PARAFINITARY LEARNING

by

DYLAN POZORSKI

(Under the Direction of O. Bradley Bassler)

ABSTRACT

This paper introduces Parafinitary Learning (PL), a learning framework that asserts the primacy of scale in the construction of artificially intelligent agents. This framework is successfully applied as an extension of neural networks and is capable of producing competitive results in simple binary classification tasks. The base neural network implementation of PL employs a special case of Oja's Rule (called Oja's Golden Rule). This special case is then further augmented with two additional mechanisms for improving efficiency and stability. Respectively, this involves employing radix economies, a concept drawn from coding theory, and prediction markets, which cast the models being developed as logical inductors over the target distribution being learned.

INDEX WORDS: Deep Learning, Machine Learning, Neural Networks, Auto-Scaling, Linear Logic, Prediction Market, Parafinite, Dynamical Systems, Golden Ratio, Radix Economy

PARAFINITARY LEARNING

by

DYLAN POZORSKI

B.S., University of Wisconsin - Madison, 2017

A Thesis Submitted to the Graduate Faculty of the
University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2021

©2021

Dylan Pozorski

All Rights Reserved

PARAFINITARY LEARNING

by

DYLAN POZORSKI

Major Professor: O. Bradley Bassler

Committee: Frederick Maier
Sarah Wright

Electronic Version Approved:

Ron Walcott
Vice Provost for Graduate Education and Dean of the Graduate School
The University of Georgia
August 2021

DEDICATION

I dedicate this work, firstly, to my parents for encouraging me to work hard, have fun, and be creative; I would not be the person I am today without your guidance and support. Secondly, I dedicate this work to my brother and sister; our lifelong friendship is something I would not trade for the world. Lastly, I dedicate this work to my nieces and nephew; you function as a constant reminder of why I am so optimistic about the future and have provided me more joy than I could have ever anticipated experiencing.

ACKNOWLEDGMENTS

Dr. O. Bradley Bassler: I want to thank (and/or curse) you for challenging me more academically and personally than any other person in my professional life. The last two years of this program have been decorated by both your passion to better understand the world and compassion for your students. I am indebted to you for the many long nights talking about Wittgenstein, Cavell, logic, math, and poetry. It is unlikely the university recognizes (or is even capable of recognizing) the void left behind by your retirement, but I am so excited to see what this new chapter brings. I am privileged to have you as a mentor and friend, and look forward to continuing our investigation of the parafinite together!

Dr. Fred Maier: Thank you so much for your mentorship while moving through this program and while working for the Career Center (and always having high-quality movie recommendations)! When I think of the AI department, you are its backbone; it makes me confident that this department will continue to investigate artificial intelligence holistically and not merely as an engineering problem, as so many are willing to do. It has been a paradoxically long and short two years working together. Though I am glad to be moving on to the next chapter of my life, I want to thank you for being the "dad" of the AI Institute. You have both helped curate a unique and interdisciplinary student experience, while keeping the order (maintaining the AI lab, keeping us apprised of deadlines and requirements, etc.). It is all so very much appreciated!

Dr. Sarah Wright: I consider myself incredibly fortunate to have been a part of your epistemology seminar this last semester and I look forward to staying in touch after graduation. With respect to the various challenges that technology imposes on contemporary society, you impressed upon me that epistemological concerns are at their core and comprehensive solutions, should they be found, will need to address these concerns directly. You have helped shift how I view the internet, privacy, and AI in ways that "applied" classes could never do. I am, and will always be, thankful for this guidance.

Ryan McArdle, Tangrui Li, Yiren Zhang, and Zach Peck: It has been such a pleasure getting to know all of you. Whether through Zoom/phone calls, emails, late nights at Boyd, or discussions over drinks, a shared thread of curiosity has always been identifiable and binding between us all. It has been a heartening experience to have developed our relationships through a mutual discomfort with the prevailing views and created a small community to test our ideas. I am privileged to have been in the same cohort with you all and hope to some day be colleagues together.

Cohort/AI Institute: Thank you to all of the other professors and students who contribute to the life and culture of the AI Institute. Undertaking this program has been an incredibly formative experience; it provided a place where challenging the technical, conceptual, and philosophical standard is an expectation. Thank you for a life-changing experience!

TABLE OF CONTENTS

Acknowledgments **v**

List of Figures **vii**

List of Tables **viii**

1 Distributed Learning Rules **1**
1.1 Hebbian Learning and Oja's Rule 1
1.2 Conceptual Prelude to Oja's Golden Rule 5
1.3 Oja's Golden Rule 9

2 Radix Economy **16**
2.1 Finding the Optimal Radix Economy 16
2.2 Division, Relativization, and other Implementation Questions 22

3 Experiment **26**
3.1 Technology 26
3.2 Data 27
3.3 Experiment 28
3.4 Results 30
3.5 Discussion 32

4 Prediction Market **35**
4.1 Introduction and Conceptual Distinctions 35
4.2 Formulation 37
4.3 (A Characterization of) Implementation 42
4.4 Conclusion and Preliminary Results 46

5 Conclusion **48**

References **49**

LIST OF FIGURES

1.3.1 Graph of the aggregate behavior (Equation 1.3.7) of the incoming edge weights to a neuron. The x-axis contains the input stimulus values and the y-axis reports the consequent aggregate weight value (Equation 1.3.6) given the same stimulus. This graph shows that both inputs of 0 and 1 are stable (i.e. $y(n+1) = 1$, if $y(n) = 1$ and $y(n+1) = 0$, if $y(n) = 0$). The green coloring is meant to indicate regions where the behavior of the graph does not oscillate and the red coloring is meant to indicate unstable regions. The red, dotted lines denote the identity equations $y = x$ and $x = 1$, respectively. 13

1.3.2 The long-term behavior of graph shifts input into a weighting scheme that roughly mirrors the step-function over a restricted domain. In this way, the system has an intrinsic activation pattern that does not have to be explicitly defined as the Heaviside. The blue lines represent the attractor that the corresponding input evolves to in the long run. . . 15

2.1.1 The above graph illustrates the empirically identified optimal radix value, Ω, (blue line) given the associated limit n. The top and bottom horizontal lines (Euler's number (orange) and ϕ^2 (green)) represent bases that are equidistant from the asymptotically defined optimal radix (purple line). 20

2.2.1 Each of the lines represent possible instantiations of Equation 1.3.5 under the relativizing conditions Equation 2.2.1 and 2.2.2. 23

2.2.2 The division strategy of the graph implementing Oja's Golden Rule with an optimal radix encoding. The maximal growth rate of the graph is ϕ^2 as is evident by the exponentially growing capacity indicated above. 24

3.2.1 The binary class datasets being evaluated. They are linearly separable (top), moons (middle), and circular (bottom). 27

3.4.2 Visualization of the classification made by each of the ensemble methods. The top row is the ground truth being predicted. The coloring indicates the classification type: True Negative (Blue), True Positive (Red), False Negative (Green), and False Positive (Yellow). 34

LIST OF TABLES

3.4.1 The summary of the ensemble and comparison classifier performance on the three data sets being evaluated. 30

CHAPTER 1

DISTRIBUTED LEARNING RULES

§ 1.1 Hebbian Learning and Oja's Rule

Hebbian learning is a distributed learning scheme first formulated in 1949 by Donald Hebb [1]. The motivation in the development of this scheme is to provide an account of neural plasticity, the ability of the nervous systems to change in response to an input stimulus, as found in biological systems [2]. Similar in motivation to this thesis, there exist contemporary adaptations that attempt to merge standard signal processing techniques like least mean square (LMS) with Hebbian learning [3].

The guiding intuition behind the Hebbian formalization is that repeated firings of neighboring neurons should be indicative of a strengthening of their relationship, while mismatched firings should correspond to a weak association. This intuition is captured in the concise conditional expression, "neurons wire together, if they fire together," coined by Singer and Löwel in the early 90's [4].

Formally, the Hebbian learning rule is defined as follows:

$$o_j = f(\sum_{i=1}^{n} w_{ij} x_i) \tag{1.1.1}$$

$$\Delta w_i = \eta x_i o_j \tag{1.1.2}$$

$$w_i(n+1) = w_i(n) + \Delta w_i \tag{1.1.3}$$

In the above equations, o_j is the output of node j with input stimuli coming from all presynaptic nodes i. Each node i contributes w_{ij} of its activation x_i as input to the postsynaptic node j. Given this aggregate input stimulus, the activation of node j is accounted for by some activation function f. Assuming that the activation function is linear, as is the case for the original formulation of Hebb's Rule, the change in the edge weight, Δw_i, is equal to $\eta x_i o_j$, where η is the selected learning rate.

From this formulation, it is clear that $\Delta w_i \propto x_i o_j$, meaning that edge weights are strengthened when adjacent (pre- and postsynaptic) neurons are activated [5]. This quality is especially visible when considering boolean networks where x_i, $o_j \in \{0, 1\}$. A visible drawback of this proportional change is a resultant instability in the network that causes edge weights to increase or decrease at an exponential rate when there exists a dominant signal in the input stimulus [6]. This dominant signal leads to repeated activations between the pre- and postsynaptic neurons.

The major drawback of the Hebbian scheme is that it is unstable when presented with inputs that have a dominant signal. That is, due to the increased associative or dissociative updates provided neighboring signals, the weights can either grow or decrease exponentially fast, approaching infinity after repeated stimulation.

Assume, for example, that there is a dominant input signal x_i that is weighted sufficiently to activate the postsynaptic neuron itself. Consequently, the output of node j is at a minimum proportional to the input from the presynaptic node i (i.e., $o_j \propto x_i w_i(n)$, minimally). Following from this observation, $\Delta w_i \propto x_i o_j \propto x_i(x_i w_i(n))$. So, a lower bound on the edge weight change associated with a dominant signal can be stated as $\Delta w_i \propto x_i(x_i w_i(n)) = \alpha x_i^2 w_i(n)$. Therefore, the following evolution of the edge weight w_i is observed:

$$
\begin{aligned}
w_i(n+1) &= w_i(n) + \Delta w && \text{(Step 1)} \\
&= w_i(n) + \alpha x_i^2 w_i(n) && \text{(Step 2)} \\
&\propto w_i(n)[1 + \alpha x_i^2] && \text{(Step 3)} \\
&\propto w_i(0)[1 + \alpha x_i^2]^n && \text{(Step 4)}
\end{aligned}
$$

To accommodate the issue of edge weight instability, alternatives such as Oja's Rule are employed, since it uses a normalizing condition to limit the magnitude of edge weights between 0 and 1 [7]. The derivation of this rule can be seen as a simple modification of Hebb's Rule that divides the weight update suggested in the prior scheme by the *p-norm* of all of the weight updates associated with the target neuron j. The *p-norm* of the weight updates is defined below:

$$
||\Delta w||_p = (\sum_{k=1}^{m} [w_k(n) + \eta x_k o_j]^p)^{\frac{1}{p}} \tag{1.1.4}
$$

The resultant update rule is,

$$
w_i(n+1) = \frac{w_i(n) + \Delta w_i}{||\Delta w||_p} \tag{1.1.5}
$$

The original formulation of this rule used the Euclidean norm, setting $p = 2$ [7]. The power-series expansion of this update rule results in the following "simplification" [7]:

$$
w_i(n+1) = \frac{w_i(n)}{(\sum_{k=1}^{m} w_k^p(n))^{\frac{1}{p}}} + \eta(\frac{o_j x_i}{(\sum_{k=1}^{m} w_k^p(n))^{\frac{1}{p}}} - \frac{w_i(n) \sum_{k=1}^{m} o_j x_k w_k^{p-1}(n)}{(\sum_{k=1}^{m} w_k^p(n))^{\frac{(p+1)}{p}}}) + O(\eta^2) \tag{1.1.6}
$$

