropriate for generators that need to be secure or have specific statistical usage.

Our focus will be on neural networks that contain hidden neurons. In particular, does there exist a neural network with a single visible neuron such that the bit sequence generated by that neuron passes statistical tests? More precisely, does there exist a neural network $\Phi : \{0,1\}^n \to \{0,1\}^n$ such that the bit sequence

$$\Phi_1(\mathbf{x}), \Phi_1^2(\mathbf{x}), \Phi_1^3(\mathbf{x}), \ldots \tag{4.1}$$

passes all the tests in *SmallCrush* when initialized with a seed $\mathbf{x}$? What is the smallest n for which we can answer this question in the affirmative? In (4.1) we are singling out the first bit of each state in the trajectory of $\mathbf{x}$ under Φ.

The shift from studying the capabilities of neural networks as pseudorandom generators in the theoretical sense to the practical implies a shift from looking at neural networks in general to looking at specific instances of neural networks. Our main candidates are the maximal neural networks of Section 3.1, permutations of them (as described in Section 3.2.2), and the shift register networks of Section 3.2.1.

The n-dimensional maximal neural networks of Section 3.1 fail all of the tests in *SmallCrush* for $32 < n < 100$. There is too much structure in their trajectories and we should not be surprised that the very stringent tests in *SmallCrush* detect this. The shift register networks perform better than this, and we turn our attention to them next.

Shift register networks

Let $\Phi : \{0,1\}^{n+2} \to \{0,1\}^{n+2}$ be a shift register network with primitive trinomial $z^n + z^m + 1$. Then the sequence (4.1) is simply the bit sequence

$$x_0, x_1, x_2, \ldots,$$

produced by the linear recurrence

$$x_{t+n} = x_{t+m} + x_t \pmod 2.$$

Over all primitive trinomials of degree less than 100 (there are 179), none of the corresponding networks passed all of the tests in *SmallCrush*. The best results were produced by 14 of these networks, which passed 7 out of the 10 tests; the *Gap* test, the *WeightDistrib* test, and the *RandomWalk1* tests were failed. These networks, described by the parameters n and m, are:

n	m
71	18
71	51
71	53
73	42
84	13
84	71
87	13
87	74
89	38
89	51
94	21
94	73
95	84
98	87

Table 4.1: The 14 shift register networks, defined by $z^n + z^m + 1$, that pass 7 of the 10 tests in *SmallCrush*.

4.2.1 Modifying output via xor

One way to improve on the number of tests that are passed is to output the *xor* of some subset of neurons of a particular network. This has been done in [15], where random orthogonal neural networks produced one bit per time step by outputting the *xor* of the first 5 neurons of the network. This idea will also allow us to combine the outputs of many neural networks into one in the next section. Although Knuth [35] cautions against this sort of technique:

> "One of the common fallacies encountered in connection with random number generation is the idea that we can take a good generator and modify it a little, in order to get an even-more-random sequence",

there are many instances where it works. For example, there are many variations on shift register sequences that yield slightly better results and are often used; Matsumoto and Nishimura's famous *Mersenne Twister* [53], Marsaglia's *xorshift* [50], and Panneton's *WELL RNG's* [62], to name a few.

Theorem 4.4:
For every neural network $\Phi : \{0,1\}^n \to \{0,1\}^n$ and for every subset S of $\{1, 2 \ldots, n\}$ of size m there exists a neural network $\Omega : \{0,1\}^{n+m+1} \to \{0,1\}^{n+m+1}$ such that for all nonnegative integers t, for all $\mathbf{x}$ in $\{0,1\}^n$, and for all $\mathbf{x}'$ in $\{0,1\}^{n+m+1}$ whose restriction on the first n components equals $\mathbf{x}$, we have

$$\Omega^{t+2}_{n+m+1}(\mathbf{x}') = \sum_{i \in S} \Phi^t_i(\mathbf{x}) \pmod{2}.$$

Proof:
The first n neurons of Ω are simply copies of the original neurons $\nu_1, \nu_2, \ldots, \nu_n$, as in Φ. We assign them weights of 0 for all incoming connections from $\nu_{n+1}, \nu_{n+2}, \ldots, \nu_{n+m+1}$; the augmenting neurons have no bearing on the original ones.

We introduce $m+1$ new neurons $\nu_{n+1}, \nu_{n+2}, \ldots, \nu_{n+m+1}$. Without loss of generality, we may assume that $S = \{1, 2, \ldots, m\}$. For $i = 1, \ldots, m$, neuron ν_{n+i} receives the output of neurons $\nu_1, \nu_2, \ldots, \nu_m$, and only these neurons. Thus, each of the m neurons $\nu_{n+1}, \nu_{n+2}, \ldots, \nu_{n+m}$ receives the same m inputs. We assign a weight of 1 for each of these inputs. The threshold value of ν_{n+i} is i, for $i = 1, 2, \ldots, m$. Thus, neuron ν_{n+i} takes the value 1 if and only if at least i of its inputs are 1. This allows us to detect the parity of the number of neurons in $\{\nu_1, \nu_2, \ldots, \nu_m\}$ that are firing as follows. We add one more neuron, ν_{n+m+1}. This neuron receives m inputs, one from each ν_{n+i} with $1 \leq i \leq m$. The weight on the input from ν_{n+i} is 1 if i is odd, and -1 if i is even. The threshold of ν_{n+m+1} is 1. The value of ν_{n+m+1} at time $t+2$ is therefore the *xor* of the values of the neurons $\nu_{n+1}, \nu_{n+2}, \ldots, \nu_{n+m}$ at time t. $\square$

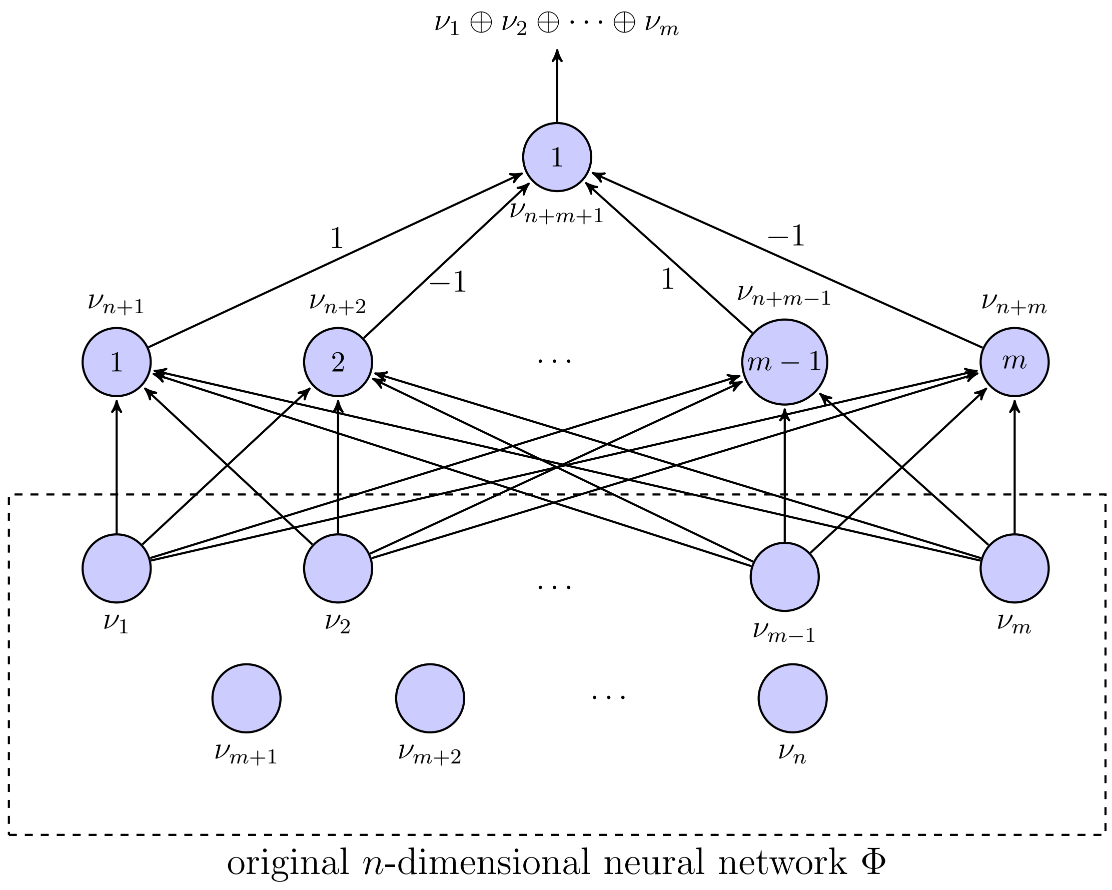


Figure 4.1: The augmented neural network Ω, producing the *xor* output of neurons $\nu_1, \nu_2, \ldots, \nu_m$ after two time steps. Unlabelled connections have a weight of 1. Here we assume m is even. The weights and thresholds of neurons in Φ are omitted.

