Eur. Phys. J. Special Topics **229**, 2629–2738 (2020)
© EDP Sciences, Springer-Verlag GmbH Germany, part of Springer Nature, 2020
https://doi.org/10.1140/epjst/e2020-000098-9

THE EUROPEAN PHYSICAL JOURNAL
SPECIAL TOPICS

Review

Evolution of novel activation functions in neural network training for astronomy data: habitability classification of exoplanets

Snehanshu Saha[1,a], Nithin Nagaraj[2], Archana Mathur[3], Rahul Yedida[4], and Sneha H R[3]

[1] CSIS and APPCAIR, BITS Pilani K K Birla, Goa Campus, Sancoale, India
[2] Consciousness Studies Programme, National Institute of Advanced Studies, Bengaluru, India
[3] Nitte Meenakshi Institute of Technology, Bengaluru, India
[4] North Carolina State University, Raleigh, USA

Received 8 May 2020 / Accepted 2 September 2020
Published online 9 November 2020

Abstract. Quantification of habitability is a complex task. Previous attempts at measuring habitability are well documented. Classification of exoplanets, on the other hand, is a different approach and depends on quality of training data available in habitable exoplanet catalogs. Classification is the task of predicting labels of newly discovered planets based on available class labels in the catalog. We present analytical exploration of novel activation functions as consequence of integration of several ideas leading to implementation and subsequent use in habitability classification of exoplanets. Neural networks, although a powerful engine in supervised methods, often require expensive tuning efforts for optimized performance. Habitability classes are hard to discriminate, especially when attributes used as hard markers of separation are removed from the data set. The solution is approached from the point of investigating analytical properties of the proposed activation functions. The theory of ordinary differential equations and fixed point are exploited to justify the "lack of tuning efforts" to achieve optimal performance compared to traditional activation functions. Additionally, the relationship between the proposed activation functions and the more popular ones is established through extensive analytical and empirical evidence. Finally, the activation functions have been implemented in plain vanilla feed-forward neural network to classify exoplanets. The mathematical exercise supplements the grand idea of classifying exoplanets, computing habitability scores/indices and automatic grouping of the exoplanets converging at some level.

[a] e-mail: scibase.snehanshu@gmail.com

Contents

1 Introduction

Since time immemorial, humanity was looking at cosmos and believing other worlds being out there, inhabited with other or, may be same, beings like us. Indian ancient texts talk about travelling to other worlds in "bodily form" (inscription on the iron pillar at Qutub Minar, probably left in 4th century BC). Ancient Greeks also believed in the existence of other planetary bodies with beings living on them (with mentions dating as far back as 6th century BC: Thales of Miletus and Pythagoras). With our technological advances, we are continuing this same quest – the quest for the habitable planet, ultimately, the second Earth; or at the least, for the answer to the question of whether we are alone in the Universe. Currently, we already know about the existence of thousands of exoplanets, and the estimates of the actual number of planets exceed the number of stars in our Galaxy alone by orders of magnitude (both bound and free-floating); small rocky planets, super-Earths, the most abundant type. Our interest in exoplanets lies in the fact that, anthropically, we believe that life can only originate and exist on planets, therefore, the most fundamental interest is in finding a habitable planet – the planet with life on it. This quest can be broadly classified into the following: looking for Earth-like conditions or the planets similar to the Earth (Earth similarity), and looking for the possibility of life in a form known or unknown to us (habitability). But what is habitability? Is it the ability of a planet to beget life – a potential habitability? Or is it our ability to detect it: a planet may host life as we know it, in other words, be inhabited, but we will not detect it unless it evolved sufficiently to change the environment on a planetary scale. In both cases, the only comparison for recognition is our planet, therefore, we are looking for the terrestrial likeness in exoplanets.

Exoplanets are planets that are found to be orbiting around stars beyond our solar system. NASA's Kepler mission and TESS have found more than 4000 exoplanets in the universe. The most exciting question that scientists want to explore is the potency of life on these planets. Apparently, the possibility of life is largely due to the presence of liquid water, and similar related characteristics exhibited by these planets. Hence, if a planet is at a reasonable distance from its star, it exhibits possibility of being habitable because correct temperature range could support water on its surface. Out of 4000 planets, quite a few are observed to be in the habitable zone of their star. Statistically, there are 40 Exoplanets that have been discovered in the year 2020, and the most interesting discovery of the latest earth-like planet is TOI 700d. Measurements like size, mass, composition and density of TOI 700d are close to those of the earth. TOI 700d orbits around its star with the orbital period of 37.4 days and its distance from the star supports a temperature range that could possibly harbour life. Another interesting discovery, Kepler-1649c, which orbits a red-dwarf star is also found to be a potentially habitable exoplanet. It is located at the habitable distance from its star, and there is a possibility that liquid water may exist on its surface. Kepler-1649c is uncovered to be potentially habitable as its size and temperature are found to be earth-like though its surface is rocky. With large number of exoplanets being discovered each year, the scope to explore planets outside solar system in terms of chances to support life increases by many folds. Though, there are many aspects which are still unexplored like presence of an atmosphere on these exoplanets. Another exciting exoplanet TRAPPIST 1d, which orbits around a ultracool dwarf star, about 40 light years away from earth, has been placed within the habitable zone. It has a terrestrial atmosphere and has about 5% of its surface in liquid water that could possibly be life friendly. The mass, radius, surface gravity and composition of TRAPPIST-1d are very similar to those of earth that makes the exoplanet very intriguing in the exploration of habitability.

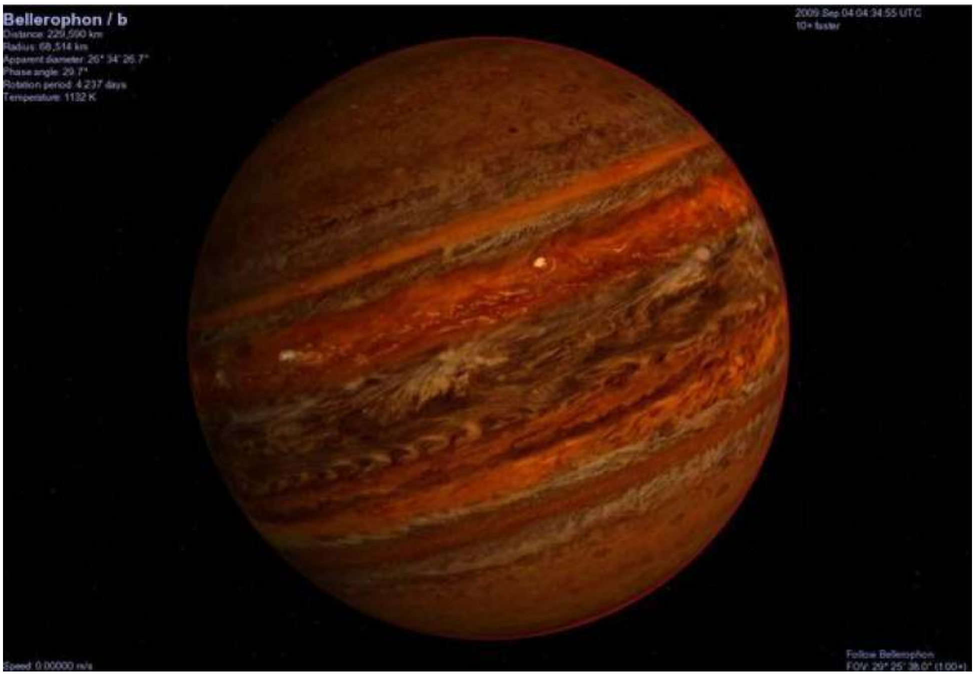


Fig. 1. 51-Pegasi-b: Image of the first exoplanet discovered, image taken from [1] is an artistic representation of the planet.

Back in the year 1995, two Swiss astronomers (Didier Patrick Queloz and Michel Mayor) made an exciting discovery of the first extra-solar planet that orbits around its star outside our solar system. This discovery completely changed the perspective of the astronomers and lead to beginning of an era where more and more exoplanets were discovered and their physical conditions and chemical compositions were studied. Didier Patrick Queloz is a professor of Physics at the University of Cambridge. He obtained his MSc from the University at Geneva. He also received his DEA in Astronomy and Astrophysics and subsequently obtained his PhD under the supervision of Michel Mayor. He is also a fellow of Trinity College, Cambridge. Michel Mayor is an astrophysicist in the department of Astronomy at University of Geneva. In 1995, Didier and his advisor Mayor discovered first extrasolar planet 51 Pegasi b (Fig. 1), that orbits around a sun-like star located 50 light years from earth. This discovery lead to the most significant revolution in astronomy because until then, no exoplanet that exist outside our solar system had been found. This discovery had transformed the idea about cosmic world and more than 4000 exoplanets have been discovered in the universe since then. Both Didier and his advisor Michel have been jointly awarded the 2019 Nobel prize in Physics for their pioneering advances for discovering the exoplanet. As more planets are being discovered, the curiosity of finding possibility of life is also building up and consequently, extensive studies related to life-sustaining characteristics of already discovered extra-solar planets is gaining momentum.