The benefit of this expansion is two-fold: 1) for a small learning rate, η, $O(\eta^2)$ goes to zero and 2) the definition of o_j can allow for a further simplification [7]. In this case, define o_j as follows:

$$o_j = \sum_{i=1}^{n} w_{ij} x_i \tag{1.1.7}$$

This change is a proportional shift to the linear activation function previously considered and not altogether a massive change from the prior definition considered in Hebb's Rule. With the condition that the presynaptic weights normalize to one, it is possible to reach the final derivation of Oja's Rule. Explicitly, this normalization condition is:

$$||\Delta w||_p = (\sum_{i=1}^{n} w_i^p)^{\frac{1}{p}} = 1 \tag{1.1.8}$$

Hence, the derivation (note: $p = 2$):

$$w_i(n+1) = \frac{w_i(n)}{(\sum_{k=1}^{m} w_k^p(n))^{\frac{1}{p}}} + \eta(\frac{o_j x_i}{(\sum_{k=1}^{m} w_k^p(n))^{\frac{1}{p}}} - \frac{w_i(n)\sum_{k=1}^{m} o_j x_k w_k^{p-1}(n)}{(\sum_{k=1}^{m} w_k^p(n))^{\frac{(p+1)}{p}}}) \qquad \text{(Step 1)}$$

$$= \frac{w_i(n)}{1} + \eta(\frac{o_j x_i}{1} - \frac{w_i(n)\sum_{k=1}^{m} o_j x_k w_k^{p-1}(n)}{((\sum_{k=1}^{m} w_k^p(n))^{\frac{1}{p}})^{(p+1)})}) \qquad \text{(Step 2)}$$

$$= w_i(n) + \eta o_j(x_i - \frac{w_i(n)\sum_{k=1}^{m} x_k w_k(n) w_k^{p-2}(n)}{1^{(p+1)}}) \qquad \text{(Step 3)}$$

$$= w_i(n) + \eta o_j(x_i - w_i(n)\sum_{k=1}^{m} x_k w_k(n) w_k^0(n)) \qquad \text{(Step 4)}$$

$$= w_i(n) + \eta o_j(x_i - w_i(n) o_j) \qquad \text{(Step 5)}$$

Replacing o_j with y yields Oja's Rule:

$$w_i(n+1) = w_i(n) + \eta y(x_i - (w_i(n)y)) \tag{1.1.9}$$

Erik Oja's original formulation of this rule showed that the weight vector that the applying neuron converges to is the first principal component of the input vector.

§ 1.2 Conceptual Prelude to Oja's Golden Rule

Conceptually, learning schemes such as those considered by Hebbs, Oja, and Sanger (who is credited with inventing the Generalized Hebbian Algorithm [8]) are concerned with the widely studied phenomenon of self-organization in computing systems. This focus is a continuation of the process started early on by Turing [9]. Yet, it should be noted that the distributed schemes currently being discussed are attempts at accounting for patterns of self-organization in a way that is "lighter-weight" in nature than many of the contemporary alternatives (e.g., Deep Learning).

This characterization of being "light-weight" is not rigorously defined here, but it is fundamentally concerned with the notion of what it means for "copying" to be a granted, or structurally provided, primitive procedure of the logical system being analyzed [10]. This observation is associated with the structural rules of "weakening" and "strengthening" identified in proof theory [11] and, ultimately, what it means for there to be "conservation" or "harmony" in a dynamical system [10]. To be explicit, weakening and strengthening (or contraction) are defined explicitly below, followed by a clarifying example of how these rules may violate what are called "harmony constraints" [12].

$$\frac{A \vdash B}{A, C \vdash B} \quad (weakening) \tag{1.2.1}$$

$$\frac{A, A \vdash B}{A \vdash B} \quad (strengthening) \tag{1.2.2}$$

What Equation 1.2.1 is meant to show is that the system transitions from only requiring resource A to yield some result B, to also requiring a second resource C to produce the same result. In this way, the system is now conditioned upon a larger number of prerequisites to produce the intended product B. The inverse is considered in the case of strengthening, which migrates the system into a context that requires fewer resources to produce the same effect.

The substructural logic known as Linear Logic [12, 13], developed by Jean-Yves Girard, explicitly recognizes this modality in the persistence operator ("!"), read "bang", which allows for the indefinite license to use the resource prepended with it (e.g., !$Quarter$ allows for the repeated use of the $Quarter$ premise). In this way, the resource is bracketed as existing in an exponential modality with respect to the rest of the system (i.e. it cannot be used up by the linear consequence relations of the system).

To help clarify, consider an example where a $Quarter$ is needed to purchase a $Coke$ from a vending machine. In most contexts, I will only have a pocketful of change (that is, a finite amount of change) to transact with the environment to obtain my desired result. Imagine that I have 8 quarters, then it means that the operating context contains 8 $Quarter$ premises to obtain 1 $Coke$ and 3 $Quarter$s (i.e., the cost of a $Coke$ is 5 $Quarters$). If weakening or strengthening is invoked, then it disturbs the harmony or conservation of the available resources that are ever available in the system.

$$\frac{8Quarter \vdash Coke, 3Quarter}{7Quarter \vdash Coke, 3Quarter} \quad (strengthening) \tag{1.2.3}$$

In the strengthening case considered above, somehow 7 quarters are able to achieve the same result as 8 quarters. This violates the harmony of the system (i.e., the quarter has spontaneously dematerialized

from the system), because Cokes decidedly cost 5 quarters. Through explicitly recognizing the effects of weakening and strengthening, which are often taken for granted in logical systems, Linear Logic attempts to better model the "physics" or transactional nature of an environment that operates deterministically (e.g., We cannot create or destroy matter without transacting some amount of energy). The use of the exponential modal operator allows for these concerns to be ignored.

Putting these simple examples aside, the exponential modality ("!") is implicit in the definition of formally defined functions and data types [14], because these are constructs that may be invoked indefinitely as generic constructs, while instances of these objects are ephemeral in nature. The issue with generic constructs, however, is that - in the context of formalisms - there is nothing required to maintain them relative to the system. The generic structure of the integer, for example, can be defined in Linear Logic similarly to how integers are defined in λ-Calculus [15], but the definition of what it means to be an integer is, itself, not maintained (intrinsically) by the system. Clift does a reasonable job at defining Turing Machines in Linear Logic [16], but it falls short of supporting the exponential modality.

The concern of this definitional externalism is that the reliance upon exponential objects (those not intrinsic to the system) allows the leak of an indefinite number of possible assumptions/resources into/out of the system being analyzed. In this way, the exponential modality obfuscates or makes tacit calls to strengthening and weakening operations. The conceptual risk of this behavior is that the mathematical objects being employed in performing some operation (such as a machine learning algorithm) is tacitly reliant upon an intractable/infinite number of resources in its definition.

As a last example of the point that is being suggested, consider the procedure of backpropagation. Backpropagation requires the persistence of error from the output layer back through the layers of the graph. This setup, therefore, makes error a global construct accessible to the nodes of the graph. Yet, this framing begs the question of how this error is defined, where it is defined (at what layer), and the

mechanism by which it persists backwards. It is only through the application of contextually external operations on the system that it is able to retroactively assign error to the nodes of any given layer, but this assignment is an external (exponential ("!")) process in itself. The fact that this process is assigned an exponential modality means that we believe there to exist a process that could be defined intrinsically (linearly) to the system that would provide this functionality. However, the merit of such existence claims is generally on very shaky grounds - usually the validity of proofs making existence claims have to rely upon the observation of an instance of such a phenomenon in the first place [17].

In short, it is generally argued that this assumption or permissiveness of exponential processes into the logical context being considered is just part of the act of abstraction being performed. This defense is understandable, but if the existence claim being made is taken seriously and such an intrinsic process does not exist (i.e., there is no such constructible instance given the available time or resources), then the coherence of the search or systems being suggested is lost. Further, any systems built upon these incoherent logical systems would themselves be incoherent. That is, they admit contradiction into their context and, hence, can (incoherently) yield any conclusion. To steal a couple of lines from the famed Austrian philosopher Ludwig Wittgenstein, the concern is a development of a system of abstraction where "language is on holiday" and the structures of it have the effect of "wheels spinning idly" [18].

Finally, as a disclaimer, this conceptual framing is simply meant as a broad overview of the concerns motivating this project, which is not primarily theoretical in nature. The conceptual "light-weightedness" of Hebb's and Oja's rules is a result of their ability to bypass the dependency of certain global structures (e.g., error).

§ 1.3 Oja's Golden Rule

The disposition that the prior section is meant to reflect is a reservation with overly abstract systems. The consequence of this reservation is a bias toward mathematically simple and distributed learning schemes that attempt to recursively construct the desired behavior, while closely adhering to the physical analog being employed (i.e., non-artificial neural networks).

Accordingly, the derivation of what will be called Oja's Golden Rule is less a derivation than a continued simplification of the abstract objects being employed. Simply being copied from above, Oja's Rule is:

$$w_i(n+1) = w_i(n) + \eta y(x_i - (w_i(n)y)) \tag{1.3.1}$$

The reason that this "copied" equation is relabeled from Equation 1.1.9 to Equation 1.3.1 is that this copied version is tacitly being modified in a way that makes it different from the original. Namely, the change is a contextual break on the assumption that $|\eta| << 1$ employed above to enable the removal of $O(\eta^2)$ in the power expansion above. This migration is not mathematically rigorous, but is being granted to allow for a conceptual simplification in the structure being evaluated.

As such, what is really being considered is a doppelgänger of Oja's Rule, where α is not constrained by the learning rate constraint and distinct from the derivation provided above yielding Equation 1.1.9:

$$w_i(n+1) = w_i(n) + \alpha y(x_i - (w_i(n)y)) \tag{1.3.2}$$

Accordingly the first simplification is that the learning rate, α, will be set to 1, which is a simplification only allowed through the creation of the assumption breaking doppelgänger equation above. This yields:

$$w_i(n+1) = w_i(n) + y(x_i - (w_i(n)y)) \tag{1.3.3}$$

However, it should be stated that the creation of this doppelgänger should not be all that contentious, because there is no explicit limitation on the learning rate encoded into the system. Without such a limitation, any implementation of Oja's Rule (Equation 1.1.9) that does not explicitly limit the learning rate is implicitly using Equation 1.3.2. Of course, it is not mathematically rigorous, but all that is being sought currently is a simple, distributed learning rule and, so, concerns for absolute rigor are being tabled to enable this exploratory process.

The conceptual reason for assigning $\alpha = 1$ is that if the neuron is to be taken as the fundamental unit of learning, then any adjustment that it makes is deterministically prefigured (e.g., A neuron never half learns nor does the weather 0.05 rain when it is in fact raining). In the conceptual program being suggested, it is necessary for these machinations (learning rate, etc.) to be made intrinsic to the systems themselves. Such a large learning rate may be concerning, because it appears prima facie indicative of broad oscillations in the hypothesis space. However, as will be shown, the resultant learning rule will mitigate these concerns to some extent.

The second simplification is likely to be even more contentious, because it suggests the forced equality of the weight and presynaptic activation vectors, i.e. $w_i(n) = x_i$ (for x_i at time n). To those used to the distinction of node activations and edge weights, this suggestion appears incredibly strange and, potentially, incoherent. The response to this concern is to point out the "unharmonious" nature of this distinction. This defense asks the question, "Shouldn't the activation of a node correspond identically

with the edge activations that it produces?". That is, treating neurons as functions being able to tolerate any stimulus (they are total functions) and inhabiting a vector space (as a consequence of their weighted output edges) is a form of dissonance from the guiding analogy of biological neural networks, because the activation of some set of neurons can lead to an infinite number of consequent activations in subsequent layers. For those concerned only with the expressiveness and power of neural networks, this concern or complaint of dissonance is not problematic. Yet, if what is of primary concern is the ability to reason about the objects (such as concerns with producing conceptual black boxes), this distinction is problematic. Since the latter concern is a motivation of this paper (see section 1.2), this suggestion is taken seriously here. It is, however, recognized that this concern is a minority position with respect to most contemporary work.