Our method of computing the *xor* is essentially the *parity gadget* used in [15]. Note that this computation is simpler than that used in the proof of Theorem 3.8, since the *xor* result does not need to be fed back into the network.

For an n-dimensional neural network Φ, we will use $\Phi^{\oplus m}$ to denote the augmented $(n+m+1)$-dimensional neural network described in Theorem 4.4.

Modified permuted maximal neural networks using *xor*

Let Φ_n be the n-dimensional maximal neural network described in Section 3.1.3, with weight matrix W_n and threshold vector Θ_n. Let $\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ be a permutation with corresponding permutation matrix P. We use $\pi(\Phi_n)$ as a shorthand for the neural network with weight matrix PW_n and threshold vector $P\Theta_n$. Fixing m such that $1 \le m \le n$, we consider the $(n+m+1)$-dimensional neural network $(\pi(\Phi_n))^{\oplus m}$, in which we output the *xor* of the first m neurons of the permuted network $\pi(\Phi_n)$.

The following was done 10 times for each dimension n and tap size m such that $30 < n+m+1 \leq 40$: a random permutation π and a random seed were generated; the neural network $(\pi(\Phi_n))^{\oplus m}$ was then tested by *SmallCrush*. Of the 550 neural networks tested, 35 passed all of the tests. The smallest such network was of dimension 37, having $m = 5$ and permutation π defined by

i	$\pi(i)$
1	30
2	25
3	13
4	9
5	3
6	10
7	11
8	22
9	26
10	17
11	20
12	28
13	18
14	1
15	21
16	23
17	7
18	2
19	12
20	24
21	14
22	15
23	6
24	5
25	19
26	8
27	27
28	4
29	31
30	29
31	16

The weight matrix of this neural network is given explicitly on the following page.

```
0,1, −1,1, −1,1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0,0, 0,0, 0,0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0,0, 0,0, 0,0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0,0, 0,0, 0,0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0,0, 0,0, 0,0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0,0, 0,0, 0,0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0,0, 0,0, 0,0, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, 29, −1
0,0, 0,0, 0,0, −13, 13, −13, 13, −13, 13, −13, 13, −13, 13, −13, 13, −13, 13, −13, 13, −13, 13, −13, 13, −13, 13, −13, 13, 307, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, −4181, 4181, −4181, 4181, −4181, 4181, −4181, 4181, −4181, 4181, −4181, 4181, 48575, −2584, −1597, −987, −610, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, −28657, 28657, −28657, 28657, −28657, 28657, −28657, 28657, 218310, −17711, −10946, −6765, −4181, −2584, −1597, −987, −610, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, −514229, 514229, 832040, −317811, −196418, −121393, −75025, −46368, −28657, −17711, −10946, −6765, −4181, −2584, −1597, −987, −610, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, 17711, −17711, 17711, −17711, 17711, −17711, 17711, −17711, 17711, 152634, −10946, −6765, −4181, −2584, −1597, −987, −610, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, −10946, 10946, −10946, 10946, −10946, 10946, −10946, 10946, −10946, 10946, 105279, −6765, −4181, −2584, −1597, −987, −610, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, 55, −55, 55, −55, 55, −55, 55, −55, 55, −55, 55, −55, 55, −55, 55, −55, 55, −55, 55, −55, 55, 1134, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, 8, −8, 8, −8, 8, −8, 8, −8, 8, −8, 8, −8, 8, −8, 8, −8, 8, −8, 8, −8, 8, −8, 8, −8, 8, 197, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, −610, 610, −610, 610, −610, 610, −610, 610, −610, 610, −610, 610, −610, 610, −610, 610, 9527, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, 144, −144, 144, −144, 144, −144, 144, −144, 144, −144, 144, −144, 144, −144, 144, −144, 144, −144, 144, 2681, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, 3, −3, 3, −3, 3, −3, 3, −3, 3, −3, 3, −3, 3, −3, 3, −3, 3, −3, 3, −3, 3, −3, 3, −3, 3, −3, 3, 80, −2, −1, −1
0,0, 0,0, 0,0, 377, −377, 377, −377, 377, −377, 377, −377, 377, −377, 377, −377, 377, −377, 377, −377, 377, 6265, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, −514229, −832040, −514229, −317811, −196418, −121393, −75025, −46368, −28657, −17711, −10946, −6765, −4181, −2584, −1597, −987, −610, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, −89, 89, −89, 89, −89, 89, −89, 89, −89, 89, −89, 89, −89, 89, −89, 89, −89, 89, −89, 89, 1746, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, −34, 34, −34, 34, −34, 34, −34, 34, −34, 34, −34, 34, −34, 34, −34, 34, −34, 34, −34, 34, −34, 34, 735, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, −75025, 75025, −75025, 75025, −75025, 75025, 421493, −46368, −28657, −17711, −10946, −6765, −4181, −2584, −1597, −987, −610, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, 832040, 514229, −514229, −317811, −196418, −121393, −75025, −46368, −28657, −17711, −10946, −6765, −4181, −2584, −1597, −987, −610, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, 6765, −6765, 6765, −6765, 6765, −6765, 6765, −6765, 6765, −6765, 6765, 71831, −4181, −2584, −1597, −987, −610, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, 21, −21, 21, −21, 21, −21, 21, −21, 21, −21, 21, −21, 21, −21, 21, −21, 21, −21, 21, −21, 21, −21, 21, 475, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, 2584, −2584, 2584, −2584, 2584, −2584, 2584, −2584, 2584, −2584, 2584, −2584, 2584, 32605, −1597, −987, −610, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, −1597, 1597, −1597, 1597, −1597, 1597, −1597, 1597, −1597, 1597, −1597, 1597, −1597, 1597, 21748, −987, −610, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, 121393, −121393, 121393, −121393, 121393, 560597, −75025, −46368, −28657, −17711, −10946, −6765, −4181, −2584, −1597, −987, −610, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, −196418, 196418, −196418, 196418, 710647, −121393, −75025, −46368, −28657, −17711, −10946, −6765, −4181, −2584, −1597, −987, −610, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, −233, 233, −233, 233, −233, 233, −233, 233, −233, 233, −233, 233, −233, 233, −233, 233, −233, 233, 4105, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, 46368, −46368, 46368, −46368, 46368, −46368, 46368, 306865, −28657, −17711, −10946, −6765, −4181, −2584, −1597, −987, −610, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, −5, 5, −5, 5, −5, 5, −5, 5, −5, 5, −5, 5, −5, 5, −5, 5, −5, 5, −5, 5, −5, 5, 128, −3, −2, −1, −1
0,0, 0,0, 0,0, 317811, −317811, 317811, 832040, −196418, −121393, −75025, −46368, −28657, −17711, −10946, −6765, −4181, −2584, −1597, −987, −610, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
0,0, 0,0, 0,0, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, −1, 1, 29
0,0, 0,0, 0,0, −2, 2, −2, 2, −2, 2, −2, 2, −2, 2, −2, 2, −2, 2, −2, 2, −2, 2, −2, 2, −2, 2, −2, 2, 55, −1, −1
0,0, 0,0, 0,0, 987, −987, 987, −987, 987, −987, 987, −987, 987, −987, 987, −987, 987, −987, 987, 14428, −610, −377, −233, −144, −89, −55, −34, −21, −13, −8, −5, −3, −2, −1, −1
```

The threshold vector is given by

$$\begin{bmatrix} 1 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 15 \\ 144 \\ 20906 \\ 85972 \\ 1 \\ 70845 \\ 43785 \\ 551 \\ 97 \\ 4271 \\ 1297 \\ 40 \\ 3017 \\ -1346268 \\ 802 \\ 341 \\ 150051 \\ 1 \\ 33826 \\ 232 \\ 15505 \\ 9583 \\ 242787 \\ 196419 \\ 1865 \\ 139105 \\ 61 \\ 317812 \\ 15 \\ 27 \\ 6910 \end{bmatrix}.$$

In this representation the first neuron is the visible one, all other neurons are hidden.