1.1 Habitability: Quantification or classification?

Quantification of habitability is a challenging task, for a couple of reasons at least. The first is to be able to define an index for habitability and more importantly, address questions about such indices, if in vogue, are good indicators or not. There exists a general public and scientist common misinterpretation of habitability concept. A very small group of astronomers has been vocal against habitability quantification. This does not represent the consensus of the community which is still developing habitability indices in one form or another. A recent white paper by Mendez et al. [2] argues for habitability fundamentally reliant on the availability of raw materials and energy to assemble and maintain life. The authors presented arguments in favor of analogy between habitability and the classical elements. They also formalized the concept in to a general quantitative framework. If habitability is proportional to the mass and energy available for life, their definition is consistent with the habitability models developed in ecology and accommodates energy considerations. The authors claim their model is applicable to any type of environment, from micro-environments to entire biospheres. While we wait for the ramifications and working of the model (the habitability master equation) to be made available to the community in due time, we propose an alternative strategy to examine habitability in quantitative manner. We begin by asking a couple of pertinent questions. Shall we adopt full-scale Machine Learning approach, with other planets as training set and be able to answer questions such as "Is this new planet potentially habitable based on training data?" and "How potentially habitable is this new planet based on our experience of previously labelled ones with identical attributes?" This, however, depends on the quality of training data and classification paradigms of choice. For example, we can consider three classes: non-habitable, mesoplanet and psychroplanet (refer to Sect. 4.1) as training data with labels for a supervised inference.

With a constantly increasing number of discovered exoplanets and the possibility that stars with planets are a rule rather than an exception, it became possible to begin characterizing exoplanets in terms of planetary parameters, types, populations and, ultimately, in the habitability potential. This is also important in understanding the formation pathways of exoplanets. But, since complete appraisal of the potential habitability needs the knowledge of multiple planetary parameters which, in turn, requires hours of expensive telescope time, it became necessary to prioritise the planets to look at, to develop some sort of a quick screening tool for evaluating habitability perspectives from observed properties. Here, the quick selection is needed for a long painstaking spectroscopic follow-up to look for the tell-tales of life – the biosignatures – atmospheric gases that only living organisms produce in abundance. It can be oxygen, ozone, methane, carbon dioxide or, better, their combinations ([3,4], and references therein). For all the upcoming space missions dedicated to the search of life in the Universe: PLATO, Euclid, JWST, etc., we need to make a list of our preferred candidates, so that this quest hopefully can be completed within our lifetimes. Extrapolation of Kepler's data shows that in our Galaxy alone there could be as many as 40 billion such planets [5–7]. And it is quite possible that soon we may actually detect most of them. But with the ultimate goal of a discovery of life, astronomers do not have millennia to quietly sit and sift through more information than even pentabytes of data.

However, all classification strategies have caveats, and some [8] reject the exercise entirely on the basis of impossibility to quantitatively compare habitability, and on the idea that pretending otherwise can risk damaging the field in the eyes of the public community. In addition, some researchers believe that the priority for the exoplanet and planetary science community is to explore the diversity of exoplanets, and not to concentrate on exclusions. However, it is not impossible to quantify habitability and the scientific community has been doing this for decades [2]. A classification strategy depends on available data and makes inference based on it, rather than trying to quantify habitability by defining a range of values. For example, a classification algorithm with probabilistic herding that facilitates the assignment of exoplanets to appropriate classes via supervised feature learning methods, producing granular clusters of habitability, may be a more rational alternative. Classification methods rely on using Machine Learning algorithm to construct and test planetary habitability functions with exoplanet data. The goal is to identify potentially habitable planets from exoplanet dataset. These methods provide one important step toward automatically identifying objects of interest from large datasets by future ground and space observatories.

1.2 Classification of exoplanets

Astronomers and philosophers have been intrigued by the fact that the Earth is the only habitable planet within the solar system. There has been scant evidence or explorations in the direction of finding planets outside the solar system which may harbor life. Essentially, the mission to find “Earth 2.0” gained momentum in recent times with NASA spearheading Kepler [9] and other missions. Initial missions exploring Earth’s neighbors, Mars and Venus, did not yield promising results. However, NASA’s missions in the past two decades led to discoveries of hundreds of exoplanets – planets that are outside our solar system, also known as extra-solar planets. The missions are carried out based on the inference that planets around stars are more frequent and the fact that actual number of planets far exceeds the number of stars in our galaxy. The mandate is to look for interesting samples from the pool of recently discovered exoplanet database [10]. Additionally, finding “Earth 2.0” based on similarity metric [9,11,12] is not the only approach. In fact, the approach has its own limitations as astronomers argue that a distant “Earth-similar” planet may not be habitable and could be just statistical mirage [13]. Therefore, a classification approach should be used to vet habitability scores of exoplanets in order to ascertain the probability of distant planets harboring life. This begets the need for sophisticated pipelines. Goal of such techniques would be to quickly and efficiently classify exoplanets based on habitability classes. This is equivalent to a supervised classification problem.

The process of discovery of exoplanets involves very careful analysis of stellar signals and is a tedious and complicated process, as outlined by Bains et al. [14]. This is due to the smaller size of exoplanets compared to other types of stellar objects such as stars, galaxies, quasars, etc. Radial velocity-based techniques and gravitational lensing are most popular methods to detect exoplanets. Given the rapid technological improvements and the accumulation of large amounts of data, it is pertinent to explore advanced methods of data analysis to rapidly classify planets into appropriate categories based on the physical characteristics.

The habitability problem has been tackled in different ways. Explicit Earth-similarity score computation [12] based on parameters mass, radius, surface temperature and escape velocity developed into Cobb–Douglas Habitability Score helped identify candidates with similar scores to Earth. However, using Earth-similarity alone [15] to address habitability is not sufficient unless model based evaluations [16] are interpreted and equated with feature based classification [17]. However, when we tested methods in [17], it was found that the methods did not work well with pruned feature sets (features which clearly mark different habitability classes were removed). Therefore, new machine learning methods to classify exoplanets are a necessity.

To address this need for efficient classification of exoplanets, we embark on design of novel activation functions in Neural networks with sound mathematical foundations. We shall justify the need to go beyond Sigmoid, hyperbolic tangent and ReLU activation functions and demonstrate the superior performance of the proposed activation functions in habitability classification of exoplanets. Such a fundamental approach to solving a classification problem has obvious merits and the activation functions that we propose could be readily applied to problems in other areas as well.

Neural networks [18] or Artificial Neural network (ANN) are systems of interconnected units called "Neurons" organized in layers (loosely inspired by the biological brain). ANN is a framework for many different machine learning algorithms to process complex input data (text, audio, video etc.) in order to "learn" to perform classification/regression tasks by considering examples and without the aid of task-specific programming rules. As of today, ANNs are found to yield state-of-the-art performance in a variety of tasks such as speech recognition, computer vision, board games and medical diagnosis [19,20] and classification of exoplanets [17].

Every neuron in ANNs is followed by an *activation function* which defines the output of that neuron given its inputs. It is an abstraction representing the rate of action potential firing in the neuron. In most cases, it is a binary non-linear function – the neuron either fires or not. The celebrated sigmoidal activation function enables a feed-forward network with a single hidden layer containing a finite number of neurons to approximate a wide variety of linear and non-linear functions [21]. Recent works have shown that activation functions allow neural networks to perform complex tasks, including gene expression analysis [20] and solving nonlinear equations [19]. The most popular activation functions are sigmoid, hyperbolic tangent (tanh) and ReLU (Rectified Linear Unit). Neural networks, although a powerful engine, often require expensive "parameter-tuning" efforts. Classification of exoplanets is an intriguing problem and extremely hard, even for a neural network to solve, especially when clear markers for separation of habitability classes (i.e. features such as surface temperature [22] and features related to surface temperature) are removed from the feature set. To this end, a version of ANNs, replicated weight neural network (RWNN), in conjunction with sigmoid activation, has been applied to the task of classification of exoplanets [17]. However, the results turned out to be less than satisfactory. Added to that, the effort in parameter tuning [23], the outcome is far from desired when applied to pruned feature sets used in this paper (please refer to the Data section).

In this work, we explore novel activation functions as a consequence of integration of several ideas leading to implementation and subsequent use in habitability classification of exoplanets. The relationship between the proposed activation functions and the existing popular ones will be established through extensive analytical and empirical evidence.

One of the key reasons to classify exoplanets is the limitation of finding out habitability candidates by Earth similarity alone, as proposed by several metric based evaluations, namely the Earth Similarity Index, Biological Complexity Index [24], Planetary Habitability Index [10] and Cobb–Douglas Habitability Score (CDHS) [12]. In [25], an advanced tree-based classifier, Gradient Boosted Decision Tree (improved version of Decision tree [26]) was used to classify Proxima b and other planets in the TRAPPIST-1 system. The accuracy was near-perfect, providing a baseline for other machine classifiers [27]. However, that paper considered surface temperature as one of the attributes/features, making the classification task significantly easier than the proposed one in the current manuscript.