The equality, $w_i(n) = x_i$, means the activation of a presynaptic neuron only ever yields an influence of $w_i(n)$ to the target neurons. What this identity is meant to impress is that the "activation" of a neuron should not be interpreted as a value itself, but operationally as the system's intrinsic ability to persist or dissipate an incoming source signal. Conceptually, this behavior should be understood as each neuron constructing a relative and local concept of value (more on this below, see chapter 3 section 2). Allowing this equality, the following equation results:

$$
\begin{aligned}
w_i(n+1) &= w_i(n) + y(x_i - (w_i(n)y)) && \text{(Step 1)} \\
&= w_i(n) + y(w_i(n) - (w_i(n)y)) && \text{(Step 2)} \\
&= w_i(n)(1 + y(1-y)) && \text{(Step 3)} \\
&= -w_i(n)(y^2 - y - 1) && \text{(Step 4)}
\end{aligned}
$$

Define the Golden Ratio equation, (y), as follows:

$$\phi(y) = -(y^2 - y - 1) \tag{1.3.4}$$

Finally, it is possible to define Oja's (doppelgänger) Golden Rule:

$$w_i(n+1) = \phi(y) w_i(n) \tag{1.3.5}$$

The intuitive useful qualities/strengths of this learning rule can be observed by graphing the components of this rule and simulating the long-term behavior of implementing neurons. This analysis is done briefly in the next few paragraphs.

With the activation-weighting identity defined above, $w_i(n) = x_i$, it is necessary to update the equation, y, defining the aggregate stimulus/activation affecting a neuron:

$$y = o_j = \sum_{i=1}^{n} w_{ij} \tag{1.3.6}$$

Since y is only a function of the aggregation of input weights to a neuron, it makes the update rule (Equation 1.3.5) only dependent upon the edge weights connected to the postsynaptic neuron j. Since all incoming edges, w_{ij}, to node j will be updated by the scalar yielded by $\phi(y)$, then it must be the case that the sum of the edges to node j also changes by $\phi(y)$. Hence, it is possible to characterize the aggregate behavior of the input weights under some dominant signal. Accordingly, the following aggregate weight update rule can be defined:

$$y(n+1) = \phi(y(n)) y(n) \tag{1.3.7}$$

Of interest, then, is the long-term, aggregate behavior of the input edge weights to a given neuron following repeated application of the update rule. The desired behavior is for the associated graph to reach a stable point as training progresses. Figure 1.3.1, below, helps characterize this behavior.

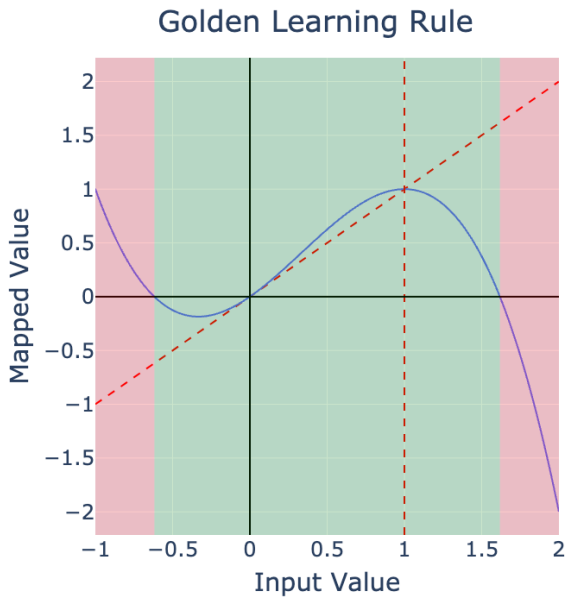

Figure 1.3.1: Graph of the aggregate behavior (Equation 1.3.7) of the incoming edge weights to a neuron. The x-axis contains the input stimulus values and the y-axis reports the consequent aggregate weight value (Equation 1.3.6) given the same stimulus. This graph shows that both inputs of 0 and 1 are stable (i.e. $y(n+1) = 1$, if $y(n) = 1$ and $y(n+1) = 0$, if $y(n) = 0$). The green coloring is meant to indicate regions where the behavior of the graph does not oscillate and the red coloring is meant to indicate unstable regions. The red, dotted lines denote the identity equations $y = x$ and $x = 1$, respectively.

The stablest region of the update rule is between the roots of the update polynomial, which occur at $-\phi^{-1}$ and ϕ, where ϕ is defined:

$$\phi = GoldenRatio = \frac{1+\sqrt{5}}{2} \approx 1.618... \tag{1.3.8}$$

Technically, the update rule is stable along a slightly wider domain that is, roughly, $(-1.18, 1.87)$, but the smaller domain $[-\phi^{-1}, \phi]$ will be opted for in the work completed here. The reason for this choice is that the slightly broader domain results in oscillations in the node's long-term behavior, while $[-\phi^{-1}, \phi]$ has

no such problems. However, any values greater than 1.87 or smaller than -1.18 are decidedly unstable and will result in unbounded weight growth and oscillatory behavior. This unbounded behavior occurs because for $y \geq 1.87$ or $y \leq -1.18$, growth is monotonic over an even number of updates (i.e. $|y(n+2)| \geq |y(n)|$).

Restricting the input domain to $[-\phi^{-1}, \phi]$ avoids the pitfalls of such unstable behavior. Further, it produces convenient long-term output behavior that can be described in the terms used in dynamical systems. Dynamical systems attempt to understand rule-based state transitions such that $x_{n+1} = f(x_n)$ [19]. This state transition function is already provided by Equation 1.3.7, but what is of particular interest are the specific attractors and basins of attraction affecting system convergence in the long run.

Surprisingly, the system restricted on the domain $[-\phi^{-1}, \phi]$ contains two basins of attraction with distinct attractors. Basins of attraction are regions of a domain space that uniquely evolve to a given attractor, points in the weights space that neighboring points (the basin of attraction) asymptotically approach [20, 21]. In the case of Oja's Golden Rule, the long-term behavior produces a dynamical and restricted form of the Heaviside step function.

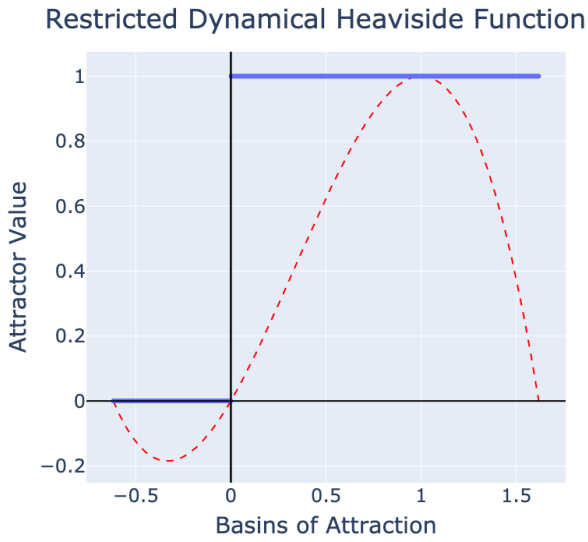

Figure 1.3.2: The long-term behavior of graph shifts input into a weighting scheme that roughly mirrors the step-function over a restricted domain. In this way, the system has an intrinsic activation pattern that does not have to be explicitly defined as the Heaviside. The blue lines represent the attractor that the corresponding input evolves to in the long run.

However, as should be evident, this reliance upon neurons that have a finite input domain is not without its issues. Of particular concern is how out-of-bounds input should be handled. This concern motivates the content of the following section.

CHAPTER 2

RADIX ECONOMY

§ 2.1 Finding the Optimal Radix Economy

As indicated in the prior section, the designation of neurons as operators in a dynamical system with fixed basins of attraction and input constraints (e.g. $x \in [-\phi^{-1}, \phi]$) leads to a slight problem with respect to implementation. Particularly, these constraints lead to neural operators and, hence, the neural network that they define as incompletely covering the input space. For example, what if the input stimulus to a node is 2? Surely this input may be meaningful with respect to the machine learning problem being evaluated and cannot simply be ignored. The solution that will be explored in managing this "coverage" problem will involve dynamic network topologies that self-organize into more comprehensive coverage states.

The algorithm being suggested here is by no means the only scheme for dynamically constructing network topologies. An early example of this sort of strategy is cascade correlation, which iteratively added hidden nodes to identify undetected features in an otherwise fixed graph [22]. This area of research has broadly become known as automated machine learning (AutoML) and pitches itself as a potential solution in overcoming the many barriers to entry in applying AI technologies (e.g., monetary cost, technical expertise, etc.) [23].

The idea being pursued is to allow neurons processing the input signal to "divide" to manage currently uncovered regions in the input space of the target distribution being modeled. In this way, the input

domain of a given network is not a fixed structure in the neural architecture being trained. Importantly, this dynamic input domain has the effect of removing the arbitrary domain of $(-\infty, \infty)$ that generally characterizes deep neural networks. Rather, the domain is indefinite in shape and grows only as large as is logically necessary to manage the input signal.

As an abstract mechanism, this operation of neural cell division is all well and fine, but how should this strategy be implemented? And, what is the criterion being applied to evaluate whether the pursued strategy is reasonable or efficacious? To answer these questions, it is helpful to adopt some tools from coding theory. Of particular interest to this strategy, is the concept of radix economies.

The radix of a system numeric simply refers to its base (e.g., binary systems are radix 2, hexadecimal systems are radix 16, etc.) [24]. Radix economies are concerned with the development of the most economical codings or representations of a given problem. Historically, this area of coding theory has suggested that base-3, or ternary, systems are more economical than binary systems. The simple reason that base-3 systems are considered (near) optimal is that Euler's number is generally considered the true optimal radix and 3 is simply the closest rounded integer value [25]. But, due to the inability to easily represent three logical states early on in the area of computation, binary implementations have since won-out [26].

These concepts are conducive to managing the problem being faced: Given an input stimulus of unknown magnitude, what is the most economical means of representing it? Coding theory's answer is the optimal value for Equation 2.1.3a below,

$$time(b, N) = \lfloor log_b(N) + 1 \rfloor, \ N \geq 1 \tag{2.1.1}$$

$$space(b) = b \tag{2.1.2}$$

$$E(b, N) = time(b, N) \cdot space(b) \tag{2.1.3a}$$

where N represents the signal magnitude being represented and b is the base of the system. As indicated, it is helpful to think of Equation 2.1.3a as being an optimization of two components: 1) time and 2) space. In this way, the economy is a measure of time-space complexity.

This time and space breakdown is helpful because it provides an analog to written digits (this analogy does extend to other physical systems as well, but the act of written digits is accessible). Given some value N and a base b, it would require the hypothetical transcriber capable of writing 1 symbol per unit of time, $time(b, N)$ units of time to complete the task. In this way, it is helpful to think of each symbol as a decision process that is able to reduce the uncertainty in N. Hence, the repeated application of such procedures reduces the uncertainty exponentially. Accordingly, $space(b)$ or just b, is the property that defines the amount of uncertainty reduced by such hypothetical processes.

In this setup, the optimal radix economy would then be the radix that best covers the domain of interest. Hence, it is the radix value, Ω, with the best average (time-space) economy over the value-domain being represented (e.g., $\mathbb{R}$, $\mathbb{N}$, etc.).