Random orthogonal weight matrices

As we saw in Section 3.2.3 on page 71, Elyada and Horn have used neural networks to generate random numbers as follows: a random $n \times n$ matrix is computed and then orthogonalized. This is used as the weight matrix of a neural network. The thresholds are then set in a way that makes each of the threshold functions self-dual (see Theorem 2.3). The output at time $t+2$ is then the *xor* of the first 5 bits of the state of the network at time t, This results in a network of size $n+6$ (see Theorem 4.4).

Elyada and Horn tested their method using the *NIST* test suite [65]. The results in [15] state that these networks passed all but 2 of the 189 tests in [65]. However, network sizes were not specified.

We tested this method using *SmallCrush*. The following was done 10 times for each n

such that $32 \leq n \leq 64$: a random $n \times n$ orthogonal weight matrix and self-dualizing threshold vector was generated. This network was tested by *SmallCrush* with a randomly generated seed. The smallest network to pass all the tests was of dimension 66 (consisting of a 60-dimensional orthogonal weight matrix and 6 extra neurons for the *xor*).

4.2.2 Combining networks via xor

Another method of modifying the output of some of our base networks is to combine two or more of them into one big network. When combining k networks, we will do this by taking the *xor* of k neurons, one from each of the initial networks. In the case where the underlying networks are all shift register networks, the combined network is equivalent to what are referred to as *combined generators* in [40] and [39].

Suppose that we have k neural networks, each of which has precisely one visible neuron producing the bit sequence

$$x_0^i, x_1^i, x_2^i, \ldots, \tag{4.2}$$

for $i = 1, 2, \ldots, k$. We combine these k sequences into one by setting

$$x_t = x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus \cdots \oplus x_t^k \qquad \text{for } t = 0, 1, 2, \ldots. \tag{4.3}$$

Thus, we take the *xor* of the k sequences (4.2). We will now show that this scheme can be implemented in a neural network.

Theorem 4.5:
For $i = 1, 2, \ldots, k$, let $\Phi_i : \{0,1\}^{n_i} \to \{0,1\}^{n_i}$ be a neural network. Setting $N = \sum_{i=1}^{k} n_i + k + 1$, there exists an augmented neural network $\Omega : \{0,1\}^N \to \{0,1\}^N$ such that for all nonnegative integers t, for all $\mathbf{x}_i$ in $\{0,1\}^{n_i}$ with $(i = 1, 2, \ldots, k)$, and for all $\mathbf{x}$ in $\{0,1\}^N$ whose restriction on the first $N - k - 1$ components equals the concatenation of $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ we have

$$\Omega_N^{t+2}(\mathbf{x}) = \sum_{i=1}^{k} \Phi_{i,1}^t(\mathbf{x}_i) \pmod 2.$$

Proof:
The proof is similar to that of Theorem 4.4. For $i = 1, 2, \ldots, k$, let $\nu_1^i, \nu_2^i, \ldots, \nu_{n_i}^i$ be the neurons of Φ_i. We begin constructing our augmented neural network Ω. We keep all thresholds and weights as before, with no new connections between neurons coming from different networks; we are simply copying each of the k neural networks and running them in parallel.

We combine these neurons using the parity gadget. For $i = 1, 2, \ldots, k$, neuron ν_{N-i} receives the output of neurons $\nu_1^1, \nu_1^2, \ldots, \nu_1^k$, and only these neurons. Each of the k neurons $\nu_{N-k}, \nu_{N-k+1}, \ldots, \nu_{N-1}$ receives the same k inputs. We assign a weight of 1 for each of these inputs. The threshold value of $\nu_{N-k+i-1}$ is i, for $i = 1, 2, \ldots, k$. Thus, neuron $\nu_{N-k+i-1}$ takes the value 1 if and only if at least i of its inputs are 1. This allows us to detect the parity of the number of neurons in $\{\nu_1^1, \nu_1^2, \ldots, \nu_1^k\}$ that are firing as follows. We add one more neuron, ν_N. This neuron receives k inputs, one from each ν_{N-i}, for $i = 1, 2, \ldots, k$. The weight on the input from $\nu_{N-k+i-1}$ is 1 if i is odd, and -1 if i is even. The threshold of ν_N is 1. The value of ν_N at time $t + 2$ is therefore the *xor* of the values of the neurons $\nu_1^1, \nu_1^2, \ldots, \nu_1^k$ at time t. □

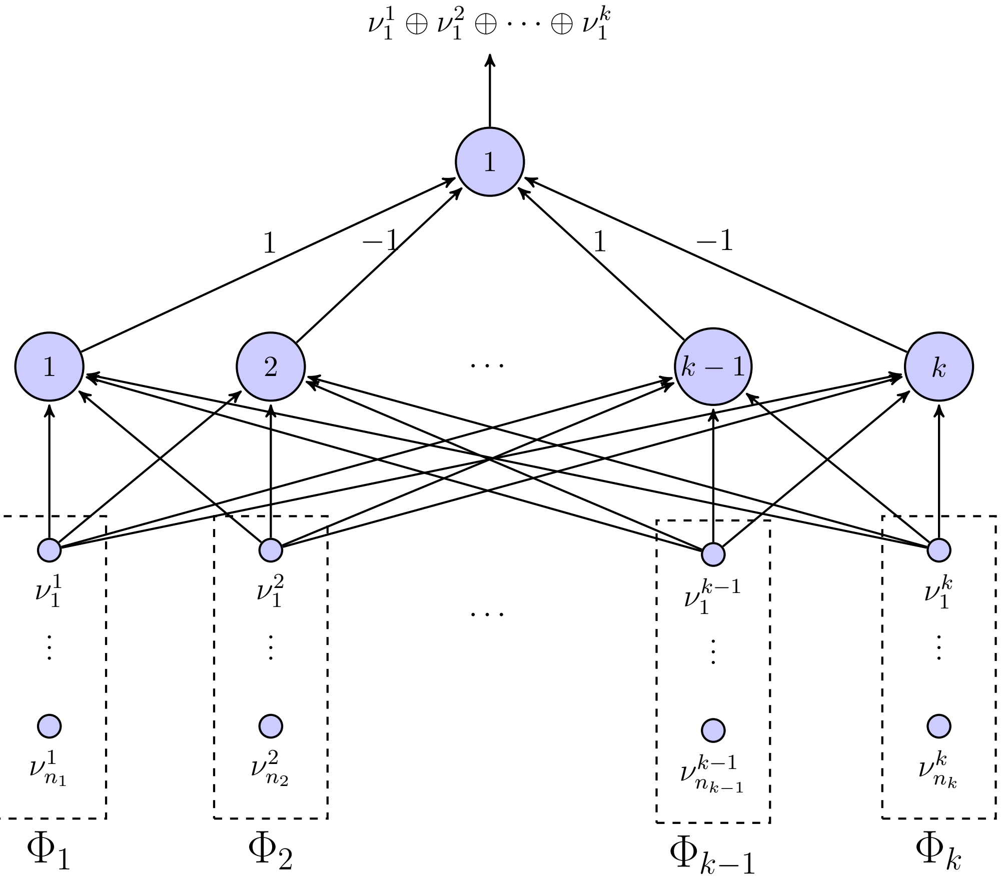


Figure 4.2: Combination of the networks $\Phi_1, \Phi_2, \ldots, \Phi_k$. Unlabelled connections have a weight of 1. Here we assume k is even.

Combined shift register networks

Let $P_1(z), P_2(z), \ldots, P_k(z)$ be primitive trinomials, where $P_i(z) = z^{n_i} + z^{m_i} + 1$ and $n_i > m_i > 0$ for $i = 1, 2, \ldots, k$. Then for $i = 1, 2, \ldots, k$, the primitive trinomial $P_i(z)$

yields a linear recurrence

$$x^i_{t+n_i} = x^i_{t+m_i} + x^i_t \pmod 2,$$

which in turns yields the bit sequence

$$x^i_0, x^i_1, x^i_2, \ldots, \tag{4.4}$$

where $(x^i_0, x^i_1, \ldots, x^i_{n_i-1})$ is the initial seed. In this setting, it has been shown that the combined sequence (4.3) is also the shift register sequence defined by the (reducible) polynomial $P(z) = P_1(z) \cdots P_k(z)$, and will have period equal to $(2^{n_1} - 1)(2^{n_2} - 1) \cdots (2^{n_k} - 1)$, if the polynomials $P_i(z)$ are relatively prime [72], [75], [38].