2 Motivation and contribution

Observing exoplanets is no easy task. In order to draw a complete picture of any exoplanet, observations from multiple missions are usually combined. For instance, the method of radial velocity can give us the minimum mass of the planet and the distance of the planet from its parent star, transit photometry can provide us with information on the radius of a planet, and the method of gravitational lensing can provide us the information regarding the mass. In contrast to this, stars are generally easier to observe and profile through various methods of spectrometry and photometry in different ranges of wavelengths of the electromagnetic spectrum. Hence, by the time a planet has been discovered around a star, we would possibly have fairly rich information about the parent star, and this includes information such as the luminosity, radius, temperature, etc. Therefore, the uniqueness of the approach is to classify exoplanets based on smaller sets of observables as features of the exoplanet along with multiple features of the parent star. We elaborate each case of carefully selected features in the Experiments section. We note, most of the sub-sets of features created from PHL-EC do not contain surface temperature. Added to that, these surface temperature values are modeled values and not measured values. In fact, the most interesting result of the paper might be that some of the methods are unable to exactly reproduce the correct classification, given the strict dependence of the classification on a single feature, surface temperature namely. Surface temperature is a hard marker when it comes to classifying exoplanets. Removing this feature enhances the complexity of classification many folds. Our manuscript tackles the challenge by adopting a foundational approach to activation functions.

However a reasonable question arises. When there exist activation functions in literature with evidence of producing good performance, is there a need to define a new activation function? If indeed the necessity is argued, can the new activation function be related to existing ones? We are motivated by these questions and painstakingly address these in subsequent Sections 4–9 in the manuscript. We show the activation functions proposed in the paper, SBAF and A-ReLU are indeed generalizations of popular activations, Sigmoid and ReLU respectively. Moreover, the theory of Banach spaces and contraction mapping has been exploited to show that SBAF can be interpreted as a solution to first order differential equation. Further, the theory of fixed point and stability has been used to explain the amount of effort saved in tuning parameters of the activation units. In fact, this is one of the hallmarks of the paper where we intend to show that our activation functions implemented to tackle the classification problem are "thrifty" in comparison to Sigmoid and ReLU. We demonstrate this by comparing system utilization metrics such as runtime, memory and CPU utilization. Additionally, we show that SBAF also satisfies the Universal Approximation Theorem and can be related to regression under uncertainty. We also demonstrate mathematically and otherwise that the approximation to ReLU, defined as A-ReLU is continuous and differentiable at "knee-point" and establish the fact that A-ReLU does not require much parameter tuning. Extensive experimentation shows conclusively, the insights gained from the mathematical theory are backed up by performance measures, beating Sigmoid and ReLU. Another commonly used approach found in many texts on machine learning/deep learning is the application of several different methods on the same data set and choose the one with the best performance. We, in the present work, chose to deviate from such approaches as it fails to provide solid reasons to choose one method over another before experiments begin. Rather, our approach is the search for a method with lasting impressions in literature, by focusing on the theoretical nuances. In summary, we state the following to argue for such rigorous and mathematical approach to activation functions in Neural Network based classification.

- Instead of a long list of comparative performance among different classifiers, we propose a parsimonious way to run a neural network.
- We propose "activation" reliant method. Dense neural networks such as RWNN or GAN are replaced by proposing novel activation functions in a simpler neural network architecture (see Sects. 5 and 6).

– We presented a bare-bones implementation of our method from scratch, called SYM-Net, instead of porting standard and already optimized libraries. These libraries are available on Github[1] (see Sect. 9).
– Instead of dealing with the process of tuning parameters in activation units, we proposed methods to choose optimal parameter settings (see Sects. 7 and 8). These sections provide arguments and proofs in favor of viable alternative to dense network architectures discussed in Section 4.
– All of the items listed above required rigorous mathematical investigation including proving theoretical results.

Apart from original contributions to machine learning literature, the theoretical exercise helped us reach optimal performance to a challenging classification problem. The presentation of proposed analytical methods also are motivated by optimal system/hardware utilization as many deep learning architectures in recent times trigger concerns about heavy carbon footprint [28]. We will show, in the due course, that our method is thrifty in computing resource utilization compared to other activation functions commonly used.

3 Exoplanets: Habitability indices

In the last decade, thousands of planets are discovered in our Galaxy alone. The inference is that stars with planets are a rule rather than exception [29], with estimates of the actual number of planet exceeding the number of stars in our Galaxy by orders of magnitude [30]. The same line of reasoning suggests a staggering number of at least 10^{24} planets in the observable Universe. The biggest question posed therefore is whether there are other life-harbouring planets. The most fundamental interest is in finding the Earth's twin. In fact, *Kepler* space telescope [31] was designed specifically to look for Earth's analogs – Earth-size planets in the habitable zones (HZ) of G-type stars [32]. More and more evidence accumulated in the last few years suggests that, in astrophysical context, Earth is an average planet, with average chemistry, existing in many other places in the Galaxy, average mass and *size*. Moreover, recent discovery of the rich organic content in the protoplanetary disk of newly formed star MWC 480 [33] has shown that neither is our Solar System unique in the abundance of the key components for life. Yet the only habitable planet in the Universe known to us is our Earth.

The question of habitability is of such interest and importance that the theoretical work has expanded from just the stellar HZ concept to the Galactic HZ [34] and, recently, to the Universe HZ – asking a question which galaxies are more habitable than others [35]. However, the simpler question – which of thousands detected planets are, or can be, habitable is still not answered. Life on other planets, if exists, may be similar to what we have on our planet, or may be in some other unknown form. The answer to this question may depend on understanding how different physical planetary parameters, such as planet's orbital properties, its chemical composition, mass, radius, density, surface and interior temperature, distance from its parent star, even parent star's temperature or mass, combine to provide habitable conditions. With currently more than 1800 confirmed and more than 4000 unconfirmed discoveries[2], there is already enormous amount of accumulated data, where the challenge lies in the selection of how much to study about each planet, and which parameters are of the higher priority to evaluate.

[1] https://github.com/sahamath/SYM-Net
[2] Extrasolar Planets Encyclopedia, http://exoplanet.eu/catalog/

Several important characteristics were introduced to address the habitability question. Reference [36] first addressed this issue through two indices, the Planetary Habitability Index (PHI) and the Earth Similarity Index (ESI), where maximum, by definition, is set as 1 for the Earth, PHI=ESI=1.

ESI represents a quantitative measure with which to assess the similarity of a planet with the Earth on the basis of mass, size and temperature. But ESI alone is insufficient to conclude about the habitability, as planets like Mars have ESI close to 0.8 but we cannot still categorize it as habitable. There is also a possibility that a planet with ESI value slightly less than 1 may harbor life in some form which is not there on Earth, i.e. unknown to us. PHI was quantitatively defined as a measure of the ability of a planet to develop and sustain life. However, evaluating PHI values for large number of planets is not an easy task. In [37], another parameter was introduced to account for the chemical composition of exoplanets and some biology-related features such as substrate, energy, geophysics, temperature and age of the planet – the Biological Complexity Index (BCI). Here, we briefly describe the mathematical forms of these parameters.

Earth Similarity Index (ESI). ESI was designed to indicate how Earth-like an exoplanet might be [36] and is an important factor to initially assess the habitability measure. Its value lies between 0 (no similarity) and 1, where 1 is the reference value, i.e. the ESI value of the Earth, and a general rule is that any planetary body with an ESI over 0.8 can be considered an Earth-like. It was proposed in the form

$$\mathrm{ESI}_x = \left(1 - \left|\frac{x - x_0}{x + x_0}\right|\right)^w , \tag{1}$$

where ESI_x is the ESI value of a planet for x property, and x_0 is the Earth's value for that property. The final ESI value of the planet is obtained by combining the geometric means of individual values, where w is the weighting component through which the sensitivity of scale is adjusted. Four parameters: surface temperature T_s, density D, escape velocity V_e and radius R, are used in ESI calculation. This index is split into interior ESI_i (calculated from radius and density), and surface ESI_s (calculated from escape velocity and surface temperature). Their geometric means are taken to represent the final ESI of a planet. However, ESI in the form (1) was not introduced to define habitability, it only describes the similarity to the Earth in regard to some planetary parameters. For example, it is relatively high for the Moon.

Planetary Habitability Index (PHI). To actually address the habitability of a planet, reference [36] defined the PHI as

$$\mathrm{PHI} = (S \cdot E \cdot C \cdot L)^{1/4} , \tag{2}$$

where S defines a substrate, E – the available energy, C – the appropriate chemistry and L – the liquid medium; all the variables here are in general vectors, while the corresponding scalars represent the norms of these vectors. For each of these categories, the PHI value is divided by the maximum PHI to provide the normalized PHI in the scale between 0 and 1. However, PHI in the form (2) lacks some other properties of a planet which may be necessary for determining its present habitability. For example, in Shchekinov et al. [38] it was suggested to complement the original PHI with the explicit inclusion of the age of the planet [3] (see their Eq. (3)).