Explicitly:

$$\Omega = \underset{b \in \mathbb{R}}{\arg\min} \sum_{N=1}^{\infty} E(b, N) \tag{2.1.4}$$

The subsequent goal is to find Ω. Luckily, this value has been demonstrated by Stanley Hurst in his analysis of multi-valued logics [27]. The proof is as follows, but it uses a variation of the radix economy formula.

$$E(b, N) = b\ log_b(N),\ N \geq 1 \tag{2.1.3b}$$

For large N, this variant is approximately equal to Equation 2.1.3a. The proof for the optimality of $\Omega = e \approx 2.718...$, under Equation 2.1.3b) is as follows [27]:

$$E(b, N) = b\, log_b(N),\ b > 0 \quad \text{(Step 1)}$$

$$= b\frac{ln(N)}{ln(b)} = ln(N)\frac{b}{ln(b)} \quad \text{(Step 2)}$$

$$\frac{d}{db}E(b, N) = \frac{d}{db}ln(N)\frac{b}{ln(b)} \quad \text{(Step 3)}$$

$$E'(b, N) = ln(N)(\frac{1}{ln(b)} - (\frac{b}{(ln(b))^2}\frac{1}{b})) \quad \text{(Step 4)}$$

$$= ln(N)(\frac{ln(b) - 1}{(ln(b))^2}) \quad \text{(Step 5)}$$

$$= ln(N)\frac{1 - 1}{(1)^2} = 0, \text{when } b = e \quad \text{(Step 6)}$$

$$E'(b, N) < 0, \text{when } b < e \quad \text{(Step 7)}$$

$$E'(b, N) > 0, \text{when } b > e$$

$$b = e \text{ is the global minimum of } E(b, N) \quad \text{(Step 8)}$$

Since, $b = e$ is the global minimum of $E(b, N)$, it is clear that $\Omega = e$. The proof above is convincing, but is the use of the variant radix economy formula, Equation 2.1.3b, as unassuming as it seems? In this variant, the $space$ representation stays the same, but the $time$ component does change considerably. This shift allows for quanta of time that are decimal in value. This change in representation may not be problematic in all systems, but if time is understood as the number of discrete (and possibly parallelized) processes needed to produce the representation then there may be trouble on the horizon. For example,

this suggestion of a fractional $time$ maps to a set of numerical systems that may include fractional places (e.g., What is meant by a fractional tens, hundreds, etc. place?). What this criticism is meant to portend is that a fractional time element implies the ability to allocate resources in a way that exactly matches the resource requirements of N and, hence, the result of the above proof is not strictly independent of N as it first seemed. An analogy of this criticism would be the ability to always provision virtual machines in a cloud environment which have the exact resource requirements for the processes being managed (i.e. The only way to know the exact resource requirements for an arbitrary input N, would be to have already known N a priori).

Therefore, it can safely be said that - for at least some systems - Equation 2.1.3a is the proper characterization of economy and not Equation 2.1.3b when having to manage arbitrary inputs N. The following question presents itself: Is the Ω characterized by Equation Equation 2.1.3a (let's call it Ω_a) the same as the Ω just previously found (let's call it Ω_b)?

This question is not as easily resolved through analytic means (due to the discontinuities caused by the floor function) and requires a more empirically founded analysis:

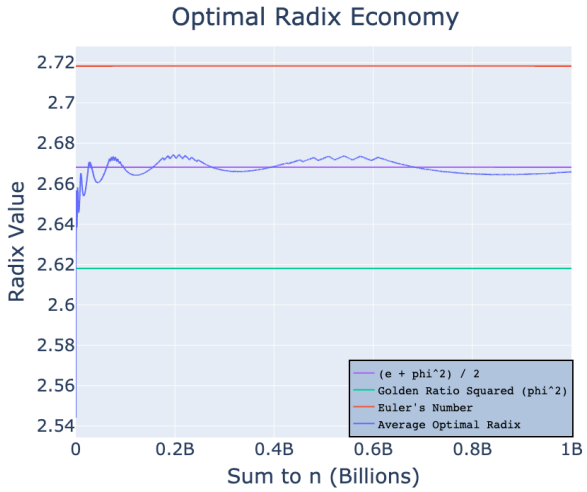

Figure 2.1.1: The above graph illustrates the empirically identified optimal radix value, Ω, (blue line) given the associated limit n. The top and bottom horizontal lines (Euler's number (orange) and ϕ^2 (green)) represent bases that are equidistant from the asymptotically defined optimal radix (purple line).

The above figure asymptotically identifies Ω_a a the midpoint between $b = e$ and $b = \phi^2$, such that:

$$\Omega_a = \frac{e + \phi^2}{2} \tag{2.1.5}$$

This result is clearly unexpected, but the belief that will be pursued is that Ω_a is an artifact of a more fundamental tension between the management of discrete and continuous input domains. Fundamental to this discussion (as visible in phenomena like the Fibonacci series) is the observation that the Golden Ratio is a point of symmetry between additive and multiplicative processes:

$$\phi^0 + \phi^1 = \phi \cdot \phi = \phi^2 \tag{2.1.6}$$

$$\phi^2 + \phi^3 = \phi^2 \cdot \phi^2 = \phi^4 = (\phi^2 \cdot Eqn.2.1.6) \tag{2.1.7}$$

The importance of this symmetry is that it allows for an additive procedure for producing multiplicative changes, which in turn allows for exponential scaling (this point will be addressed further below in section 2, which covers implementation).

Now, leveraging some of the prior work from chapter 1 section 3, it has been shown that operators of the dynamical system defined to adjust their incoming edge weights with Equation 1.3.5 (Oja's Golden Rule) will in the long-run, provided graph convergence, lead to a system that has drifted to a position where aggregate edge behavior, y, can be characterized as one of the two identified attractors (i.e., $y \in \{0, 1\}$). The result of this convergence, however, meets exactly the tacit assumption at work in Equation 2.1.3b, i.e. the trained graph leads to an encoding where the possible inputs N provided from the input distribution are memoized relative to the capacity of the processing unit, the neuron. In this way, if it is assumed that the implemented learning scheme's neurons are radix ϕ^2, then a trained and converged graph will lead to

an encoding that meets the conditions of Equation 2.1.3b and, hence, could be re-encoded with $b = e$. The aggregate economy of the system, however, will be proportional to Ω_a and the optimal economy achieved. That is, for the optimal economy to be achieved, it is necessary for two distinct encodings, e_1 and e_2, to be produced with bases $b_1 = \phi^2$ and $b_2 = e$, respectively. The average economy of such behavior, then, is Ω_a (the descriptive value of the tension between discrete and continuous representations). With this setup in mind, the next section addresses implementation of this encoding.

§ 2.2 Division, Relativization, and other Implementation Questions

This section briefly discusses some of the nuances encountered in implementing this algorithm. Of primary importance to the foregoing discussion is the implementation of division. Figure 2.2.1 below indicates the growth strategy of any given neuron, j. Assume that neuron j has a capacity of ϕ^2, this assumption is arbitrary and j could have any non-zero externally defined capacity. Finally, for any aggregated input $y = o_j \notin [-\phi^{-1}, \phi]$, "divide" the neuron according to the following scheme:

- *Division Rule 1.* Create a node j' with incoming and outgoing edges to all of the same nodes as node j. Further, each of these edges should be weighted such that $w_{ij} = w_{ij'}$ for incoming edges and $w_{jk} = w_{j'k}$ for outgoing edges.
- *Division Rule 2.* For each incoming edge weights w_{ij}, scale it down by a factor of ϕ^{-2}, such that $w_{ij}(n+1) = \phi^{-2} w_{ij}(n)$. Further, for each incoming edge weights $w_{ij'}$, scale it down by a factor of ϕ^{-1}, such that $w_{ij'}(n+1) = \phi^{-1} w_{ij'}(n)$.
- *Division Rule 3.* Lastly, increase the outgoing edge weights for j', by scaling up by a factor of ϕ, such that $w_{j'k}(n+1) = \phi w_{j'k}(n)$.

These reweightings are likely confusing with respect to traditional network architectures and it brings up a point that has only indirectly been addressed thus far. The point being, each of the neurons develops its own relative (intrinsic) notion of value. That is, each neuron defines its own identity value (i.e., the value

1), which determines how it behaves with respect to input stimuli. Explicitly, for a node j, the value for unity is a relatively defined object dependent on its outgoing edge weights:

$$(internal)\ 1_j = \sum_{k=1}^{m} |w_{jk}|\ (external) \tag{2.2.1}$$

Accordingly, the input to any neuron, $y = o_j$, is a quantity relative to 1_j. Hence,

$$y_j = \frac{\sum_{i=1}^{n} w_{ij}}{\sum_{k=1}^{m} |w_{jk}|} \tag{2.2.2}$$

This relativization does not change any of the conclusions made thus far, but it does change how Oja's Golden Rule, Equation 1.3.5, should be looked at. This reinterpretation means that Equation 1.3.5 operates across scale and is not an explicitly fixed construct across distinct neurons i, j, and k.

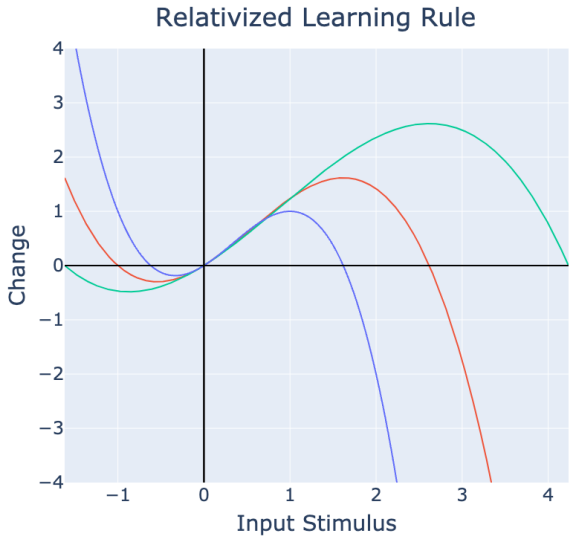

Figure 2.2.1: Each of the lines represent possible instantiations of Equation 1.3.5 under the relativizing conditions Equation 2.2.1 and 2.2.2.

With this additional context provided, the scaling suggested in the defined division scheme is more tractable. Scaling the input weights for j and j' by ϕ^{-2} and ϕ^{-1}, respectively, (*Division Rule 2*) causes the

system to maintain the magnitude of the input signal. This consequence is visible through an application of Equation 2.1.6: $\phi^{-2} + \phi^{-1} = \phi^0 = 1 = (\phi^{-2} \cdot Eqn.2.1.6)$.

Furthermore, the scaling up of j' by a factor of ϕ (*Division Rule 3*), leads to the expected increase in graph capacity. Again, by application of Equation 2.1.6, $1_j + \phi_j = \phi_j^2$. Hence, the capacity of the system moves from 1_j at time n to ϕ_j^2 at time $n + 1$. This growth is indicated in the figure below.

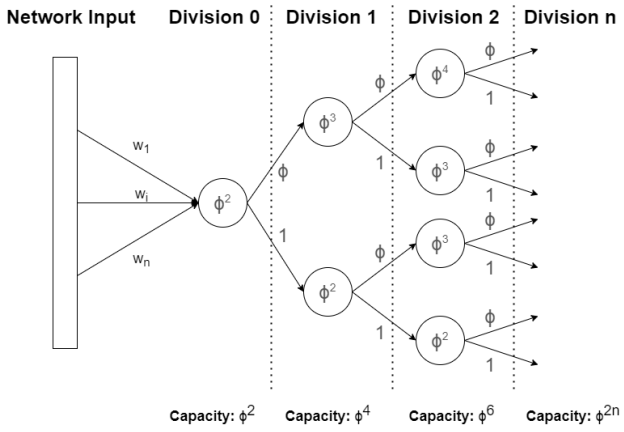

Figure 2.2.2: The division strategy of the graph implementing Oja's Golden Rule with an optimal radix encoding. The maximal growth rate of the graph is ϕ^2 as is evident by the exponentially growing capacity indicated above.