Combining two shift register networks

We consider primitive trinomials with degree less than 30. There are 55 of these, and thus 1485 pairs to consider. None of these pairs passed all of the tests in *SmallCrush*, but 143 pairs passed all but one test; the *MatrixRank* test. Shift register sequences and some of their variants are known to fail this test [51], [48]. The smallest network that passed all but the *MatrixRank* test consisted of 52 neurons, and in fact there were 4 of these, listed below.

n_1	m_1	n_2	m_2
20	3	25	22
20	17	25	7
20	17	25	18
22	1	23	18

Table 4.2: The 4 combined shift register networks $\{z^{n_1} + z^{m_1} + 1, z^{n_2} + z^{m_2} + 1\}$ consisting of 52 neurons that pass all but the *MatrixRank* test in *SmallCrush*.

Combining three shift register networks

The smallest networks that passed all tests consisted of 75 neurons. There were 37 of these:

n_1	m_1	n_2	m_2	n_3	m_3
11	2	25	3	29	2
11	2	25	7	29	27
11	2	25	18	29	2
11	2	25	18	29	27
11	9	25	3	29	2
11	9	25	3	29	27
11	9	25	18	29	27
11	9	25	22	29	2
17	3	23	5	25	18
17	3	23	9	25	7
17	3	23	9	25	18
17	3	23	9	25	22
17	3	23	14	25	7
17	3	23	18	25	3
17	3	23	18	25	18
17	5	23	9	25	3
17	5	23	18	25	7
17	6	23	9	25	3
17	6	23	9	25	22
17	6	23	18	25	18
17	11	23	9	25	3
17	11	23	9	25	7
17	11	23	9	25	22
17	11	23	18	25	22
17	12	23	5	25	22
17	12	23	9	25	7
17	12	23	9	25	22
17	12	23	14	25	22
17	12	23	18	25	7
17	12	23	18	25	18
17	14	23	5	25	3
17	14	23	5	25	18
17	14	23	9	25	18
17	14	23	14	25	7
17	14	23	14	25	22
17	14	23	18	25	3
17	14	23	18	25	7

Table 4.3: The 37 combined shift register networks $\{z^{n_1}+z^{m_1}+1, z^{n_2}+z^{m_2}+1, z^{n_3}+z^{m_3}+1\}$ consisting of 75 neurons that pass all tests in *SmallCrush*.

Combining a shift register network and a permuted maximal network

We can also combine a shift register network ψ and a permuted maximal network Φ by outputting the *xor* of a neuron in ψ and a neuron in Φ. The following search was repeated 10 times: for all n-dimensional shift register networks ψ and m-dimensional maximal networks Φ such that $30 \leq n+m \leq 40$, a random m-dimensional permutation π and random seed was generated. The resulting network was tested in *SmallCrush*. The smallest network to pass all tests had dimension 40, consisting of the shift register network with primitive trinomial $z^{15}+z^{11}+1$, the maximal neural network of dimension 20, and the permutation π defined by

i	$\pi(i)$
1	5
2	18
3	16
4	20
5	19
6	4
7	15
8	14
9	11
10	17
11	7
12	8
13	9
14	1
15	12
16	3
17	10
18	2
19	6
20	13

Table 4.4: The permutation for the 20-dimensional maximal neural network.

4.2.3 Many visible neurons

So far, we have focused on neural networks where all neurons are hidden except one. The successive states of this neuron are used to generate a bit sequence that, in some cases, looks random. A reasonable question now is: can we use the neural networks with one visible random neuron to create a new neural network with many visible neurons that behave randomly? One way to go about this is to take one of the neural networks with a single random neuron described above, make N copies of the network, initialize each copy with a random seed, and run each of the N networks in parallel. Thus we take N networks, each with 1 visible neuron, and combine them into a network with N visible neurons. If the original visible neuron in each of the respective neural networks behaved randomly, then we would intuitively expect that the successive N-bit strings generated at each time step would appear random as well. To test this idea, the 37-dimensional permuted maximal neural network given on page 86 was rigged in this manner: 32 copies were made, each with 1 visible neuron, to generate a 32-bit string at each time step, thus resulting in a neural network with 1184 neurons. These 32-bit strings were tested by *SmallCrush*. All tests were passed, as one might expect. Finally, this experiment illustrates the time vs. space trade off: previously, if we wanted our network to produce 32 random bits, we would run our network 32 time steps, getting 1 bit at each step. The network described in this section gives us 32 bits at once, at the cost of needing more neurons in our network.

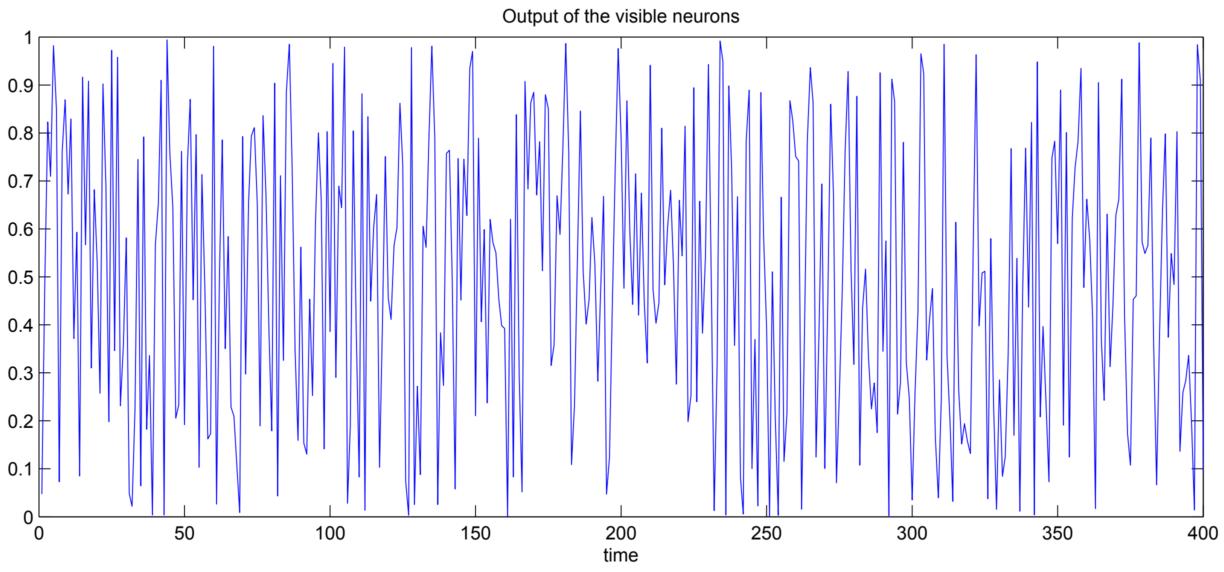


Figure 4.3: Output of the 32 visible neurons in the 1184-dimensional network for 400 time steps. The 32-bit strings are interpereted as integers, and then normalized to $[0, 1]$.

4.2.4 Summary of experimental tests

The smallest network found to pass all of the tests in *SmallCrush* was a permuted maximal neural network consisting of 37 neurons; 31 neurons for the original maximal network, and 6 neurons to compute the *xor* of the first 5 neurons in the permuted network. This network is given on page 86. Another successful network was a combined one consisting of 40 neurons; 17 neurons for the shift register network with primitive polynomial $z^{15}+z^{11}+1$, and 20 neurons for the permuted maximal neural network with permutation given in Table 4.4. The remaining 3 neurons were needed to compute the *xor* of the visible neurons.

Chapter 5

Conclusion

This work originated from the desire to conduct interdisciplinary research in order to bridge the gap between mathematical models of the brain and the observations commonly found by neurologists. In this sense, a more appropriate title may have been "Can neural networks fool a neurologist?" In particular, our initial goal was to find a recurrent neural network whose output looks like an EEG recording showing the evolution of a seizure: beginning with a random-looking output, progressing to a more patterned signal, followed by full-on synchronization (the seizure), and then a reversion back to the random-looking output. In an attempt to model this progression, it became clear that the simulation of the normal resting state of the brain was a difficult enough challenge in itself, and that is what we have focused on here.

The goal of simulating an EEG recording ultimately led to the study of randomness and pseudorandomness, and some conclusions about the capabilities of neural networks as pure mathematical objects. On the one hand, we showed that a neural network cannot be a pseudorandom number generator, in the sense that its trajectory can be distinguished from a truly random source in polynomial time. On the other hand, we have given explicit constructions of some neural networks that can pass all of the tests in the *SmallCrush* battery of tests, and this was an unexpected success. That being said, we do not claim that any of the neural networks described here can compete with the state of the art pseudorandom number generators.