Biological Complexity Index (BCI). To come even closer to defining habitability, yet another index was introduced, comprising the above mentioned four parameters of the PHI and three extra parameters, such as geophysical complexity G, appropriate temperature T and age A [37]. Therefore, the total of seven parameters were initially considered to be important for the BCI. However, due to the lack of information on chemical composition and the existence of liquid water on exoplanets, only five were retained in the final formulation,

$$\mathrm{BCI} = (S \cdot E \cdot T \cdot G \cdot A)^{1/5}\,. \tag{3}$$

It was found in [37] that for 5 exoplanets the BCI value is higher than for Mars, and that planets with high BCI values may have low values of ESI.

All previous indicators for habitability assume a planet to reside within in a classical HZ of a star, which is conservatively defined as a region where a planet can support liquid water on the surface [39,40]. The concept of an HZ is, however, a constantly evolving one, and it has been since suggested that a planet may exist beyond the classical HZ and still be a good candidate for habitability [41,42]. Though presently all efforts are in search for the Earth's twin where the ESI is an essential parameter, it never tells that a planet with ESI close to 1 is habitable. Much advertised recent hype in press about finding the best bet for life-supporting planet – Gliese 832c with ESI = 0.81 [43], was thwarted by the realization that the planet is more likely to be a super-Venus, with large thick atmosphere, hot surface and probably tidally locked with its star.

The remainder of the paper is organized as follows. We begin with a general introduction to classification of objects in Astronomy by supervised machine learning techniques. This includes a primer on data processing and context in classification, followed by deep architectural interventions in neural networks to achieve reasonable classification performance metrics. Section 4 is an elaborate journey in that direction. As deep neural networks are difficult to interpret, we endeavor to treat neural networks with greater mathematical rigor in the following sections. We hope to not only accomplish better results but are able to offer explanations for such enhanced performance. Section 5 offers guiding principles and a novel approach to train neural networks, backed by mathematical proofs. A novel activation function to train an artificial neural network (ANN) is introduced in the same section. We discuss the theoretical nuances of such a function in Section 5. We relate SBAF to binary logistic regression in Section 6. The following section presents a mathematical treatment of the evolution of SBAF from theory of differential equations. Next section introduces A-ReLU as an approximation exercise. We follow up with network architecture in the the next section, detailing the back propagation mechanism paving the foundation for ANN based classification of exoplanets. Consequently, the following section presents results on all data sets with performance comparison including system utilization. Before we proceed to concluding remarks, we re-assess the habitable candidate determination problem by an unsupervised clustering method where we try to pose the habitable planets as anomaly detection problem from a large pool of exoplanets from the PHL-EC. The goal is to cross-match these candidates obtained via clustering with the predicted class labels determined by supervised methods as explained in Sections 5–9. We conclude by discussing the efficacy of the proposed method.

4 Classification of astronomical objects: Data is the key

The efficacy of machine learning (ML) in characterizing exoplanets into different classes is a fascinating problem and never been attempted before. The source of the data used in this work is University of Puerto Rico's Planetary Habitability Laboratory's Exoplanets Catalog (PHL-EC). The predictive analysis on exoplanets demands a detailed investigation of the structure of the data and methods that can be used to effectively categorize new exoplanet samples. The approach needs to be two-fold. We elaborate on the results obtained by using ML algorithms by stating the accuracy of each method used and propose a paradigm to automate the task of exoplanet classification for relevant outcomes. In particular, we focus on the results obtained by novel neural network architectures for the classification task, as they have performed very well despite complexities that are inherent to this problem. The exploration led to the development of new methods fundamental and relevant to the context of the problem and beyond. The data exploration and experimentation also result in the development of a general data methodology and a set of best practices which can be used for exploratory data analysis experiments.

For hundreds of years, astronomers and philosophers have considered the possibility that the Earth is a very rare case of a planet as it harbors life. This is partly because so far, missions exploring specific planets like Mars and Venus have found no traces of life. However, over the past two decades, discoveries of exoplanets have poured in by the hundreds and the rate at which exoplanets are being discovered is increasing. Scientists are now discussing the conditions, circumstances, and various possibilities that can lead to the emergence of life [14]. In order to find interesting samples from the massive ongoing growth in the data, a sophisticated computational pipeline should be developed which can quickly and efficiently classify exoplanets based on their physical properties into relevant habitability classes.

Machine classification requires features of data. The features we use for demonstrating the efficacy of the classifiers are small in number and are related to the real-world observations of exoplanets. For this, we use the data from PHL's exoplanet catalog, and at the present time, given that the number of samples that are potentially habitable (in the psychroplanet and mesoplanet classes, respectively; elaborated in Sect. 4.1), we use all the samples available, but use only a very small set of planetary parameters. The goal is to develop a pipeline based on various approaches which can detect interesting exoplanetary samples in databases, quickly after their discovery, thus potentially accelerating the thermal characterization of exoplanets before all the attributes are fully discovered and cataloged.

4.1 Data for classification

Combined with stellar data from the Hipparcos catalog [44], the PHL-EC dataset [10,45] consists of a total of 68 features (of which 13 are categorical and 55 are continuous valued) and more than 3800 confirmed exoplanets (at the time of writing of this paper); the catalog includes important features like atmospheric type, mass, radius, surface temperature, escape velocity, earth's similarity index, flux, orbital velocity etc. The PHL-EC consists of observed as well as estimated attributes. Hence, it presents interesting challenges from the analysis point of view. There are six classes in the dataset, of which we can use three in our analysis as they are sufficiently large in size. These are namely non-habitable, mesoplanet, and psychroplanet classes. These three classes or types of planets can be described on the basis of their thermal properties as follows:

(1) **Mesoplanets**: these are referred to as M-planets. These planets have mean global surface temperature between $0\,^{\circ}\mathrm{C}$ and $50\,^{\circ}\mathrm{C}$, a necessary condition for complex terrestrial life. These are generally referred as Earth-like planets.

(2) **Psychroplanets**: these planets have mean global surface temperature between $-50\,^{\circ}$C and $0\,^{\circ}$C. Hence, the temperature is colder than optimal for sustenance of terrestrial life.
(3) **Non-habitable**: planets other than mesoplanets and psychroplanets do not have thermal properties required to sustain life.

The remaining three classes in the data are those of thermoplanet, hypopsychroplanet and hyperthermoplanet. However, at the time of writing this paper, the number of samples in each of these classes is too small (each class has less than 3 samples) to reliably take them into consideration for an ML-based exploration.

While running the classification methods, we consider multiple features of the parent sun of the exoplanets that include mass, radius, effective temperature, luminosity, and the limits of the habitable zone. In addition to this we consider the following planetary attributes in separate experimental settings:

(1) **Minimum mass** and **Distance from the parent star**, corresponding to observations of radial velocity.
(2) **Mass**, corresponding to observations using gravitational microlensing.
(3) **Radius**, corresponding to observations using transit photometry.

As a first step, data from PHL-EC is pre-processed. An important challenge in the dataset is that a total of about 1% of the data is missing (with a majority being of the feature P. Max Mass) and in order to overcome this, we used a simple method of removing instances with missing data after extracting the appropriate features for each experiment, as most of the missing data is from the non-habitable class. Following this, the ML approaches were used on these preprocessed datasets. The online data source for the current work is available at [46].

4.2 Dense architectures in machine classification

A classification framework works, roughly, in the following manner. There is a data set with class labels, say +1 or −1, if we assume there are only two classes, namely stars and quasars. The data set is split into three slices: training, testing and validation. The training slice is usually the largest, consisting of at least 70% of the total size, used to train the classifier so that it is able to discriminate any new data point with labels (+1/−1) accurately. The algorithm is expected to learn well so that the accuracy on the training is 100%. Next, the algorithm is tested on training and validation slices ensuring that the validation slice does not appear at all during training. The efficacy of classification is evaluated based on standard performance metrics enunciated in the results section. For details on machine classification, please refer to Appendix A. The following are the ML classification algorithms used frequently in Astronomy data and other areas of Science and Engineering.

– **Neural networks**: in recent times, artificial neural networks have become the most popular family of methods for classification [47]. Artificial neural networks draw inspiration from biology and the mechanism is roughly similar to that of the functioning of neurons. A network is trained to perform classification of labels by using gradient descent optimiser [48] that incorporates computing error gradient at every layer and optimizing the weights in each node in such a way that the error gets minimized. An example of a neural network is shown in Figure 2.
– **Other classifiers**: the other popularly used classifiers are those of tree-based classifiers, probabilistic classifiers, SVM, and KNN [49]. However, the restricted feature set used as a part of this work is too complicated for conventional machine

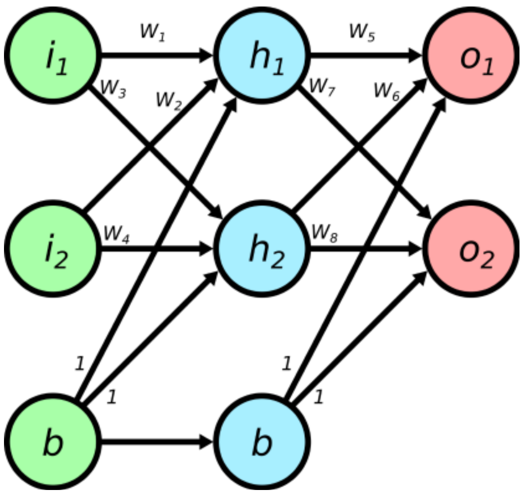


Fig. 2. The basic structure of a neural network. Here, the nodes are represented by the circles. The nodes labeled i are the input, h are hidden and o are output nodes.

learning methods to handle. The method that we focus on is that of neural networks. However, it is important to note that, the level of complexity in the data responsible for the difficulty in classification of astronomical objects presented in the current work, could not be mitigated by changing neural network architectures. This is the reason new activation functions and novel theory had to be invented.