An additional constraint is added to improve the encoder's stability. Namely, a hyperparameter, π, is added to denote the division refractory period. This refractory period indicates the number of inputs that must occur between cell divisions. After division occurs and the cells enter the refractory period, they cannot divide again until π inputs (time-cycles) have passed, but they are still able to propagate input signals during this time.

With the outline of division provided, there remain a few housekeeping items necessary to wrap-up implementation as a topic. The first of these points covers signal propagation. The basins of attraction are visible in Equation 1.3.7, but it is reasonable to prevent signal propagation unless a certain activation threshold is met. Since the attractors for any given neuron are 0 and 1_j, it makes sense to base this activation

threshold to the active attractor 1_j. Let there exist some bias, β_j, (defined as a hyperparameter), such that the neuron is activated only if $y_j \geq 1_j - \beta_j$, otherwise it does not contribute to the activations of nodes in the subsequent layer.

With node activations addressed, it is clear that the graph has a preference for persisting positive signals and dampening negative ones. To address this bias, the idea of node *polarity* is introduced. If a node is positively polarized, then it operates in the manner so far discussed. If a node is negatively polarized, then it simply means that they obey a learning rule similar to Equation 1.3.5, but reflected across the line $y = -x$ and have an inverted activation scheme: $y_j \leq -1_j + \beta_j$.

As a final item, since the experimental goal being pursued here is binary classification (see below), the output layer consists of a single node using the standard logistic activation function. Like the other nodes, the output node depends upon y_j, but since it does not have any outgoing edges it is uncertain how 1_j should be defined. It is clear, however, that this value will have to be externally defined. For simplicity, this value is fixed such that $1_j = C$, where C is an arbitrarily chosen integer of sufficient magnitude (to approximate the Golden Ratio). The edge weights coming into the output node are trained using backpropagation. Since the backpropagation algorithm is only being applied at the output node, the concerns of global ("exponential") context data raised in chapter 1 section 2 are not problematic.

CHAPTER 3

EXPERIMENT

§ 3.1 Technology

The technology this project is implemented is the Python programming language. The primary libraries used include Sci-Kit Learn (sklearn), Pandas, Matplotlib, and Plotly, and NumPy. Sci-Kit Learn contains APIs for generating example data. Plotly provides objects for clean data visualization.

Due to the dynamic nature of the neural network topologies considered, Tensorflow and PyTorch were not considered as viable packages for graph definition. Accordingly, all graph data structures are implemented manually. If interested, then please contact me to get access to the code used in this paper; this involved developing a custom network architecture which was not a trivial task in itself.

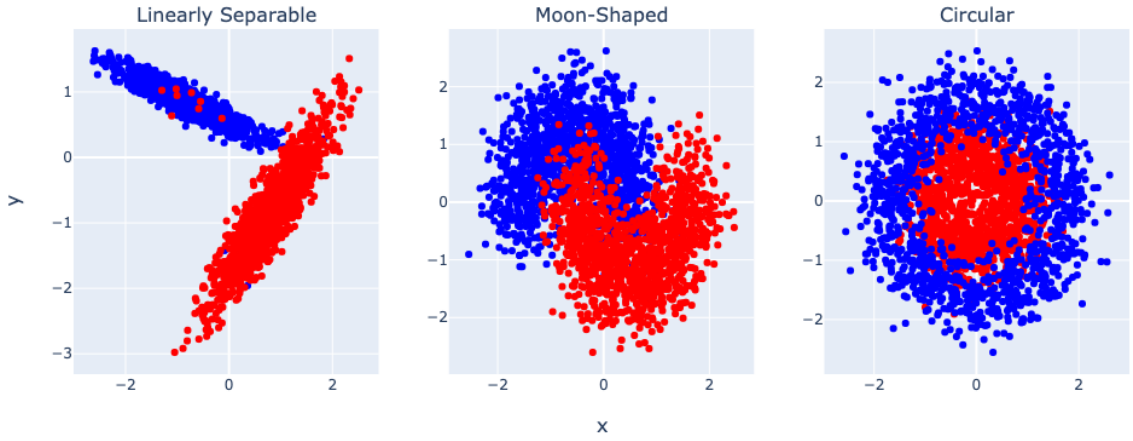

Figure 3.2.1: The binary class datasets being evaluated. They are linearly separable (top), moons (middle), and circular (bottom).

§ 3.2 Data

The data set for this project is generated via Sci-kit learn's "datasets" library. The data consists of three binary classification data sets: 1) linearly separable clusters, 2) moon-shaped clusters, and 3) a circular cluster surrounded by a cloud of points. The linearly separable cluster has a small noise transform applied to it that amounts to a uniform perturbation between 0 and 0.1. The moon-shaped data set has a slightly larger amount of uniform noise added (between 0 and 0.3). The circular cluster surrounded by points of the opposite class has a uniform perturbation bounded by 0.2 and a scale factor of 0.5 between the inner and outer circles.

Each of the three data sets contains 2500 records defined by an x, y coordinate and the associate class label, $L \in \{0, 1\}$. The data is divided such that 80% of the data (2000 records) are used for training and 20% are used for testing (500 records). Furthermore, due to the fact that all nodes in the architecture being evaluated here are entirely binary, it is necessary to preprocess the x, y coordinates of each of the inputs

and convert it into its binary representation. After each coordinate value is cast into a 32-bit representation, corresponding to 32 binary input nodes per variable. This conversion is only required for the graphs being developed in this paper and the raw x, y coordinate values are used by the comparison classifiers described in the next section.

§ 3.3 Experiment

Since a lot of the work discussed so far is very theoretical in nature and untested in any formal capacity, the experiment being suggested here is a very simple one focused on assessing the viability of this methodology and identifying particularly troubling bottlenecks in performance. As such, the goal of this experiment is to develop several simple ensemble learners: A voting ensemble and four stacked ensembles. Each of the base-learners in these ensembles is a network trained using the newly smithed Oja's Golden Rule learning algorithm. The performance of these ensembles will then be compared to a broad selection of classifiers, including: K-Nearest Neighbors, Linear SVC, Radial Basis Function SVC, Gaussian Process Classifier, Random Forests, Multi-layer Perceptron, AdaBoost Classifier, Naive Bayes Classifier, and Logistic Regression. A full analysis of each of these classifiers is out of the scope of this paper, but a listing of the configurations used are provided in the appendix.

The experimental setup, then, is fairly simple. For each of the binary classification data sets (linearly separable, moon, and circle), each of the above comparison classifiers is trained on the 2000 designated training samples and evaluated on the 500 remaining test records. The experimental setup for the ensemble classifiers is a little more involved.

§ 3.3.1 Voting Classifier

For the voting ensemble, 25 encoders are generated for each of the data sets. Each of these encoders uses the entirety of the training set. These encoders are graphs that contain an input layer, a single and dynamic hidden layer, and an output layer consisting of a single, fixed node with a logistic activation function. Using a decision threshold of 0.5, the testing data is evaluated by each of the voting members. The majority class is the predicted class of the ensemble learner.

§ 3.3.2 Stacked, Re-encoded Classifier

Since the motivation of this architecture is to develop a model that efficiently encodes the source signal, the stacked, re-encoded classifier attempts to utilize the dense knowledge representation in the hidden-layer suggested in the foregoing discussion. In this way, this classifier is similar to autoencoders in that the final layers are peeled away and the final encoding layer is used as a compact representation of the input data.

Accordingly, since the dense encoding is going to comprise the feature set for the meta-learner (which is just a logistic regressor), a 10-fold cross-validation type of strategy is used. However, since every single training instance for the meta-learner needs the encoding provided by each of the base-learners, it is not possible to perform training in isolation as is usual for stacked learning schemes. That is, it is not possible to train the learner on all of the in-fold samples and produce the out-of-fold samples as the "unbiased" input for the meta-learner. So, what is done as an alternative strategy is 10-fold cross-validation such that every fold is encoded by the trained encoder, but the fold left out of training does provide some "unbiased" input to reach the meta-learner.

Once all of the 10 base-learners are trained, all of the training samples are passed through them and the activations they cause in the hidden layer are recorded. These activations are the re-encoding provided by

the base-learning graphs and the features used by the meta-learner. Before training the meta-learner, however, the new binary feature set is ranked using the composite ranking formed by averaging the rankings produced via the features χ^2 and mutual information statistics. This ranking serves as a feature selection mechanism to reduce possibly redundant encodings produced in the ensemble. The top 32, 16, 8, and 4 feature sets are used as inputs to a logistic regressor for final classification.

§ 3.4 Results

The performance of the comparison and ensemble classifiers is provided in the below table:

Table 3.4.1: The summary of the ensemble and comparison classifier performance on the three data sets being evaluated.

Algorithm	**Linear**	**Moon**	**Circle**	**Average**
Gaussian Process Classifier	0.9860	0.9300	0.9000	0.9387
Radial Basis Function SVC	0.9860	0.9300	0.8980	0.9380
AdaBoost Classifier	0.9800	0.9220	0.8940	0.9320
Multi-layer Perceptron	0.9800	0.9180	0.8920	0.9300
Random Forests	0.9820	0.9160	0.8840	0.9273
K-Nearest Neighbors	0.9860	0.9120	0.8600	0.9193
Naive Bayes Classifier	0.9720	0.8720	0.8940	0.9127
Stacked Ensemble (Top 32 Feat. - 32 bit)	0.9740	0.9120	0.8060	0.8973
Stacked Ensemble (Top 16 Feat. - 16 bit)	0.9720	0.8760	0.7600	0.8693
Stacked Ensemble (Top 8 Feat. - 8bit)	0.9580	0.8740	0.7640	0.8653
Stacked Ensemble (Top 4 Feat. - 4 bit)	0.8760	0.8300	0.7640	0.8233
Voting Ensemble	0.9340	0.8280	0.6500	0.8040
Linear SVC	0.9760	0.8740	0.5340	0.7947
Logistic Regression	0.9800	0.8780	0.4660	0.7747

Of the ensembles, the re-encoded, stacked classifier using 32 features (32 bits) performed the best, achieving an average accuracy of 89.73% across all data sets. This, unsurprisingly, outperformed linear methods such as Linear SVC and Linear Regression, while only narrowly less performant than the other comparison classifiers (0.5% - 4% difference in average performance). The most striking divergence in performance clearly occurs in the data set of circularly arranged classes. It is clear that this methodology is unable to manage non-linearities in a robust capacity. The 32-bit stacked ensemble achieved 81 accuracy and the worst (voting) 65%. This issue is likely resolvable if network depth were to be increased.

Performance on the data set containing moon-shaped class distributions is similar to the top-performing classifier (91.20% vs. 93.00%). Further, it is impressive that the 4-bit ensemble yielded an 82.33% average accuracy. This point is made, because dense encodings of the input distributions are the motivation of the discussion on radix economies.

A visualization is provided below to show the classifications made by the ensembles discussed here.

§ 3.5 Discussion

As is clear, the weak-learners developed in this paper can successfully be used to produce reasonable prediction functions. The overall performance is not that impressive and further work needs to be employed to understand the scalability of this scheme; but, as the debut of this methodology, the observed performance does indicate the promise of future improvements. As was stated, the main goal at the outset of this paper is not to develop the best learning scheme possible, but to develop a learning scheme that can reasonably piggy-back off of an economically self-organizing system.