The results that we have provided lead to many unanswered follow-up questions. We saw that the maximal period networks described in Chapter 3 are not exhaustive, yet there is no known generalized construction for any others. What is the structure of the other maximal neural networks? Our experimental results were based on networks in which a significant number of neurons were hidden; what is the smallest neural network with no hidden neurons that can pass a reasonable testing suite? Finally, we have restricted ourselves to neural networks using threshold functions as model neurons, in which no external input is allowed, and all neurons fire in sync. What happens when we relax these conditions or use a model that is more biologically credible?

Bibliography

[1] Random.org. https://www.random.org/.

[2] S. Altunay, Z. Telatar, and O. Erogul. Epileptic EEG detection using the linear prediction error energy. *Expert Systems with Applications*, 37(8):5661–5665, 2010.

[3] M. Anthony. *Discrete Mathematics of Neural Networks: Selected Topics.* Society for Industrial and Applied Mathematics, 2001.

[4] S. Arimoto. Periodic sequences of states of an autonomous circuit consisting of threshold elements. *(Japanese), Trans. Inst. Electron. Comm. Eng., Studies on Information & Control 2*, pages 17–22, 1963.

[5] J. Beck. On a geometric problem of Erdös, Sárközy, and Szemerédi concerning vector sums. *European Journal of Combinatorics*, 4(1):1–10, 1983.

[6] J. Boyar. Inferring sequences produced by pseudo-random number generators. *Journal of the ACM (JACM)*, 36(1):129–141, 1989.

[7] T. R. Browne and G. L. Holmes. *Handbook of Epilepsy.* Jones & Bartlett Learning, 2008.

[8] W. A. Chaovalitwongse. Optimization and data mining in epilepsy research: A review and prospective. In *Handbook of Optimization in Medicine*, pages 1–32. Springer, 2009.

[9] C. K. Chow. Boolean functions realizable with single threshold devices. In *Proceedings of the Institute of Radio Engineers*, Vol. 49, pages 370–371, 1961.

[10] V. Chvátal and M. Goldsmith. Can brains generate random numbers? *arXiv:1208.6451 [math, q-bio]*, 2012.

[11] R. R. Clancy. Interictal sharp EEG transients in neonatal seizures. *Journal of Child Neurology*, 4(1):30–38, 1989.

[12] T. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-14(3):326–334, 1965.

[13] S. N. Elaydi. *Discrete Chaos, Second Edition: With Applications in Science and Engineering.* Chapman and Hall/CRC, Boca Raton, 2 edition edition, 2007.

[14] C. C. Elgot. Truth functions realizable by single threshold organs. In *Proceedings of the 2Nd Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1961)*, FOCS '61, pages 225–245. IEEE Computer Society, 1961.

[15] Y. M. Elyada and D. Horn. Can dynamic neural filters produce pseudo-random sequences? In *Artificial Neural Networks: Biological Inspirations (ICANN) 2005*, pages 211–216. Springer, 2005.

[16] A. M. Frieze, J. Hastad, R. Kannan, J. C. Lagarias, and A. Shamir. Reconstructing truncated integer variables satisfying linear congruences. *SIAM Journal on Computing*, 17(2):262–280, 1988.

[17] M. Gardner. Mathematical games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223(4):120–123, 1970.

[18] O. Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools.* Cambridge University Press, 2007.

[19] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32. ACM, 1989.

[20] O. Goldreich, N. Nisan, and A. Wigderson. On Yaos XOR-lemma. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 273–301. Springer, 2011.

[21] M. Goldsmith. The maximal Lyapunov exponent of a time series. Master's thesis, Concordia University Montreal, Quebec, Canada, 2009.

[22] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[23] E. Goles and S. Martinez. Exponential transient classes of symmetric neural networks for synchronous and sequential updating. *Complex Systems*, 3(6):589–597, 1989.

[24] C. Gotsman and N. Linial. Spectral properties of threshold functions. *Combinatorica*, 14(1):35–50, 1994.

[25] D. O. Hebb. *The Organization of Behavior.* New York: Wiley, 1949.

[26] G. E. Hinton and T. J. Sejnowski. Optimal perceptual inference. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, Washington DC*, pages 448–453, 1983.

[27] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[28] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[29] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-sate neurons. *Proceedings of the National Academy of Sciences*, 81:3088–3092, 1984.

[30] J. M. Hughes. Pseudo-random number generation using binary recurrent neural networks. Bachelor's thesis, Kalamazoo College, 2007.

[31] L. D. Iasemidis, J. C. Sackellares, H. P. Zaveri, and W. J. Williams. Phase space topography and the Lyapunov exponent of electrocorticograms in partial seizures. *Brain Topography*, 2(3):187–201, 1990.

[32] L. D. Iasemidis, D.-S. Shiau, J. C. Sackellares, P. M. Pardalos, and A. Prasad. Dynamical resetting of the human brain at epileptic seizures: application of nonlinear dynamics and global optimization techniques. *IEEE transactions on bio-medical engineering*, 51(3):493–506, 2004.

[33] J. Katz and Y. Lindell. *Introduction to Modern Cryptography: Principles and Protocols.* Chapman and Hall/CRC, 2007.

[34] L. G. Khachiyan. A polynomial algorithm in linear programming. *(Russian), Doklady Akademii Nauk SSSR*, (244):1093–1096, 1979.

[35] Knuth. *The Art Of Computer Programming, Volume 2: Seminumerical Algorithms, 3/E.* Pearson Education, 1998.

[36] H. Krawczyk. How to predict congruential generators. *Journal of Algorithms*, 13(4):527–545, 1992.

[37] A. D. Krystal, C. Zaidman, H. S. Greenside, R. D. Weiner, and C. E. Coffey. The largest Lyapunov exponent of the EEG during ECT seizures as a measure of ECT seizure adequacy. *Electroencephalography and Clinical Neurophysiology*, 103(6):599–606, 1997.

[38] P. L'Ecuyer. Uniform random number generation. *Annals of Operations Research*, 53(1):77–120, 1994.

[39] P. L'Ecuyer. Maximally equidistributed combined Tausworthe generators. *Mathematics of Computation*, 65(213):203–213, 1996.

[40] P. L'Ecuyer and F. Panneton. $\mathbf{F}_2$-linear random number generators. In *Advancing the Frontiers of Simulation: A Festschrift in Honor of George Samuel Fishman*, pages 169–193. Springer-Verlag, 2009.

[41] P. L'Ecuyer and R. Simard. TestU01: A C library for empirical testing of random number generators. *ACM Trans. Math. Softw.*, 33(4), 2007.

[42] P. L'Ecuyer and R. Simard. TestU01: A software library in ANSI C for empirical testing of random number generators. user's guide, compact version. http://simul.iro.umontreal.ca/testu01/tu01.html, 2009.

[43] K. Lehnertz. Epilepsy: extreme events in the human brain. In *Extreme events in nature and society*, pages 123–143. Springer, 2006.

[44] K. Lehnertz, R. G. Andrzejak, J. Arnhold, T. Kreuz, F. Mormann, C. Rieke, G. Widman, and C. E. Elger. Nonlinear EEG analysis in epilepsy: its possible use for interictal focus localization, seizure anticipation, and prevention. *Journal of Clinical Neurophysiology: Official Publication of the American Electroencephalographic Society*, 18(3):209–222, 2001.

[45] L. A. Levin. One way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

[46] A. Liu, J. S. Hahn, G. P. Heldt, and R. W. Coen. Detection of neonatal seizures through computerized EEG analysis. *Electroencephalography and Clinical Neurophysiology*, 82(1):30–37, 1992.

[47] F. Lopes da Silva, W. Blanes, S. N. Kalitzin, J. Parra, P. Suffczynski, and D. N. Velis. Epilepsies as dynamical diseases of brain systems: basic models of the transition between normal and epileptic activity. *Epilepsia*, 44 Suppl 12:72–83, 2003.

[48] G. Marsaglia. A current view of random number generators. In *Computer Science and Statistics, Sixteenth Symposium on the Interface. Elsevier Science Publishers, North-Holland, Amsterdam*, pages 3–10, 1985.

[49] G. Marsaglia. Diehard: a battery of tests of randomness. http://stat.fsu.edu/~geo/diehard.html, 1996.

[50] G. Marsaglia. Xorshift rngs. *Journal of Statistical Software*, 8(14):1–6, 2003.

[51] G. Marsaglia and L.-H. Tsay. Matrices and the structure of random number sequences. *Linear Algebra and its Applications*, 67:147–156, 1985.

[52] M. Matsumoto and Y. Kurita. Strong deviations from randomness in m-sequences based on trinomials. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 6(2):99–106, 1996.

[53] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.