– Neural networks can be used in two novel architectures, namely replicated weight neural networks (RWNN), and generative adversarial networks for classification. These are briefly mentioned here, in Appendices B.3.2 and B.3.4 respectively and explained in detail in Appendices B.3.2–B.3.5. These results have been further improved (better classification accuracy, machine performance, efficiency etc) by proposing novel activation functions in a simpler neural network architecture (see Sect. 5). The improvement in performance is due to dense mathematical structure and theory demystifying the traditional "black-box" operating paradigm of neural networks.

4.2.1 Replicated weight neural network (RWNN)

Usually, an Artificial Neural network (ANN) trains itself by updating an initial random choice of weights through back propagation. RWNN is a scheme to train the network, built with randomly initialized weights where learning rate and number of neurons across layers are kept same as in ANN. The first iteration generates a balanced training set by random-oversampling and trains the network. This is identical to the original vanilla feed forward neural net. However, a RWNN is different from a standard ANN architecture in the sense that the optimized weights obtained from first iteration are utilized in training network during the second iteration. Please refer to B.3.2 for details (Fig. 3).

4.2.2 Synthetic oversampling using Parzen window estimation

Artificial oversampling [50] is an approach to handle the effects of bias due to a class with a large sample size [51] in data. Typically, people try to artificially add

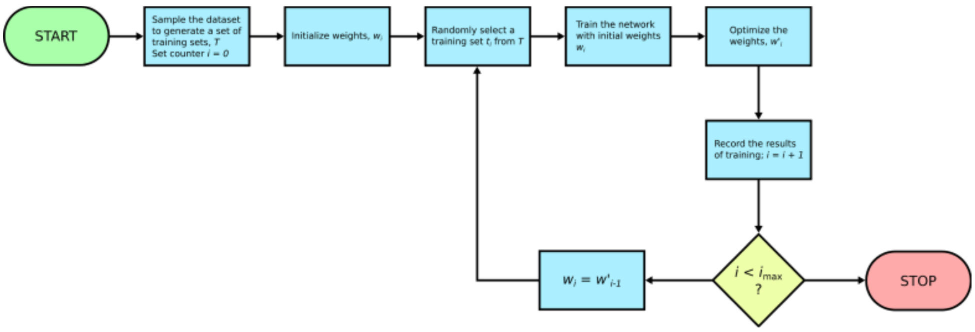

START
Sample the dataset to generate a set of training sets, T Set counter $i = 0$
Initialize weights, w_i
Randomly select a training set t_i from T
Train the network with initial weights w_i
Optimize the weights, w'_i
Record the results of training; $i = i + 1$
$i < i_{max}$?
STOP
$w_i = w'_{i-1}$

Fig. 3. Architecture pipeline for replicated weight neural network. (For a better readability please refer to the online version).

samples to the classes with lesser number of samples, namely the mesoplanet and the psychroplanet classes. Essentially, the paradigm of choice is to analyze the class-wise distribution of the data and to generate reasonable samples which reflect the general properties of their respective classes. Please refer to B.3.3 for details.

4.2.3 Generative adversarial networks

Generative adversarial networks (GANs) are a powerful framework of a pair of competing neural networks where the generative network imitates the input and discriminating network differentiates between authentic and synthetic samples. It can also be used a semi-supervised learning technique where the discriminator learns the target function based on both labeled and unlabeled generated data. Its a method worth exploring in binary classification of astronomical objects. Appendices B.3.4–B.3.6 contain details about GAN implementation on the habitability data set (PHL-EC).

4.2.4 Training the GAN for categorical semi-supervised learning

Standard GANs can be turned from a discriminator to a classifier that is only useful for determining whether a given data point x belongs to the input data distribution or not. For classification, we aim to train a discriminator that classifies the data into k categories by assigning a label y to each input sample x [52,53]. The Figure B.3 presents an overview of the architecture of the networks (Fig. 4).

We reiterate that, such architectural innovations using dense neural networks are usually not backed by hard Mathematical proofs as human neural networks are too complex to mimic. To be precise, it is often, not possible to explain why a particular architecture such as the one described here, succeeds or fails! *For a comprehensive introduction on Machine Learning and Neural Networks, please refer to Appendix B.*

Section 4 provides us the necessary information on neural net architectures but do not elaborate on the alternatives. The first four sections cover the background, problem perspective and general significance of machine learning methods in the classification tasks of exoplanets. We now proceed to the crux of the problem solving which is the deep theoretical analysis of the role of the activation functions in making efficient neural network based predictions and classification. We begin Section 5 and hope that the mathematical analysis in Sections 5–7 provides a viable alternative to dense network architectures discussed in Section 4.

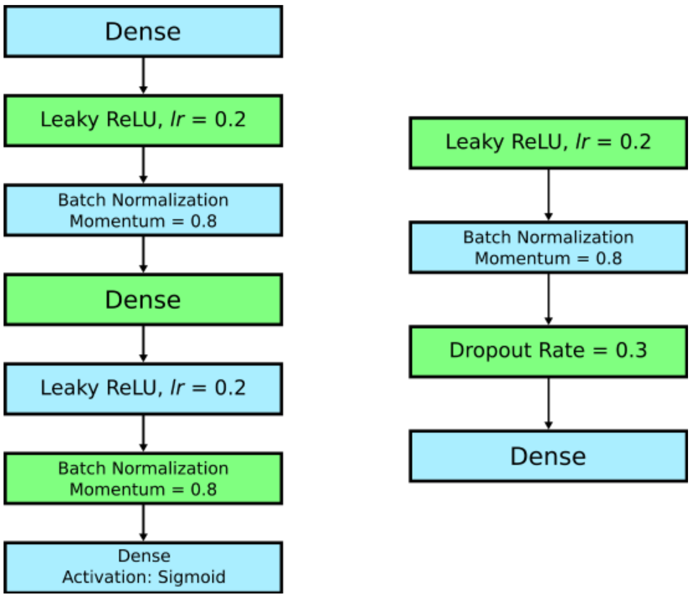

Dense
Leaky ReLU, lr = 0.2
Batch Normalization Momentum = 0.8
Dense
Leaky ReLU, lr = 0.2
Batch Normalization Momentum = 0.8
Dense Activation: Sigmoid
Leaky ReLU, lr = 0.2
Batch Normalization Momentum = 0.8
Dropout Rate = 0.3
Dense

Fig. 4. Layers of the generator and discriminator in GAN model with batch-size of 128.

5 Saha-Bora Activation Function (SBAF)

We shall introduce SBAF neuron to address the issues elicited in the previous sections. We begin with a brief description of mathematical concepts and definitions will be used throughout the remainder of the manuscript.

Definition of key terms:

– Returns to Scale: for the objective function, $y = kx^{\alpha}(1-x)^{\beta}, k > 0, 0 < \alpha < 1, 0 < \beta < 1$ feasible solution that maximizes the objective function exists, called an optimal solution under the constraints *Returns to scale*. When $\alpha + \beta > 1$, it is called increasing returns to scale (IRS) and $\alpha + \beta < 1$ is known as decreasing returns to scale (DRS). $\alpha + \beta = 1$ is known as constant returns to scale and ensures proportional output, y to inputs $x, 1-x$. y is used to define production functions in Economics [54] and the form is inspirational to designing the activation functions proposed in the manuscript. Note, we set $\beta = 1 - \alpha$ in the activation function formulation ensuring CRS and therefore existence of global optima.
– Absolute error is the magnitude of the difference between the exact value and the approximation.
– Relative error is the absolute error divided by the magnitude of the exact value.
– Banach Space: a complete normed vector space i.e. the Cauchy sequences have limits. A Banach space is a vector space.
– Contraction Mapping: let (X, d) be a Banach space equipped with a distance function d. There exists a transformation, T such that $T : X \rightarrow X$ is a contraction, if there is a guaranteed $q < 1, q$ may be zero which squashes the distance between successive transformations (maps). In other words, $d(T(x), T(y)) < qd(x, y) \quad \forall x, y \in X$. q is called Lipschitz constant and determines speed of convergence of iterative methods.
– Fixed point: a fixed point x_* of a function $f : \mathbb{R} \rightarrow \mathbb{R}$ is a value that is unchanged by repeated applications of the function, i.e. $f(x_*) = x_*$. Banach space endowed with a contraction mapping admits of a unique fixed point. Note, for any

transformation on Turing Machines, there will always exist Turing Machines unchanged by the transformation.