This analysis was incredibly helpful in elucidating the current limitations of this architecture and providing a foundation of what adjustments to implementation need to be made. Some possibilities for future work can be enumerated as follows:

- **Increased Network Depth**. The ability to create deep networks was programmed, but they exhibited some oscillatory behavior and instability. These negative effects would lead to failures in convergence and continued layer growth. However, it is likely the only means by which non-linear distributions will be successfully modeled.
- **Over Reliance on Initial Conditions**. Due to the fact the hidden layer grows, the initial weights of the graph heavily determine the later success of the encoder. This determinism can cause the encoder to become "stuck" in predicting a single output value.
- **Convergence Conditions and the need for an Annealing Mechanism**. The network, even after performing well on many batches consecutively, can be guided into performing poorly after encountering some bad inputs. It appears that this behavior is due to the on-line nature of the graph and the lack of parameters for adjusting learning rate dynamically. This leads to performance metrics (accuracy, etc.) oscillating during training. It would be helpful to provide some stabilizing mechanisms to later iterations.
- **Coverage**. The notion of "coverage" is not heavily considered in the discussion above, but the graphs developed here are not considered total functions. That is, they may not yield any result for a provided input. This behavior is intentional and is hoped to help produce "specialization" in ensemble methods, where some input is deliberately ignored by models to optimize performance in subsections of the input space. This topic needs to be fleshed out in more detail.

However, in light of these potential new avenues of study, this architecture is not without its drawbacks:

- **No External Libraries**. There is no available external framework or code library implementing this learning scheme and, hence, it all needs to be coded manually.
- **Narrow Domain/Little Related Work**. Most of the guiding work for this project belongs to coding theory, which is not necessarily immediately translatable into machine learning friendly concepts. Further, with deep learning entrenched in its fairly rigid form, there is little popular work accessible. For example, cascade correlation is pretty much lost to the ages.
- **Input Binarization**. The need to convert real-valued inputs into their binary forms leads to a dimensional explosion that is going to be problematic for most contemporary, or "big-data centric", problems.
- **Hyperparameter Introduction**. As indicated in Section 2, one of the peripheral intents of this project was to reduce the number of hyperparameters that needed to be externally defined by the user. The hope was that this would reduce technical knowledge in implementing dynamic schemes such as this one (AutoML), while also leading to a more intrinsically self-reliant structure. There was mild-success in this initiative, but it also introduced new hyperparameters (e.g., division refractory period, activation bias, etc.).

In spite of these problems and the wanting results, this project was an invigorating and stimulating undertaking aimed at blending an efficient representation of signal data with a learning scheme.

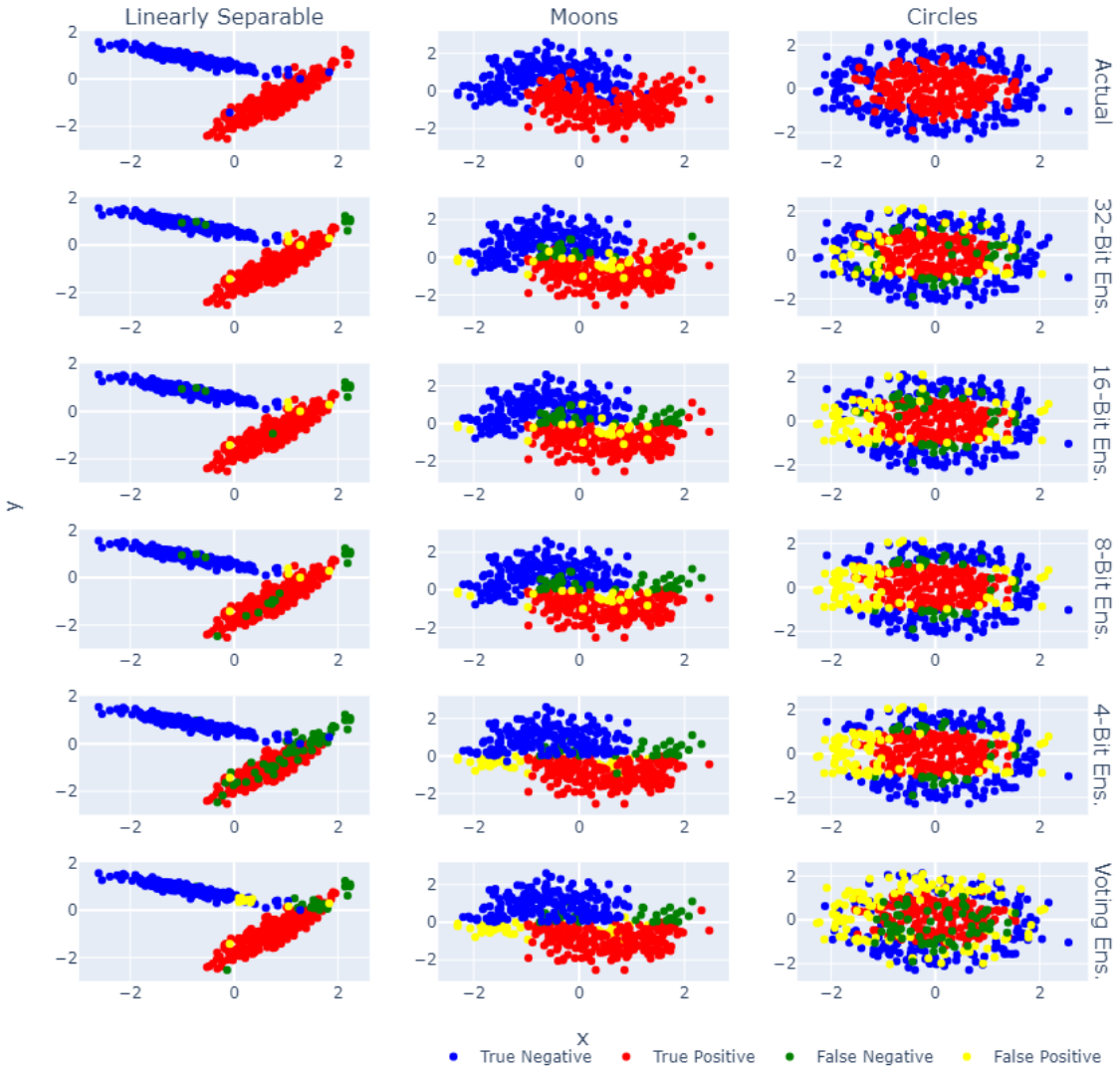

Figure 3.4.2: Visualization of the classification made by each of the ensemble methods. The top row is the ground truth being predicted. The coloring indicates the classification type: True Negative (Blue), True Positive (Red), False Negative (Green), and False Positive (Yellow).

CHAPTER 4

PREDICTION MARKET

§ 4.1 Introduction and Conceptual Distinctions

As alluded to earlier, the learners being trained become unstable as the training procedure continues. Instability, in this context, refers to a tendency of the models to either produce output values or develop edge weights of an increasingly large and unbounded magnitude. The hypothesized reason for this undesirable property is that a scaled quantity (e.g. the integer value output by a given model) is being employed to estimate an unscaled value (e.g. the probability that a given training instance belongs to a certain target class). Accordingly, since the logistic activation function only asymptotically approaches the probabilities of 0 or 1, even a fairly well-performing model will always be corrected to produce larger and larger (integer) activations to better approximate the labeled training examples. The intent of this section will be to develop (but not implement) a possible solution to this problem.

In a typical machine learning context, labeling with these "extreme" probability values is not problematic, but it proves to be of consequence in this setting. One potential solution to reduce the influence of this asymptotic pull is to provide an assignment of labels not drawn strictly from the set of "extreme" probability values. The general idea of this solution being that intermediate probability values are able to be produced by the models being developed and, hence, the scale creep found by exclusively pursuing asymptotic values is attenuated. Of course, the immediate question then becomes how to employ such a non-extreme labeling scheme when provided only the true target classes.

An approach for easing the asymptotic pressure that is being placed on the learning process may be made available by making a slight adjustment to how the "0" or "1" labels should be interpreted and, hence, used in training. If "0" or "1" are not taken to function as extreme "attractors" of the models being developed, but are simply taken to denote a domain of possible probability values, then a significant loosening of the constraints imposed on a model has occurred. For example, with this reframing in place, it is possible to interpret a training instance x with label "0" as belonging to the domain $D_0 = [0, 0.5)$, while any instance y with label "1" belongs to the domain $D_1 = (0.5, 1]$. This interpretation is unsurprising and what is generally denoted when given a classifier decision boundary of 0.5. The next question becomes how such a (re)assignment can be made.

A naive strategy for accomplishing this task is to simply identify all instances x_i with label "0" with a random variable $r_i \in [0, 0.5)$ and all instances x_j with label "1" with a random variable $r_j \in (0.5, 1]$.While being able to provide classification pressure (i.e. instances with label "1" are pushed to have an average labeling/probability assignment greater than instances labeled with "0"). However, this approach has the clear disadvantage of being unable to assign similar probabilities to instances, x_i and x_k that may be close in the input space since there is no relation between r_i and r_k. To remedy this issue, it is apt to turn to some theoretical work completed by the Machine Intelligence Research Institute of UCLA.

The general idea being introduced by this prior work is the development of a hypothetical community of "mathematicians," where each mathematician is tasked with assigning the likelihood that a given statement ϕ belonging to a formal language $\mathcal{L}$ is true (e.g., the probability that the statement "$1 + 1 = 2$" is true in Peano arithmetic). In this way, the group of "mathematicians" functions to provide a valuation for every possible statement in the language $\mathcal{L}$. This valuation sets up the potential for a "marketplace" where shares for the various statements of $\mathcal{L}$ can be bought and sold at the prevailing market price (i.e. the prevailing belief that a given statement is true/false). In tandem with this valuation process, there exists a

deductive procedure that operates according to the rules, consequence relations, and prior facts discovered by the market about $\mathcal{L}$. This deductive procedure proves the truth (1) or falsity (0) of a finite number of statements currently being traded on the market during each trading period T_i. For those traders that have netted shares for a given statement, the deductive procedure presents itself as a "probability collapse" to 0 or 1, where the market's error is the difference between the prevailing market prices (prior to collapse), at which shares were bought and sold, and the collapsed price (0 or 1).

This event can be thought of as the termination of a futures contract. These sorts of contracts are widely invested in by large organizations in an attempt to reduce future financial uncertainty. For example, an airline may purchase F amount of jet fuel at an agreed upon price one year from the current date; this behavior allows for entities to hedge their risks. If the market price of jet fuel is dramatically higher a year from contract start, then the airline successfully mitigated a cost. If the market price of jet fuel is lower, then they incurred an undue cost. Good and poor investments are expected, but if an entity is able to perform well enough (i.e. averaging the market performance), then it is able to assess its future resource constraints reliably and has successfully reduced its uncertainty. The goal then is to cast a typical machine learning problem into a prediction market.

§ 4.2 Formulation

Garrabrant et al. define $\mathcal{L}$ as the language of propositional logic and S to be the set of all well-formed sentences in $\mathcal{L}$ (e.g. $(\phi \wedge \psi) \rightarrow \beta$) (Garrabrant et al.). Yet, since the current project is not concerned with any particular formalism, $\mathcal{L}$ and S are defined more loosely.

Definition 4.2.1: Let $\mathcal{D}$ be any bounded distribution mapping from the d-dimensional real-space into the binary space $\{0, 1\}$, denoted $\mathbb{B}$, i.e. $\mathbb{R}^d \rightarrow \mathbb{B}$. The "bounded" condition is meant to indicate that there exists a bounding condition $U : \mathbb{Q}^d$ such that for all instances $x :< x_1, ..., x_d >$ drawn from the provided distribution $\mathcal{D}$, $|x_i| \leq U_i$. Further, let all $U_i \geq 1$. The purpose of this condition is to guarantee a uniform level of precision in description across all instances x being considered. Note that this uniform level of description is a significant break from the case presented by Garrabrant et al., which allows for any logical statement (as part of the inductive definition of well-formed formulae), regardless of its rank, to be considered. The purpose of this note is to highlight a difference between the types of description being employed. Logical description permits statements of any rank to be evaluated while the distributive description being used in this application uses a fixed descriptive "rank" or precision. However, what contributes to the fixed description may be of any order, but this aspect is abstracted out by the descriptive limitations (e.g. all the sort of noise that influence a distribution, but are not explicitly recognized in the description of problem instances).