[54] M. Matsumoto and T. Nishimura. A nonempirical test on the weight of pseudorandom number generators. In *Monte Carlo and Quasi-Monte Carlo methods 2000*, pages 381–395. Springer, 2002.

[55] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[56] M. L. Minsky and S. A. Papert. *Perceptrons.* MIT Press, 1969.

[57] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[58] S. Muroga. *Threshold Logic and its Applications.* Wiley-Interscience New York, 1971.

[59] S. Muroga, T. Tsuboi, and C. R. Baugh. Enumeration of threshold functions of eight variables. *IEEE Transactions on Computers*, C-19(9):818–825, 1970.

[60] H. Ocak. Automatic detection of epileptic seizures in EEG using discrete wavelet transform and approximate entropy. *Expert Systems with Applications*, 36(2, Part 1), 2009.

[61] R. O'Donnell. *Analysis of Boolean Functions.* Cambridge University Press, June 2014.

[62] F. Panneton. *Construction d'ensembles de points basée sur des récurrences linéaires dans un corps fini de caractéristique 2 pour la simulation Monte Carlo et l'intégration quasi-Monte Carlo.* PhD thesis, Université de Montréal, 2004.

[63] C. Robinson. *Dynamical Systems: Stability, Symbolic Dynamics, and Chaos.* CRC Press, Boca Raton, Fla, 1998.

[64] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

[65] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, National Institute of Standards and Technology (NIST), 2001.

[66] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[67] K.-Y. Siu, V. Roychowdhury, and T. Kailath. *Discrete Neural Computation: A Theoretical Foundation.* Prentice Hall PTR, 1995.

[68] J. C. Sprott. *Chaos and Time-Series Analysis.* Oxford University Press, 2003.

[69] J. Stern. Secret linear congruential generators are not cryptographically secure. In *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pages 421–426. IEEE, 1987.

[70] F. Takens. Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, Warwick 1980*, number 898 in Lecture Notes in Mathematics, pages 366–381. Springer Berlin Heidelberg, 1981.

[71] R. C. Tausworthe. Random numbers generated by linear recurrence modulo two. *Mathematics of Computation*, 19(90):201–209, 1965.

[72] S. Tezuka and P. L'Ecuyer. Efficient and portable combined Tausworthe random number generators. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 1(2):99–112, 1991.

[73] T. Ueda. Circular nonsingular threshold transformations. *Discrete Mathematics*, 105:249–258, 1992.

[74] T. Ueda. An enhanced Arimoto Theorem on threshold transformations. *Graphs and Combinatorics*, 17(2):343–351, 2001.

[75] D. K. Wang and A. Compagner. On the use of reducible polynomials as random number generators. *Mathematics of Computation*, 60(201):363–374, 1993.

[76] C. Werndl. Are deterministic descriptions and indeterministic descriptions observationally equivalent? *Studies in History and Philosophy of Science Part B: Studies in History and Philosophy of Modern Physics*, 40(3):232–242, 2009.

[77] B. Widrow and M. E. Hoff. Adaptive switching circuits. *1960 IRE Western Electric Show and Convention Record, Part 4*, pages 96–104, 1960.

[78] R. Winder. Partitions of n-space by hyperplanes. *SIAM Journal on Applied Mathematics*, 14(4):811–818, 1966.

[79] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano. Determining Lyapunov exponents from a time series. *Physica*, pages 285–317, 1985.

[80] A. C. Yao. Theory and application of trapdoor functions. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 80–91. IEEE, 1982.

[81] N. Zierler and J. Brillhart. On primitive trinomials (mod 2), II. *Information and Control*, 14(6):566–569, 1969.

Appendix A

Linear Algebra

Theorem A.1 (Gordan's Theorem):
For any $m \times n$ matrix A precisely one of the following statements is true:

$$\exists\, \mathbf{x} \in \mathbf{R}^n \textit{ such that } A\mathbf{x} > \mathbf{0}; \tag{A.1}$$

or

$$\exists\, \mathbf{y} \in \mathbf{R}^m \textit{ such that } A^T\mathbf{y} = \mathbf{0} \textit{ where } \mathbf{y} > \mathbf{0} \textit{ and } \mathbf{y} \neq \mathbf{0}. \tag{A.2}$$

Proof:
We will use the Duality Theorem from Linear Programming (see [1]) to prove this. Let $\mathbf{e} = (1, 1, \ldots, 1)$ in $\mathbf{R}^m$. Consider the linear programming problem with variables $y_1, y_2, \ldots, y_m$ given by

$$\begin{array}{rr} \max & \mathbf{e}^T\mathbf{y} \\ \text{s.t.} & A^T\mathbf{y} = \mathbf{0}; \\ & \mathbf{y} \le \mathbf{e}; \\ & \mathbf{y} \ge \mathbf{0}. \end{array}$$

The corresponding dual problem has variables $x_1, x_2, \ldots, x_n$ and $z_1, z_2, \ldots, z_m$ and is given by

$$\begin{array}{rr} \min & \mathbf{e}^T\mathbf{z} \\ \text{s.t.} & A\mathbf{x} + \mathbf{z} \ge \mathbf{e}; \\ & \mathbf{z} \ge \mathbf{0}. \end{array}$$

The dual problem has a feasible solution ($\mathbf{x} = \mathbf{0}$, $\mathbf{z} = \mathbf{e}$, for example) and its objective function is bounded below by 0. Therefore, it has an optimal solution $\mathbf{z}^*$ such that

$\mathbf{e}^T\mathbf{z}^* \geq 0$. If (A.1) does not hold, then $z_i^* > 0$ for some i, and we have $\mathbf{e}^T\mathbf{z}^* > 0$. By the Duality Theorem, the primal problem has an optimal solution $\mathbf{y}^*$ such that $\mathbf{e}^T\mathbf{y}^* = \mathbf{e}^T\mathbf{z}^* > 0$. We conclude that (A.2) holds.

Conversely, suppose (A.1) holds. Then there exists an $\mathbf{x}^*$ such that $A\mathbf{x}^* \geq \mathbf{e}$. Therefore, the dual has an optimal solution and its value is 0. By duality, the primal problem has an optimal solution $\mathbf{y}^*$ such that $\mathbf{y}^* = \mathbf{0}$. Thus, if $\mathbf{y}$ is any feasible solution to the primal, then it must be such that $\mathbf{e}^T\mathbf{y} \leq \mathbf{e}^T\mathbf{y}^* = 0$, so that $\mathbf{y} = \mathbf{0}$. This show that (A.1) does not hold. $\square$

References

[1] V. Chvátal, *Linear programming*, W. H. Freeman, September 1983.

Appendix B

Shift Register Sequences

In this section we characterize shift register sequences with maximal period length. A classic text on this topic is that of Golomb's [2], but the theory dates back much further (see chapter 17 of [1]). The bulk of this section can be found in [3], but is reproduced here in our specific setting: we restrict ourselves to the field $\mathbf{F}_2$. Thus, all operations in this section are taken modulo 2.

Definition B.1:
Let $x_0, x_1, x_2, \ldots$ be a sequence. If there exists integers p and t_0 such that

$$x_{t+p} = x_t \text{ for all } t \geq t_0, \tag{B.1}$$

then the sequence is *ultimately periodic* and p is a *period* of the sequence. The smallest number among all possible periods is the *least period* of the sequence. The least positive integer t_0 for which (B.1) with p the least period holds is the *preperiod* of the sequence. An ultimately periodic sequence is called *periodic* if its preperiod is 0.

Lemma B.1:
Every period of an ultimately periodic sequence is divisible by the least period.

Proof:
Let $x_0, x_2, x_2, \ldots$ be an ultimately periodic sequence with least period p and let r be an arbitrary period of the sequence. Then there exist t_0 and t_1 such that

$$x_{t+r} = x_t \quad \text{for } t \geq t_0;$$

$$x_{t+p} = x_t \quad \text{for } t \geq t_0.$$

Suppose $p \nmid r$. Then there exist integers b and c such that $r = bp + c$, where $b \geq 1$ and $0 < c < p$. Then for $t \geq \max(t_0, t_1)$ we have

$$x_t = x_{t+r} = x_{t+bp+c} = x_{t+(b-1)p+c} = x_{t+(b-2)p+c} = \cdots = x_{t+c}.$$

This implies that c is a period of the sequence, but $c < p$ contradicts the fact that p is the least period of the sequence. □

Definition B.2:
Let k be a positive integer and let $a_1, a_2, \ldots, a_{k-1}$ be in $\{0, 1\}$. A sequence $x_0, x_1, x_2, \ldots$ satisfying

$$x_{t+k} = a_{k-1}x_{t+k-1} + a_{k-2}x_{t+k-2} + \cdots + a_1 x_{t-1} + x_t \pmod 2 \tag{B.2}$$

is a *k-th order linear recurrence sequence.* The values $x_0, x_1, \ldots, x_{k-1}$ are the *seeds* of the sequence.