- Stability: consider a continuously differentiable function $f : \mathbb{R} \to \mathbb{R}$ with a fixed point x_*, $f(x_*) = x_*$. The fixed point x_* is defined to be *stable* if $f'(x_*) < 1$. If $f'(x_*) > 1$, then x_* is defined as *unstable*.
- First Return Map: consider an iterative map on the set S, $f : S \to S$. Starting from an initial value $x_0 \in S$, we can iterate the map f to yield a trajectory: $x_1 = f(x_0)$, $x_2 = f(x_1)$ and so on. The first-return map is a plot of x_{n+1} vs. x_n for integer $n > 0$.
- Contour Plot: a contour plot is a 2-D representation of a 3-D surface. An alternative to surface plot, it provides valuable insights to changes in y as input variables $x\&1 - x$ change.

This section defines SBAF [23,55], computes its derivative and focuses on estimation of parameters used in the function. We compute the derivative of SBAF for two reasons: to use the derivative in back-propagation and to show that SBAF possesses optima instead of saddle point (note, sigmoid has saddle point). SBAF, defined with parameters k, α (these will be estimated in subsequent sections and no effort is spent on tuning these parameters while training) and input x (Data in the input layer), produces output y as follows:

$$\begin{aligned}
y &= \frac{1}{1 + kx^{\alpha}(1-x)^{1-\alpha}};\\
\Rightarrow \ln y &= \ln 1 - \ln(1 + kx^{\alpha}(1-x)^{1-\alpha})\\
&= -\ln(1 + kx^{\alpha}(1-x)^{1-\alpha})\\
\Rightarrow \frac{1}{y}\frac{dy}{dx} &= -\frac{1}{(1 + kx^{\alpha}(1-x)^{1-\alpha})} \cdot \left[k\alpha x^{\alpha-1}(1-x)^{1-\alpha} - kx^{\alpha}(1-\alpha)(1-x)^{1-\alpha-1}\right]\\
&= -\frac{k}{(1 + kx^{\alpha}(1-x)^{1-\alpha})} \cdot \left[\alpha x^{\alpha-1}(1-x)^{1-\alpha} - (1-\alpha)x^{\alpha}(1-x)^{-\alpha}\right]\\
\Rightarrow \frac{dy}{dx} &= -y^2\left[\frac{\alpha}{x} - (1-\alpha)\frac{1}{1-x}\right]kx^{\alpha}(1-x)^{1-\alpha}\\
&= -y^2\left[\frac{\alpha(1-x) - (1-\alpha)x}{x(1-x)}\right]kx^{\alpha}(1-x)^{1-\alpha}\\
&= -y^2\left[\frac{\alpha - x}{x(1-x)}\right]kx^{\alpha}(1-x)^{1-\alpha}. \qquad (4)
\end{aligned}$$

From the definition of the function, we have:

$$\begin{aligned}
y &= \frac{1}{1 + kx^{\alpha}(1-x)^{1-\alpha}}\\
\Rightarrow kx^{\alpha}(1-x)^{1-\alpha} &= \frac{1-y}{y}.
\end{aligned} \qquad (5)$$

Substituting equation (5) in (4),

$$\begin{aligned}
\frac{dy}{dx} &= -y^2 \cdot \frac{\alpha - x}{x(1-x)} \cdot \frac{1-y}{y}\\
&= \frac{y(1-y)}{x(1-x)} \cdot (x - \alpha).
\end{aligned} \qquad (6)$$

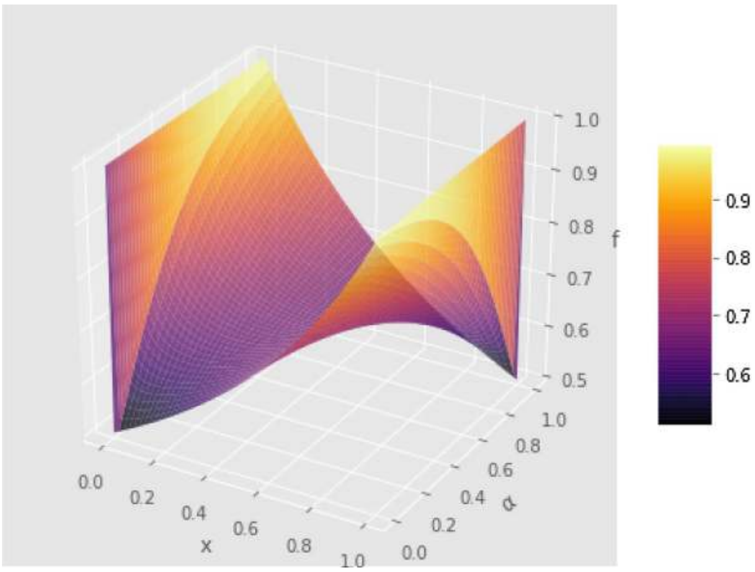


Fig. 5. Surface Plot of SBAF: The x-axis shows x, and the y axis shows α, both varied from 0 to 1. k is fixed to 1.

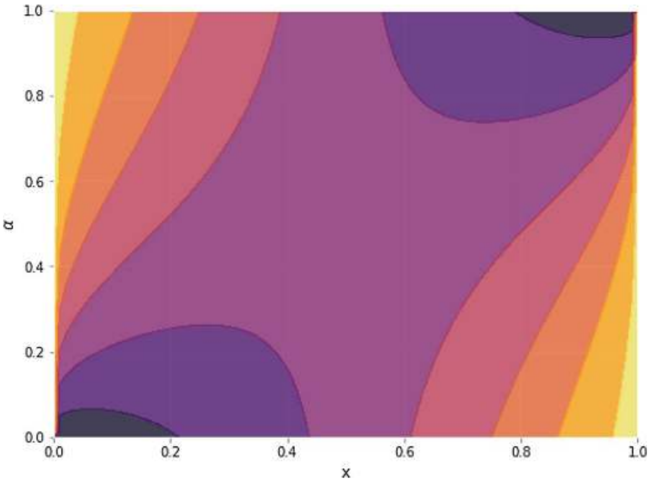


Fig. 6. Contour plot for SBAF, with x and α varied from 0 to 1. k is fixed to 1.

Figures 5 and 6 show various plots of the activation function. Figure 5 shows a surface plot by varying x and α. Figure 6 shows the contour plot. In both plots, we fixed $k = 1$. Note that the contour plot reveals a symmetry about $x = \alpha$. Further, as discussed in Section 8, the approximation of other activation functions using SBAF can be quite easily seen in the contour plot, by looking at the horizontal lines corresponding to $\alpha = 0$ and $\alpha = 1$.

We note, the motivation of SBAF is derived from using $kx^{\alpha}(1-x)^{1-\alpha}$ as discriminating function to optimize the width of the two separating hyperplanes in an SVM-like formulation. This is equivalent to the CDHS formulation when CD-HPF [12] is written as $y = kx^{\alpha}(1-x)^{\beta}$ where $\alpha + \beta = 1, 0 \le \alpha \le 1, 0 \le \beta \le 1$, and k is suitably assumed to be 1 (CRS condition). The representation ensures global maxima (maximum width of the separating hyperplanes) under such constraints [12]. Please also note, when $\beta = 1 - \alpha$, CRS condition [25] is satisfied automatically and we obtain the form of the activation, SBAF.

Our activation function has an optima. From the visualization of the function below, we observe less flattening of the function, tackling the "vanishing gradient" problem. Moreover, since $0 \leq \alpha \leq 1, 0 \leq x \leq 1, 0 \leq 1 - x \leq 1$, we can approximate, $kx^{\alpha}(1-x)^{1-\alpha}$ to a first order polynomial. This helps us circumvent expensive floating point operations without compromising the precision.

5.1 Existence of optima: Second order differentiation of SBAF for neural network

From equation (6),

$$\frac{dy}{dx} = \frac{y(1-y)}{x(1-x)} \cdot (x - \alpha).$$

It is easy to see,

$$\begin{aligned}\Rightarrow \frac{d^2y}{dx^2} &= -\frac{x(1-x)(\alpha - x)\cdot(1-2y)\cdot\left[\frac{y(1-y)}{x(1-x)}\cdot(\alpha - x)\right] + x(1-x)\cdot y(y-1) + y(y-1)\cdot(\alpha - x)\cdot(1-2x)}{(x(1-x))^2}\\ &= -\frac{y(y-1)[x(1-x) + (\alpha - x)\cdot(1-2x) + (\alpha - x)^2\cdot(1-2y)]}{(x(1-x))^2}\end{aligned}$$

when $\alpha = x$,

$$\begin{aligned}\Rightarrow \frac{d^2y}{dx^2} &= -\frac{x(1-x)\cdot y(y-1)}{(x(1-x))^2}\\ &= \frac{y(1-y)}{x(1-x)}.\end{aligned}$$

Clearly, the first derivative vanishes when $\alpha = x$, and the derivative is positive when $\alpha < x$ and is negative when $\alpha > x$ (implying range of values for α so that the function becomes increasing or decreasing). We need to determine the sign of the second derivative when $\alpha = x$ to ascertain the condition of optima (corresponding to minima of gradient descent training ensuring optimal discrimination between habitability classes). Assuming $0 < x < 1$, the condition of optimality, $0 \leq \alpha \leq 1$, y by construction lies between $(0, 1)$. Hence, $\frac{d^2y}{dx^2} > 0$ ensuring optima of y (Fig. 7).