Definition 4.2.2: Let b be the selected base/radix by which the distribution $\mathcal{D}$ is going to be encoded.

Definition 4.2.3: Let l be the finite, maximum precision (the number of "bits") by which the input values of the distribution $\mathcal{D}$ should be considered. This precision will exclude, for example, irrational numbers from being represented with complete fidelity, but it will allow for an arbitrary precision.

For the purposes of this application,

$l = \lfloor log_b(\arg\max(U)) + 1 \rfloor + p$, where p is simply a non-negative, integer precision constant. One can think of p as the constant that allows for arbitrarily selected decimal precision (since the log component only guarantees integer precision).

Definition 4.2.4: Let $\mathcal{D}_l$ be the encoding of the provided distribution $\mathcal{D}$ with radix b and max precision of l. This encoding is effectively a rationalization of $\mathcal{D}$ (i.e. $\mathbb{R} \rightarrow \mathbb{Q}$), since all encodings are relative to a power of b.

Provided that $b = 2$, $\mathcal{D}_l$ is a mapping: $\mathbb{B}^{lXd} \rightarrow \mathbb{B}$. Note that this mapping is a formalization of the binarization considered previously when applying Oja's Golden Rule above.

Definition 4.2.5: Let S be the set of l-precise statements drawn from the encoded distribution $\mathcal{D}_l$. Let ϕ denote a single l-precise statement from S.

Definition 4.2.6 (Valuation): Let a **valuation** be any function $\mathbb{V} : S \rightarrow [0, 1]$. We refer to $\mathbb{V}(\phi)$ as the value of ϕ according to $\mathbb{V}$. A valuation is called rational if its image is in $\mathbb{Q}$ (Garrabrant et al.).

Definition 4.2.7 (Pricing): Let a **pricing** be any function $\mathbb{P} : S \rightarrow \mathbb{Q} \cap [0, 1]$. Garrabrant et al. refer to pricings as computable valuations, because of the inherent limitation in precision in computing systems (e.g. 32-bit or 64-bit representation of π). In this way, computability is equated with rational descriptions.

Definition 4.2.8 (Market): A **market** $\overline{\mathbb{P}} = (\mathbb{P}_1, \mathbb{P}_2, ...)$ is a computable sequence of pricings $\mathbb{P}_i : S \rightarrow \mathbb{Q} \cap [0, 1]$ (Garrabrant et al.).

Definition 4.2.9 (Belief State): Let a **belief state** be any computable rational valuation $\mathbb{P} : S \rightarrow \mathbb{Q} \cap [0, 1]$ with finite support, where $\mathbb{P}(\phi)$ is interpreted as the probability of ϕ (where $\mathbb{P}(\phi) = 0$ for all but finitely many ϕ) (Garrabrant et al.).

Definition 4.2.10 (Computable Belief Sequence): A **computable belief sequence** $\overline{\mathbb{P}} = (\mathbb{P}_1, \mathbb{P}_2, ...)$ is a computable sequence of belief states, interpreted as a reasoner's explicit beliefs about the instances ϕ drawn from the distribution $\mathcal{D}$ as they are refined over time.

For a given computable belief sequence, the subscripts in the belief states denote the time at which the belief state was held by a given reasoner. Hence, all belief states in the considered belief sequence were those held by a single reasoner. As will be expressed later, one may think of $\mathbb{P}_i$ as a vector of output probabilities produced by a single model for each ϕ in S (or other batch S'). In this way, the weights and output of this continually trained model (between times $1, 2, ...$) represent a hashing of the prior belief states. That is, a trained model can be thought of as a block-chained hashing of the reasoner's prior belief states $(\mathbb{P}_1, \mathbb{P}_2, ...)$.

Definition 4.2.11 (Maturation Process): A **maturation process** $\overline{M} : \mathbb{N}^+ \to Fin(S)$ is a computable nested sequence $M_1 \subseteq M_2 \subseteq M_3 \ldots$ of finite sets of instances from $\mathcal{D}$. Here, the domain $\mathbb{N}^+$ indicates the point in time at which the maturation process is operating and $Fin(S)$ is meant to denote the space of all finite subsets of instances drawn from $\mathcal{D}$. For example, at time 0 the number of "matured" instances is the empty set, $M_0 = \{\}$. The subsequent period may be characterized as $M_0 = \{\phi_{10}, \psi_5\}$, where ϕ_{10} and ψ_5 represent arbitrarily selected "matured" instances. Going with the "futures" analogy suggested above, one can think of the matured instances as contracts that have reached their end date. Though not discussed here, the maturation process is the distributional analog to the "Deductive Process" employed by Garrabrant et al. to manage the "probability collapse" mapping predicted probabilities to logical certainties (i.e. $C : [0, 1] \to \{0, 1\}$). In the implementation that will be characterized, however, the "collapse" function is of a slightly different nature, $C : [0, 1] \to [0, 1]$. $M_\infty = \cup_n M_n$ and represents

the terminal maturation of the distribution (i.e. all of the instances of the distribution, or sampling, have matured) (Garrabrant et al.).

Definitions 4.2.6 to 4.2.11 borrow heavily from Garrabrant et al.'s definitions (some being nearly verbatim).

The above definitions are enough for the work rendered in this paper, but an extension is still in development and mentioned here as future work to be completed. The reasoners and markets considered so far are not as refined as they should be to define a proper market where shares can be bought or sold. Rather, the market posed so far is an egalitarian ensemble where reasoner belief states reach an equilibrium of aggregate confidence with respect to any given ϕ in S. The intended benefit of this ensemble is to help attenuate trader growth. Notwithstanding this intended function, a remaining issue is that there is no explicit quantification of shares being bought/sold or profits made/lost with respect to instances and their later maturation, given $\overline{M}$. Accordingly, there is no interface for "budgeting" resources and, hence, there is no selective pressure limiting reasoner participation in the market space (i.e. A firm on Wall Street is only likely to be a significant player and be awarded a lot of investment capital if it performs well on average). This pressure is on par with the selective pressure used to foster candidate solutions in evolutionary algorithms. A more complete strategy would attempt to harness this selective pressure to eliminate bad reasoners with the intent of improving the market's overall predictive quality.

To better allow for such mechanisms to be installed, additional definitions are needed. These definitions are largely on par with the following constructs from Garrabrant et al.'s work: Valuation Feature, Price Feature, Expressible Feature, Trading Strategy, Trader, Firm, and Budget. The Valuation Feature, Price Feature, and Expressible Feature constructs operate to convert reasoner belief into expressions denoting the number of shares a given trader will purchase/sell for some ϕ. The trading strategy of a reasoner represents the overall purchase/sell schedule that a given reasoner is going to execute on a given day. Fur-

ther, a trading strategy can be thought of as an affine combination $TS = c + \xi_1\phi_1 + ... + \xi_k\phi_k$, where ξ is the number of shares and ϕ represents the prevailing market price (Garrabrant et al.). A trader is a reasoner with a sequence of trading strategies for each day n. A firm is a collection of traders and a budget is a resource constraint limiting the volume of interactions (purchases/sales) that a firm/trader may make.

§ 4.3 (A Characterization of) Implementation

Given the toy classification problems considered above, we have distributions $\mathcal{D}_{linear}$, $\mathcal{D}_{moon}$, and $\mathcal{D}_{circle}$, which have successfully been converted into binary mappings of the form $\mathbb{B}^{lXd} \rightarrow \mathbb{B}$. For the time being, refer to any of these defined distributions as $\mathcal{D}_{test}$ and consider the co-domain, $\mathbb{B}$, not as a class assignment, but as the result/outcome of an offered futures contract with a fixed valuation of 0.5 for any given instance ϕ. If one is only ever able to do as well as chance against a market where failure is as dominant as success, then average performance would be centered around 0.5. The idea of this interpretation of class labelings being that the models are competing against (making a contract with) a reasoner of only change/average ability. The labeling of an instance as "1," then, indicates that the futures contract terminates with a market valuation greater than 0.5 (vice versa for instances labeled a "0"). Lastly, with respect to the data set, the distributions referred to by $\mathcal{D}_{test}$ are known to be bounded and l is set to 32 (i.e. the 32-bit representation).

Reasoners in this prediction market correspond to a truncated version of the models using Oja's Golden Rule. This truncation is associated with their use of the logistic equation for activation and back-propagation. As discussed previously, the logistic function only asymptotically approaches the extreme probability values and, hence, continually produces values of a larger magnitude to approximate them. To reduce this magnitude creep, the following changes are made to the learning scheme:

Definition 4.3.1 (Reasoner): Define a **reasoner**, T, as any function $S \to \mathbb{N}$ which is equivalent to $\mathbb{B}^l \to \mathbb{N}$, given that $\mathcal{D}$ has been encoded. The integer output, o, from a given model is referred to as its activation.

Definition 4.3.2 (Market Cap): Define the **market cap** as follows: $\kappa = \underset{T \in \Gamma,\, \phi \in S}{\arg\max}\, T_i(\phi_j)$, where Γ is the set of all reasoners trading on the market.

Definition 4.3.3 (Market Relative Activation): Define **market relative activation** to be a function $\rho : \mathbb{N} \to \mathbb{Q} \cap [0, 1]$. For the purpose of this implementation, equation 4.3.1 is used:

$$\rho(o, \kappa) = \frac{o}{\kappa} \tag{4.3.1}$$

Definition 4.3.4 (Probabilistic Market Relative Activation): Define **probabilistic market relative activation** to be a mapping $p : \mathbb{Q} \cap [0, 1] \to \mathbb{Q} \cap [0, 1]$. For the purposes of this implementation, equation 4.3.2 is used:

$$\alpha(x) = p(x) = \frac{1}{1 + e^{-tan(\frac{\pi}{2}x)}}, \text{ where } x = \rho(o, \kappa) \tag{4.3.2}$$

As a note, Equation 4.3.2 is considered as a function into the rationals because it is a computed value with finite precision, while the function itself belongs to the real space. Accordingly, the reasoner, along with equations 4.3.1 and 4.3.2, is a function from $S \to \mathbb{Q} \cap [0, 1]$. With this function provided, a pricing (Definition 4.2.7) for any given instance can be made (via a reasoner T). Also, the tangent function is used in the exponent as a means for the relative activations to exponentially approach the extreme probability values as they approach -1 or 1.

Definition 4.3.5 (Selection Process): Let $\overline{S'}$ be a maturation process called a **selection process**, such that $\overline{S'} : \mathbb{N}^+ \rightarrow Fin(S)$ is a computable nested sequence $S'_1 \subseteq S'_2 \subseteq S'_3 ...$ of finite sets of instances from $\mathcal{D}$. This process is called a "selection process," because it identifies a finite set of instances from $\mathcal{D}$ (or S) that are available for training, valuation, and/or trading. Informally, it means that at any time n there exist $|S'_n|$ records recognized in some capacity by the market. One can think of $|S'_n|$ as the size of the present market and its history, while all $\psi \notin S'_n$ are simply instances that have not been introduced to the market yet (e.g. shares for Starbucks or Apple were not capable of being traded before the companies existed).

Further, let μ be a positive integer, referred to in this context as the **selection rate**, where $S'_{k+1} = S'_k \cup \Delta_k$ and $|\Delta_k| = \mu$. For each $\phi_i \in |\Delta_k|$ it must be the case that $\phi_i \notin S'_k$ for some randomly and uniquely chosen $i \in \mathbb{N}^+$.