Definition B.3:
For a k-th order linear recurrence sequence $x_0, x_1, x_2, \ldots$, the vector

$$\mathbf{x}_t = (x_t, x_{t+1}, \ldots, x_{t+k-1})$$

is the *state vector at time t.*

Lemma B.2:
Every k-th order linear recurrence sequence is ultimately periodic with least period at most $2^k - 1$.

Proof:
Consider the state vectors $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ of the associated sequence. If $\mathbf{x}_t$ is the zero vector for some t, then the proof is finished, since the sequence eventually consists of only zeros and thus has period 1. Thus, we may assume that no state vector is the zero vector. Consider the 2^k state vectors $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{2^k-1}$. Since there are $2^k - 1$ distinct bit vectors of length k that are not the zero vector, there exists i and j such

that $0 \leq i < j \leq 2^k - 1$ and $\mathbf{x}_i = \mathbf{x}_j$. The recurrence (B.2) then yields $\mathbf{x}_{t+j-i} = \mathbf{x}_t$ for $t \geq i$. Noting that $j - i \leq 2^k - 1$, we conclude that the sequence is ultimately periodic with least period at most $2^k - 1$. □

Corollary B.1:
Every k-th order linear recurrence sequence is periodic.

Proof:
Let $x_0, x_1, x_2, \ldots$ be a k-th order linear recurrence sequence. By Lemma B.2, the sequence is ultimately periodic. Let p be the least period of the sequence, and let t_0 be the preperiod of the sequence. Then we have $x_{t+p} = x_t$ for $t \geq t_0$. Suppose $t_0 \geq 1$ and set $t = t_0 + p - 1$. Then by (B.2), we have

$$\begin{aligned} x_{t_0-1+p} &= x_{t_0-1+p+k} - a_{k-1}x_{t_0-1+p+k-1} - a_{k-2}x_{t_0-1+p+k-2} - \cdots - a_1 x_{t_0-1+p+1} \\ &= x_{t_0-1+k} - a_{k-1}x_{t_0-1+k-1} - a_{k-2}x_{t_0-1+k-2} - \cdots - a_1 x_{t_0-1+1} \end{aligned}$$

Similarly, by (B.2) we have

$$x_{t_0-1} = x_{t_0-1+k} - a_{k-1}x_{t_0-1+k-1} - a_{k-2}x_{t_0-1+k-2} - \cdots - a_1 x_{t_0-1+1},$$

so that $x_{t_0-1+p} = x_{t_0-1}$, which contradicts the definition of the preperiod. □

With every k-th order linear recurrence sequence obeying

$$x_{t+k} = a_{k-1}x_{t+k-1} + a_{k-2}x_{t+k-2} + \cdots + a_1 x_{t-1} + x_t \pmod 2,$$

we associate a $k \times k$ matrix of bits defined by

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \cdots & 0 \\ & & & \vdots & & & \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & a_1 & a_2 & \cdots & a_{k-3} & a_{k-2} & a_{k-1} \end{bmatrix}.$$

We shall refer to A as the *recurrence matrix* of the associated linear recurrence sequence. Note that the state vectors $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ obey

$$\mathbf{x}_{t+1} = A\mathbf{x}_t.$$

Lemma B.3:
Consider a k-th order linear recurrence relation and seeds $x_0 = x_1 = x_2 = \cdots = x_{k-1} = 0$ and $x_k = 1$. The k state vectors $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{k-1}$ form a basis for the vector space of k-dimensional bit strings.

Proof:
It is enough to show that the k vectors $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{k-1}$ are linearly independent. Suppose that $c_0\mathbf{x}_0 + c_1\mathbf{x}_1 + \cdots + c_{k-1}\mathbf{x}_{k-1} = \mathbf{0}$ for bits $c_0, c_1, \ldots, c_{k-1}$. We will show that $c_{k-1-i} = 0$ for $i = 0, 1, \ldots, k-1$ by induction on i. When $i = 0$, we must have $c_{k-1} = 0$ since $\mathbf{x}_{k-1}$ is the only state vector with a 1 in its first coordinate. Now assume that $c_{k-1} = c_{k-2} = \cdots = c_{k-1-(i-1)} = 0$. Then we must also have $c_{k-1-i} = 0$ since $\mathbf{x}_{k-1-i}$ is the only state vector in $\{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{k-1-i}\}$ with a 1 in coordinate $i + 1$. $\square$

Lemma B.4:
Let A be the recurrence matrix of a k-th order linear recurrence, and consider the seeds $x_0 = x_1 = x_2 = \cdots = x_{k-1} = 0$ and $x_k = 1$. For this particular linear recurrence sequence we have

$$\mathbf{x}_i = \mathbf{x}_j \iff A^i = A^j.$$

Proof:
If $A^i = A^j$ then $\mathbf{x}_i = A^i\mathbf{x}_0 = A^j\mathbf{x}_0 = \mathbf{x}_j$.

Conversely, suppose $\mathbf{x}_i = \mathbf{x}_j$. Then for $t \geq 0$ we have $\mathbf{x}_{i+t} = \mathbf{x}_{j+t}$. Therefore, we have

$$A^i\mathbf{x}_t = A^iA^t\mathbf{x}_0 = A^t\mathbf{x}_i = A^t\mathbf{x}_j = A^jA^t\mathbf{x}_0 = A^j\mathbf{x}_t. \quad \text{for } t \geq 0. \tag{B.3}$$

Now suppose $A^i \neq A^j$. Then there is a non-zero vector v such that $A^iv \neq A^jv$. By Lemma B.3, we can find constants $c_0, c_1, \ldots, c_{k-1}$ such that $v = c_0\mathbf{x}_0 + c_1\mathbf{x}_1 + \cdots + c_{k-1}\mathbf{x}_{k-1}$. Therefore,

$$A^i(c_0\mathbf{x}_0 + c_1\mathbf{x}_1 + \cdots + c_{k-1}\mathbf{x}_{k-1}) \neq A^j(c_0\mathbf{x}_0 + c_1\mathbf{x}_1 + \cdots + c_{k-1}\mathbf{x}_{k-1}),$$

which implies

$$c_0\left(A^i\mathbf{x}_0 - A^j\mathbf{x}_0\right) + c_1\left(A^i\mathbf{x}_1 - A^j\mathbf{x}_1\right) + \cdots + c_{k-1}\left(A^i\mathbf{x}_{k-1} - A^j\mathbf{x}_{k-1}\right) \neq \mathbf{0}.$$

But this constradicts (B.3). $\square$

The set of nonsingular $k \times k$ matrices with entries in $\{0, 1\}$ forms a finite group under matrix multiplication, denoted $GL(k, \mathbf{F}_2)$. Let A be a member of $GL(k, \mathbf{F}_2)$. Since $GL(k, \mathbf{F}_2)$ is finite, so is the subgroup $\{A, A^2, A^3, \ldots\}$. Thus, there is a finite number

m such that A^m is the identity matrix; the smallest positive m for which this holds is the *order* of A, and denoted by $\text{ord}(A)$.

Lemma B.5:
The least period of a linear recurrence sequence divides the order of its associated recurrence matrix, $\text{ord}(A)$.
Proof:
Note that $\det(A) = 1$, so that A is nonsingular and indeed a member of $GL(k, \mathbf{F}_2)$. Let $m = \text{ord}(A)$. Then $\mathbf{x}_{t+m} = A^{t+m}\mathbf{x}_0 = A^t\mathbf{x}_0 = \mathbf{x}_t$, so that m is a period of the sequence. Therefore, the least period of the sequence divides $\text{ord}(A)$ by Lemma B.1.□

Lemma B.6:
Let A be the recurrence matrix of a k-th order linear recurrence relation. When initialized with the seeds $x_0 = x_1 = x_2 = \cdots = x_{k-1} = 0$ and $x_k = 1$, the least period of the resulting sequence is equal to $\text{ord}(A)$.
Proof:
Let p be the least period of the sequence. By Lemma B.5, we know that $p \mid \text{ord}(A)$. Furthermore, $\mathbf{x}_p = \mathbf{x}_0$, so by Lemma B.4 we have $A^p = A^0$. Therefore, $p = \text{ord}(A)$. □

Let M be a square matrix over a field $\mathbf{F}$. We recall the following definitions:

the *characteristic polynomial* of M, denoted by χ_M, is the polynomial over $\mathbf{F}$ defined by $\chi_M(z) = \det(z\mathbf{I} - M)$;

the minimal polynomial of M, denoted by μ_M, is the monic polynomial over $\mathbf{F}$ of least degree such that $\mu_M(M) = 0$.