5.2 Saddle points of sigmoid activation and a comparative note with SBAF

We note briefly that as opposed to the sigmoid activation function, SBAF has local minima and maxima. For example, for $k = 0.91, \alpha = 0.5$, a local minima occurs at $x = \alpha$. On the other hand, the sigmoid function has neither local minima nor maxima; it is easy to show that it only has a saddle point at $x = 0$: SBAF, however, has a critical point at $x = \alpha$.

Proof. Note that if $f(x)$ denotes the sigmoid function, then $f'(x) = f(x)(1 - f(x)); f''(x) = f(x)(1-f(x))(1-2f(x))$. Then, $f'(x) = 0$ implies that $f(x) \in \{0, 1\}$. However, for both these values, the second derivative is 0. $\square$

5.3 Visual analysis of SBAF

Figures 8–11 show different plots for SBAF by varying k and α.

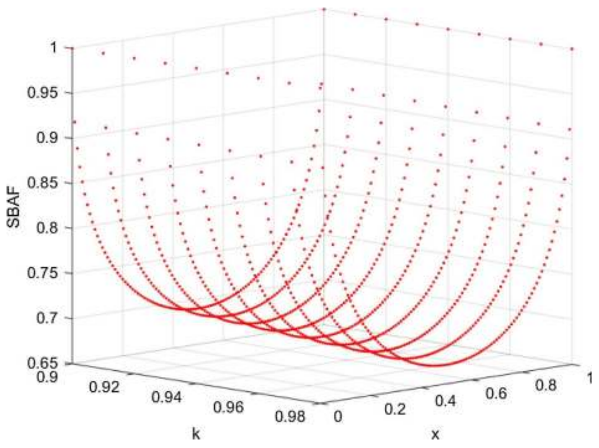


Fig. 7. Plot of SBAF function: it shows a minima for all values of $k(0.90-0.98)$ at $x=0.5$. Empirical test for stability confirms stable fixed point at $k=0.98, \alpha=0.5$ (Note, mimima of SBAF is obtained at $x=\alpha$ and thus we set $\alpha=0.5$). These are the k, α values we used in SBAF to train the classifier.

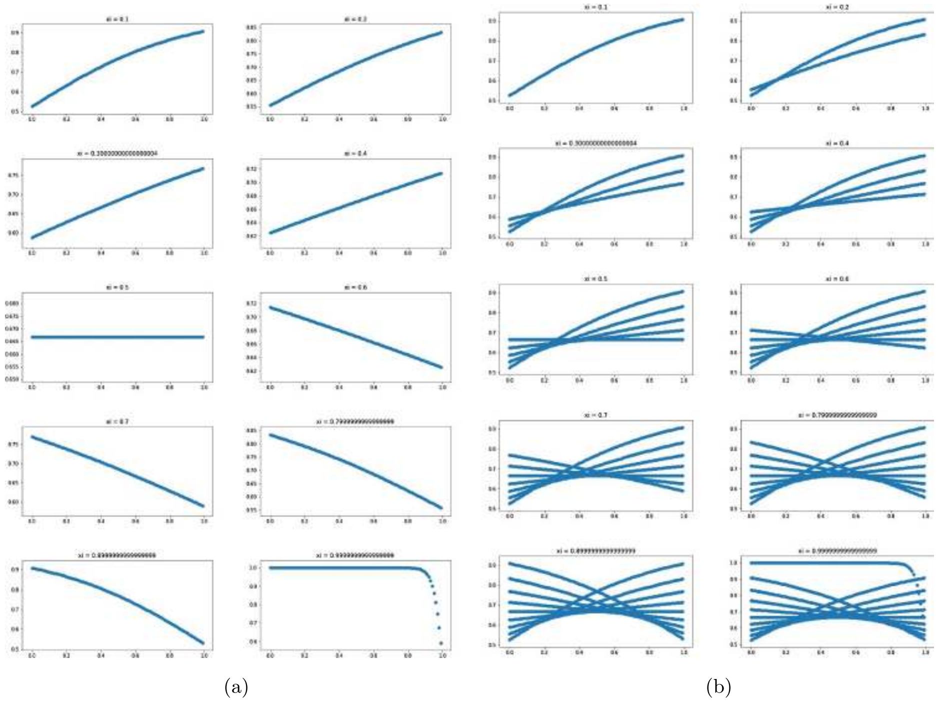


Fig. 8. Behavior of SBAF under different values of input, x. (a) Visualization of the activation: local oscillation seems to be absent. (b) Visualization of the activation: local oscillation seems to be absent.

5.4 Monotonicity of SBAF

It has not escaped our notice that SBAF is a non-monotonic function, if analyzed over the entire domain. We argue that activation functions in neural networks do not need to be monotonically increasing, or more generally, monotone at all. As an example,

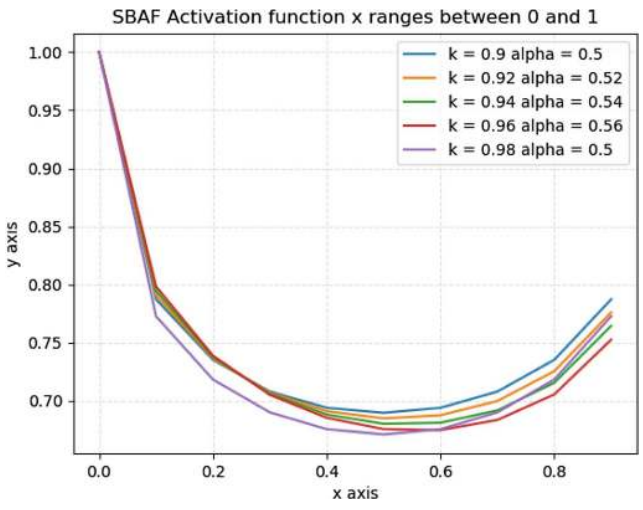


Fig. 9. 2-D plot of SBAF, α, K are varied in the admissible ranges for minima and fixed point of the SBAF (we establish later that SBAF has a fixed point for $k > 0.9$). We observe monotone behavior of SBAF.

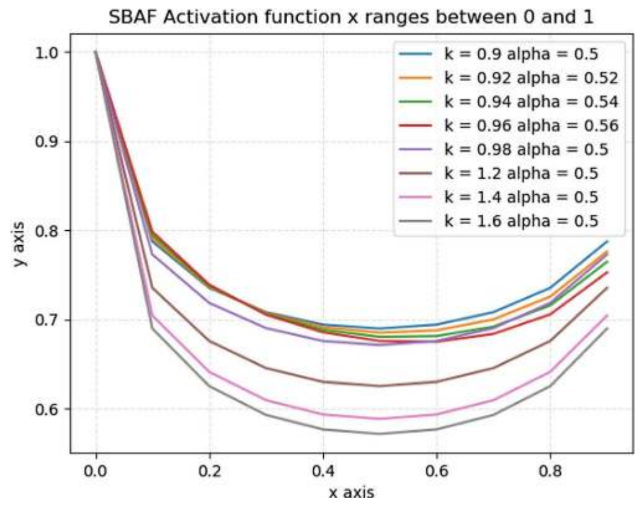


Fig. 10. 2-D plot of SBAF, α, is kept in the neighborhood of 0.5 for local minima and k is varied in the admissible ranges (0.91 for stable fixed point of SBAF). We observe non-monotone behavior of SBAF.

consider the simple parabola, $f(x) = x^2$, a clearly non-monotonic function. Indeed, for $\alpha = 1$, the Taylor series of SBAF is indeed parabolic (for reference, this expansion is $f(x) = 1-kx+k^2x^2+\mathcal{O}(x^3)$). If we were to coerce this into a monotonic function, we might consider using $f(x) = -x^2$ for $x < 0$. Our intuition is that if a neural network does not "like" the non-monotonic behavior of the standard parabolic activation function, it may simply set the corresponding weight (for two specific neurons in adjacent layers) to the opposite sign of if we had used the "coerced" version instead. Additionally, we need to observe that SBAF satisfies Cybenko Approximation and the non-monotonicity of SBAF has no effect on Cybenko (see Sect. 5.6).

To demonstrate this without going off-topic, we train a convolutional neural network with only parabolic activations (except for the last layer, where we use softmax)

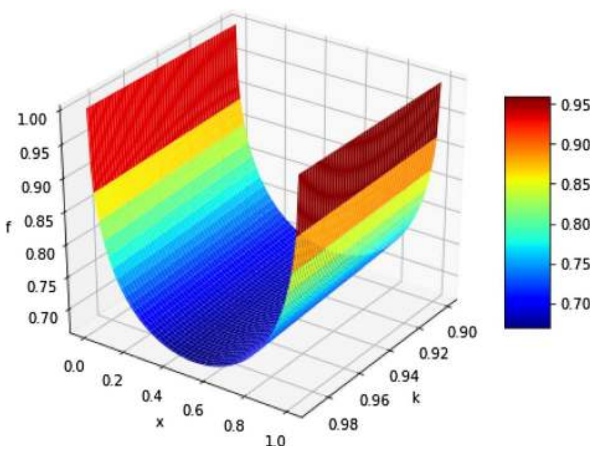


Fig. 11. Plot of $f = \frac{1}{1+kx^{\alpha}(1-x)^{1-\alpha}}$, SBAF, α is kept constant ($\alpha = x = 0.5$) and k is varied in the admissible range.

on the MNIST dataset for 20 epochs. For simplicity, we optimize the cross-entropy loss using SGD with a constant learning rate of 0.1. To eliminate the possibility of a chance result, we run the model 20 times. We observe a median accuracy of 96.24% and an inter-quartile range of 3.29 (25th percentile 93.98%, 75th percentile 97.27%). Clearly, the network did not have trouble performing well with this activation.