Definition 4.3.6 (Future Maturation Process): Let $\overline{F}$ be a maturation process called a **future maturation process**, such that $\overline{F} : \mathbb{N}^+ \rightarrow Fin(S)$ is a computable nested sequence $F_1 \subseteq F_2 \subseteq F_3 ...$ of finite sets of instances from $\mathcal{D}$. Further, let it be the case that $F_n \subseteq S'_{n-1}$ and $F_0 = \{\}$.

In the case being considered, this means that the future maturation process is one that selects instances from the target distribution that have already been selected via the concurrent selection process. The constraint being made explicit in this subset relation is that future contracts can only be made (and come to mature) with respect to publicly traded objects (i.e. those already selected to be on the market).

Further, let μ be a positive integer, referred to in this context as the **future maturation rate** (distinct from the selection rate above), such that $F_{k+1} = F_k \cup \Delta_k$ and $|\Delta_k| \leq \mu$. For each $\phi_i \in |\Delta_k|$ it must be the case that $\phi_i \notin F_k$, $\phi_i \in S'_k$, and $\phi_i \notin S'_{k+1}$ for some randomly and uniquely chosen $i \in \mathbb{N}^+$. As a

note to avoid any confusion, the Δ_k sets referred to in the selection and future maturation processes are distinct sets local to each of the processes.

Let $\Gamma = (T_1, T_2, ..., T_i, ...)$ be a set of reasoners available to us, S'_0 be initialized to a random sampling of instances, and F_0 be assigned the empty set. Accordingly, for all traders $T_i \in \Gamma$ and all $\phi_j \in S'_0$, it is possible to produce $|\Gamma|X|S'_0|$ pricings $\mathbb{P}_{ij} = T_i(\phi_j)$. With j fixed, define the market price for ϕ_j as follows:

If $\phi_j \notin F_n$, then $\mathbb{P}_j = \alpha(\Pi(\phi_j)) = \alpha(\frac{1}{\kappa|\Gamma|}\sum_{i=1}^{|\Gamma|} T_i(\phi_j))$,
If $\phi_j \in F_n$ and $\phi_j \notin F_k$ $(k < n)$ and $\mathcal{D}(\phi_j) =$ "1" and $\alpha(\Pi(\phi_j)) > 0.5$ then $\mathbb{P}_j = rand(\alpha(\Pi(\phi_j)), 1)$,
If $\phi_j \in F_n$ and $\phi_j \notin F_k$ $(k < n)$ and $\mathcal{D}(\phi_j) =$ "1" and $\alpha(\Pi(\phi_j)) \leq 0.5$ then $\mathbb{P}_j = rand(0.5, 1)$,
If $\phi_j \in F_n$ and $\phi_j \notin F_k$ $(k < n)$ and $\mathcal{D}(\phi_j) =$ "0" and $\alpha(\Pi(\phi_j)) \leq 0.5$ then $\mathbb{P}_j = rand(0, \alpha(\Pi(\phi_j)))$,
If $\phi_j \in F_n$ and $\phi_j \notin F_k$ $(k < n)$ and $\mathcal{D}(\phi_j) =$ "0" and $\alpha(\Pi(\phi_j)) > 0.5$ then $\mathbb{P}_j = rand(0, 0.5)$

where $rand(lower, upper)$ indicates that a random value x is drawn such that $x \in \mathbb{Q} \cap [0, 1]$ and $lower \leq x \leq upper$. Additionally, $\mathcal{D}(\psi)$ indicates the labeling provided by the target distribution for a given instance ψ. As a result, $\mathbb{P}_j$ will be fixed or bound to a value following its inclusion into the matured futures set F. Since, as previously discussed, it is a design choice to interpret the labels provided by $\mathcal{D}$ as the completion of a futures contract centered at 0.5, it becomes necessary to define the terminal pricing, $\mathbb{P}_j$, explicitly.

In the cases where the market agrees with the futures contract (i.e., $\mathcal{D}(\phi_j) =$ "1" and $\alpha(\Pi(\phi_j)) > 0.5$ or $\mathcal{D}(\phi_j) =$ "0" and $\alpha(\Pi(\phi_j)) \leq 0.5$), market confidence is favored. Here, "market confidence" is meant to indicate the market's proximity to one of the extreme probability values (0 or 1). In cases where the market is incorrect, corrective pressure is applied by fixing all future pricings of ϕ_j to a value that matches the distributional label. As a result of all of this work, a computable sequence $\overline{\mathbb{P}} = (\mathbb{P}_1, \mathbb{P}_2, ..., \mathbb{P}_j, ...)$ for

each $\phi_j \in \overline{S'_0}$ has been created and, by Definition 4.2.8, a market is instantiated. By successively iterating the selection and future maturation processes, a new market, $\mathbb{P}_n$, can be generated.

Now, the reader may have noticed that this process is still incomplete and will lead to a fairly unexciting pricing progression over time. That is, as defined, $\mathbb{P}_i^k = \mathbb{P}_i^{k+1}$ across all time steps. The reason for this market stagnancy is that nothing has yet been said about when model/reasoner training takes place. For brevity, a formal presentation of this step will be avoided, but training is taken to occur following the assignment of all $\mathbb{P}_j^n$ given a selected and matured set, $\overline{S'_n}$ and F_n. That is, during the evaluation of $\mathbb{P}_j^n$, all reasoners T_i are considered to be fixed (i.e., weights, etc.). Once the pricing assignment $\mathbb{P}_j^n$ is defined, however, a training period follows in which all the instances in $\overline{S'_n}$ are used (though evaluated randomly) for training. In this way, $\overline{S'_n}$ can be thought of as the instances of a training epoch at time n.

Finally, since the modified and market relative activation function (Equation 4.3.2) is a modified form of the logistic function, the resultant update rule is easily derived (where $\alpha(x)$ is the modified activation function, Equation 4.3.2).

$$\Delta w_{ij} = \frac{\pi}{2}(\mathbb{P}_j - \alpha(x))\alpha(x)(1 - \alpha(x))sec^2(\frac{\pi}{2}x) \tag{4.3.3}$$

§ 4.4 Conclusion and Preliminary Results

Some preliminary results of this scheme have been collected and they will be discussed in general here. Unfortunately, the hypothesis that the asymptotic approach to the extreme probability values, 0 and 1, appears to have only provided a partial answer in diagnosing the source of instability in graph growth/size. The insufficiency of this approach to manage the problem, however, lends credence to another potential

source. Namely, this potential alternative source of instability may be a consequence of the fact that activation values are centered at 0, which represents a predicted probability of 0.5 in the market space. Since activation values are centered at 0, the inhibitory (negative integers) and excitatory (positive integers) signals may produce the same composite (net result/sum) in an infinite number of ways. For example, $-100 + 200 = 100$, $-101 + 201 = 100$, $-1001 + 1101 = 100$, etc. As a result, a more promising solution to the problem being confronted would likely acknowledge the net magnitude of the output stimulus explicitly. Such an approach may be eliminative or bounding in nature, or it could map instances like "$-100 + 200 = 100$" and "$-1001 + 1101 = 100$" to different values.

Further, it may very well be the case that some of the missing market features (such as the explicit limitation of resource consumption through a reasoner/firm "budget") could provide some relief. For example, if both "$-100 + 200 = 100$" and "$-1000 + 1100 = 100$" are allowed to retain the same interpretation (i.e. probability value), but the second of the two requires 7 times ($= \frac{2100}{300}$) the amount of "energy" to produce the same result, while the market "return" for correct predictions is fixed (and, possibly, significantly less than 2100), then activations with a large net magnitude would be selected against. While it is unlikely that such a scheme is capable of resolving the stability problem alone, it presents itself as an additional mechanism favoring economical resource usage.

In any case, the performance (accuracy) of the models is largely retained and the use of non-extreme, market equilibrium values in training helps prevent individual models from becoming trivial prediction functions (predicting all 0s or 1s). It will be the subject of future work to better manage graph stability and characterize what a successful "market-ensemble" looks like.

CHAPTER 5

CONCLUSION

The present work successfully introduced a new learning paradigm called Parafinitary Learning, which is capable of efficiently adapting and growing a network structure to manage target input stimuli. Though there exist some outstanding features and issues that remain to be resolved (e.g. stability, market shares/ resources, etc.), the results are promising and it is believed that future work will improve upon the shortcomings of the current scheme. Thank you for your time.

BIBLIOGRAPHY

[1] Hebb, D.O. "The Organization of Behavior: A Neuropsychological Theory." *Routledge & CRC Press*, Taylor & Francis Group, 1949.

[2] Cech, Donna. "Neural Plasticity - an Overview." *ScienceDirect*, 2012.

[3] Widrow, Bernard, et al. "The Hebbian-LMS Learning Algorithm." *IEEE Xplore*, 2015.

[4] Lowel, S. and Singer, W. "Selection of Intrinsic Horizontal Connections in the Visual Cortex by Correlated Neuronal Activity." *Science*, American Association for the Advancement of Science, 10 Jan. 1992.

[5] "Hebbian Theory." *Wikipedia*, Wikimedia Foundation, 4 May 2021.

[6] Euliano, Neil. "Neural and Adaptive Systems: Fundamentals Through Simulations." *Neural and Adaptive Systems: Fundamentals Through Simulation*, Principe, 1997.

[7] Oja, Erkki. "Simplified Neuron Model as a Principal Component Analyzer." *Journal of Mathematical Biology*, Springer-Verlag, 1982.

[8] Sanger, Terence D. "Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network." *Neural Networks*, Pergamon, 6 Mar. 2003.

[9] Haykin, Simon O. "Neural Networks: A Comprehensive Foundation." *Pearson*. 2005.

[10] Pfenning, Frank. "Lecture Notes on Harmony." Linear Logic, Carnegie Mellon University, 23 Jan. 2012, .

[11] Restall, Greg. "Substructural Logics." *Stanford Encyclopedia of Philosophy*, Stanford University, 21 Feb. 2018.

[12] Di Cosmo, Roberto, and Dale Miller. "Linear Logic." *Stanford Encyclopedia of Philosophy*, Stanford University, 24 May 2019, .

[13] Pfenning, Frank. 15-816 Linear Logic, Carnegie Mellon University, 2012, .

[14] Murfet, Daniel. "Linear Logic and Deep Learning." *The Rising Sea*, June 2016.

[15] Murfet, Daniel. "Logic and Linear Algebra: an Introduction." *ArXiv.org*, 4 Jan. 2017.

[16] Clift, James and Murfet, Daniel. "Encodings of Turing Machines in Linear Logic." *ArXiv.org*, 28 May 2018.

[17] Nolt, John. "Free Logic." *Stanford Encyclopedia of Philosophy*, Stanford University, 11 Dec. 2020.

[18] Wittgenstein, Ludwig. "Philosophical Investigations." *Blackwell*, 1953.

[19] "Dynamical System." *Wolfram MathWorld*.

[20] "Basin of Attraction." *Wolfram MathWorld*.

[21] "Attractor." *Wolfram MathWorld*.

[22] Fahlman, Scott and Lebiere, Christian. "Cascade-Correlation." *Springer*, 1990.

[23] He, Xin, et al. "AutoML: A Survey of the State-of-the-Art." *Knowledge-Based Systems*, Elsevier, 24 Nov. 2020.

[24] "Base." *Wolfram MathWorld*.

[25] Etiemble, Daniel. "Ternary Circuits: Why R=3 Is Not the Optimal Radix for Computation." *ArXiv.org*, 19 Aug. 2019.

[26] Hayes, Brian. "Third Base." *American Scientist*, vol. 89, no. 6, pp. 490–494, 2001.

[27] Hurst. "Multiple-Valued Logic-Its Status and Its Future." *IEEE Xplore*, 1984.