Lemma B.7:
Let A be the recurrence matrix of a k-th order linear recurrence relation. We have $\chi_A = \mu_A$.
Proof:
By definition, we have $\chi_A(z) = z^k - a_{k-1}z^{k-1} - a_{k-2}z^{k-2} - \cdots - a_1 - 1$. By the Cayley-Hamilton Theorem, we know that $\chi_A(A) = 0$, so that χ_A is a candidate for the minimal polynomial of A. We must show that no polynomial of lesser degree has A as a root. To this end, suppose that $m < k$ and that $p(z) = z^m + b_{m-1}z^{m-1} + b_{m-2}z^{m-2} + \cdots + b_1z + b_0$ satisfies $p(A) = 0$. For $i = 1, 2, \ldots, k$, let $\mathbf{e}_i$ be the i-th standard basis vector; e_i has coordinate i equal to 1, all other coordinates are 0. Note that $A\mathbf{e}_i = \mathbf{e}_{i+1}$ for $i = 1, 2, \ldots, m$. Therefore,

$$\begin{aligned} 0 = p(A)\mathbf{e}_1 &= A^m\mathbf{e}_1 + b_{m-1}A^{m-1}\mathbf{e}_1 + \cdots + b_1A\mathbf{e}_1 + b_0\mathbf{e}_1 \\ &= \mathbf{e}_{m+1} + b_{m-1}\mathbf{e}_m + \cdots + b_1\mathbf{e}_2 + b_0\mathbf{e}_1. \end{aligned}$$

This implies that $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_m$ are linearly dependent, which is a contradiction. $\square$

Results from the theory of polynomials over finite fields assures us that the following is well-defined (see the first chapter of [3], for instance).

Definition B.4:
If f is a non-zero polynomial such that $f(0) \neq 0$ then the *order* of f is the least positive integer e such that $f(z) \mid z^e - 1$ over $\mathbf{F}[z]$, and is denoted by $\operatorname{ord}(f)$.

Lemma B.8:
Let A be the recurrence matrix of a k-th order linear recurrence relation. Then we have $\operatorname{ord}(\chi_A) = \operatorname{ord}(A)$.

Proof:
We will show that for every positive integer e, we have $A^e = I$ if and only if $\chi_A(z) \mid z^e - 1$.

First, suppose $A^e = I$. Then $A^e - I = 0$, but also $\chi_A(A) = 0$. Therefore, $\chi_A(z) \mid z^e - 1$ since χ_A is the minimal polynomial of A by Lemma B.7.

Conversely, suppose $\chi_A(z) \mid z^e - 1$. Then $z^e - 1 = \chi_A(z)g(z)$ for some polynomial g. Thus, $A^e - I = \chi_A(A)g(A) = 0$, so that $A^e = I$. $\square$

Corollary B.2:
Let $x_0, x_1, x_2, \ldots$ be a k-th order linear recurrence sequence with recurrence matrix A and least period p. Then $p \mid \operatorname{ord}(\chi_A)$. Furthermore, if $x_0 = x_1 = x_2 = \cdots = x_{k-1} = 0$ and $x_k = 1$, then $p = \operatorname{ord}(\chi_A)$.

Proof:
The first claim follows from Lemma B.5 and Lemma B.8. The second claim follows from Lemma B.6 and Lemma B.8. $\square$

Lemma B.9:
Let $x_0, x_1, x_2, \ldots$ be a k-th order linear recurrence sequence with a period p and recurrence matrix A. Then

$$\chi_A(z)s(z) = (1 - z^p)h(z) \tag{B.4}$$

holds with

$$s(z) = x_0 z^{p-1} + x_1 z^{p-2} + \cdots x_{p-2} z + x_{p-1} \tag{B.5}$$

and

$$h(z) = \sum_{j=0}^{k-1} \sum_{i=0}^{k-1-j} a_{i+j+1} x_i z^j, \tag{B.6}$$

where we set $a_k = a_0 = -1$.

Proof:

Let c_t be the coefficient of z^t on the left-hand side of B.4 and let d_t denote the coefficient of z^t on the right-hand side of B.4. Recall that $\chi_A(z) = -\sum_{i=0}^{k} a_i z^i$. Thus,

$$c_t = -\sum_{\substack{0 \le i \le k \\ 0 \le j \le p-1 \\ i+j=t}} a_i x_{p-1-j} \qquad \text{for } 0 \le t \le k+p-1. \tag{B.7}$$

By the definition of k-th order linear recurrence relations, we have

$$\sum_{i=}^{k} a_i x_{n+i} = 0 \qquad \text{for } n \ge 0. \tag{B.8}$$

We will use equations(B.7) and (B.8) and the periodicity of the sequence to complete the proof in four cases:

If $k \le t \le p-1$, then

$$c_t = -\sum_{i=0}^{k} a_i x_{p-1-t+i} = 0 = d_t.$$

If $t \le p-1$ and $t < k$ then

$$c_t = -\sum_{i=0}^{t} a_i x_{p-1-t+i} = \sum_{i=t+1}^{t} a_i x_{p-1-t+i} = \sum_{i=t+1}^{t} a_i x_{-1-t+i} = \sum_{i=0}^{k-1-t} a_{i+t+1} x_i = d_t.$$

If $t \ge p$ and $t \ge k$ then

$$c_t = -\sum_{i=t-p+1}^{k} a_i x_{r-1-t+i} = -\sum_{i=0}^{k-1-t+r} a_{i+t-p+1} x_i = d_t.$$

If $p \le t < k$ then

$$c_t = -\sum_{i=t-p+1}^{t} a_i x_{p-1-t+i} = -\sum_{i=0}^{p-1} a_{i+t-p+1} x_i = -\sum_{i=p}^{k-1-t+p} a_{i+t-p+1} x_i - \sum_{i=0}^{k-1-t+p} a_{i+t-p+1} x_i$$
$$= -\sum_{i=0}^{k-1-t} a_{i+t+1} x_{i+p} - \sum_{i=0}^{k-1-t+p} a_{i+t-p+1} x_i = d_t$$

since $x_{i+p} = x_i$. $\square$

Lemma B.10:
Consider a k-th order linear recurrence sequence with recurrence matrix A. If $\chi_A(z)$ is irreducible then then the least period of the sequence is equal to $\operatorname{ord}(\chi_A)$.

Proof:
On the one hand, $p \mid \operatorname{ord}(\chi_A)$ by the first part of Corollary B.2. On the other hand, by Lemma B.9, we have $\chi_A(z)s(z) = (1 - z^p)h(z)$, so that

$$\chi_A(z) \mid (z^p - 1)h(z).$$

Since s is non-zero, so is h. Furthermore, the degree of h is less than the degree of χ_A. Therefore $\chi_A(z) \mid z^p - 1$ by the irreducibility of $\chi_A(z)$. This implies that $p \geq \operatorname{ord}(\chi_A)$. □

To wrap things up, we will need the notion of a primitive polynomials. There are many characterisations of primitive polynomials, we will use the following:

Definition B.5:
A polynomial f of degree k is *primitive* if it is monic, $f(0) \neq 0$ and $\operatorname{ord}(f) = 2^k - 1$.

It is well known and that primitive polynomials are irreducible. (see Chapter 2 of [3]).

Theorem B.1:
Let $x_0, x_1, x_2, \ldots$ be a k-th order linear recurrence sequence with non-zero seed and recurrence matrix A. If $\chi_A(z)$ is primitive then the sequence has least period as large as possible, that is the least period is equal to $2^k - 1$.

Proof:
Since $\chi_A(z)$ is primitive, $\operatorname{ord}(\chi_A) = 2^k - 1$. Furthermore, $\chi_A(z)$ is irreducible, so the least period of the sequence is equal to $\operatorname{ord}(\chi_A)$ by Lemma B.10. The maximality of the sequence length comes from Lemma B.2. □

References

[1] Leonard E. Dickson, *History of the Theory of Numbers*, Washington, Carnegie Institution of Washington, 1919.

[2] Solomon W. Golomb, *Shift Register Sequences*, Aegean Park Pr, June 1981.

[3] Rudolf Lidl and Harald Niederreiter, *Finite Fields (Encyclopedia of Mathematics and its Applications)*, Addison-Wesley, 1983.