Briefly, the CNN we used had 5 Conv – BatchNorm – Parabola blocks. In the second and fourth blocks, we add a max pooling layer after the convolution. In the fifth block, we flatten the results of the Conv layer and then add a linear layer before batch normalization. All convolution layers use a kernel size of 3, and we use 32 filters in the first two convolutions, 64 in the next two, and 128 in the last one. We use valid padding in the first two convolutions, and same padding in the others.

As further proof of the efficacy of our proposed activation function, prior work by Saha and his group, Makhija et al. [56] uses a special case of our activation function in the discriminator network of a Generative Adversarial Network (GAN) to achieve exemplary performance in separating stars from quasars, outperforming the same architecture with the sigmoid activation. We defer to Makhija et al. [56] for a more detailed explanation of the method. The same activation was also used on the Gene Expression Omnibus (GEO) dataset by Sridhar et al. [57], and was found to outperform both the sigmoid and ReLU activations. These empirical evidence aside, we state that monotonicity is non-essential for Cybenko Universal Approximation (see Sect. 5.6). We revisit this conjecture in Section 8 at the time of introducing A-RELU, a variant of SBAF.

5.5 Exploring the vanishing gradient problem for SBAF

The vanishing gradient problem occurs during the phase when a neural network is trained. On close examination of back propagation, we see that the error gradient consists of derivative of activation function used in the network (along with the the other components). Hence to explore the vanishing gradient problem, one needs to investigate the behaviour of derivative of the activation function. Its important to note that during back propagation, the derivative keeps getting multiplied with itself (due to the chain rule used in the computation of gradient), making the gradient smaller and smaller. As we move all the way till input layer, the gradient vanishes off

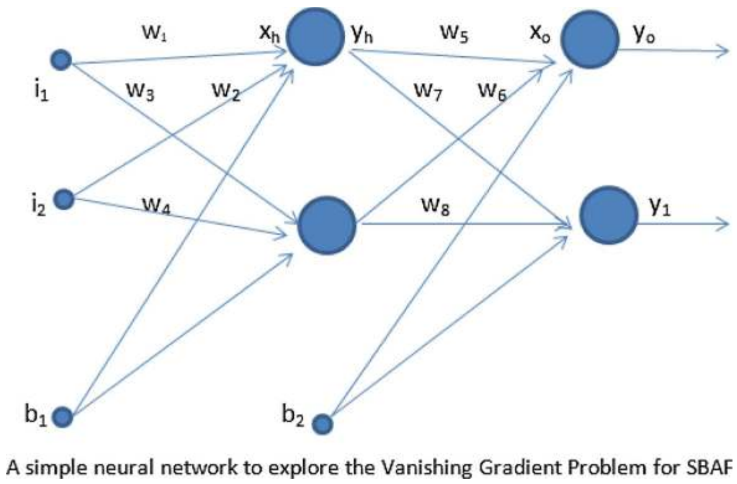

W1
Xh
Yh
W5
Xo
Yo
i1
W3
W2
W7
W6
i2
W4
W8
Y1
b1
b2
A simple neural network to explore the Vanishing Gradient Problem for SBAF

Fig. 12. Sample neural network.

completely from the network resulting in a state where the weights and biases stop getting updated. If we inspect the derivative (dy/dx) values for sigmoid, its value is small, for large values of x (x is the net input to neuron). Consequently, in deep learning models with fairly large number of hidden layers, a small derivative may lead to a state where the network may not train at all and result in non-convergence. To investigate whether SBAF suffers from this problem, let's use a simple 2-layered neural network (Fig. 12). We assume that the forward pass for one training sample is completed and we begin with the computation of error gradient ($\frac{\partial E_T}{\partial w_5}$) which is propagated back to the network. The error gradient with respect to weight w_5 can be shown as (for details, refer equations of Appendix C for computing the error gradient for output layer).

$$\frac{\partial E_T}{\partial w_5} = \frac{\partial E_T}{\partial y_o} \cdot \frac{\partial y_o}{\partial x_o} \cdot \frac{\partial x_o}{\partial w_5}.$$

Performing a close inspection of each derivative term, $\frac{\partial E_T}{\partial y_o}$ and $\frac{\partial x_o}{\partial w_5}$ holds a value between 0 and 1. This can be visualized from the derivation in Appendix C.3, $\frac{\partial E_T}{\partial y_o}$ is same as $-(y_{\text{target}} - y_o)$. Likewise, $\frac{\partial x_o}{\partial w_5}$ is same as y_h and the value of neuron output with SBAF as activation function, is between 0 and 1 (refer to Fig. 10). Figure 13 demonstrates a plot between $\frac{dy}{dx}$ and x, and the gradient is never zero even for values of x close to 1. Combining the three terms, one can make out that the derivative can reach a small value but never zero.

If we go backwards one layer further, and try to compute gradient for hidden layers as well, (refer Appendix B.2) we see:

$$\frac{\partial E_{\text{Total}}}{\partial w_1} = \frac{\partial E_{\text{Total}}}{\partial y_o} \cdot \frac{\partial y_o}{\partial x_o} \cdot \frac{\partial x_o}{\partial y_h} \cdot \frac{\partial y_h}{\partial x_h} \cdot \frac{\partial x_h}{\partial w_1}. \tag{7}$$

We know $\frac{\partial E_{\text{Total}}}{\partial y_o} \cdot \frac{\partial y_o}{\partial x_o}$ is between 0 and 1. The value for $\frac{\partial x_o}{\partial y_h}$ is same as w_5 which is not zero at any time during the training. We have already seen that derivative of SBAF represented by $\frac{\partial y_h}{\partial x_h}$ can become small for high values of x, but never touches zero. Lastly, $\frac{\partial x_h}{\partial w_1}$ is the input i_1 which is a normalized input feature value. Hence combing all the five terms we can conclude that the gradient never becomes zero even at the hidden layers of the network.

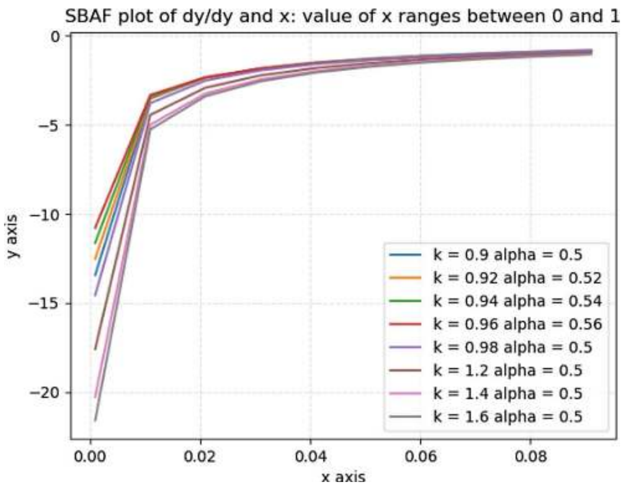


Fig. 13. Plot of dy/dx vs. x.

5.6 Universal approximation theorem for SBAF

It is well known that a feed-forward network with a single hidden layer containing a finite number of neurons satisfies the universal approximation theorem. This is important since it guarantees that simple neural networks can represent a wide variety of interesting linear/non-linear functions (with appropriately chosen parameters). Cybenko [21] proved one of the first versions of this theorem for sigmoid activation functions. We show that SBAF is also sigmoidal and hence satisfies the Universal Approximation Theorem.

Following Cybenko [21], we shall define the following:

- I_n: n-dimensional unit cube, $[0,1]^n$.
- $C(I_n)$: space of continuous functions defined on I_n.
- $M(I_n)$: the space of finite, signed regular Borel measures on I_n.
- $\sigma(t)$: a univariate function is defined as being *sigmoidal* if

$$\begin{aligned}\sigma(t) &\to 1 \text{ as } t \to +\infty,\\ \sigma(t) &\to 0 \text{ as } t \to -\infty.\end{aligned}$$

- σ is defined to be *discriminatory* if for a measure $\mu \in M(I_n)$,

$$\int_{I_n} \sigma(y^T x + \theta) \quad d\mu(x) = 0$$

$\forall y \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$ implies that $\mu = 0$.

Approximation Theorem [21]. *Let σ be any continuous discriminatory function. Then finite sums of the form*

$$G(x) = \sum_{j=1}^{N} \alpha_i \sigma\left(y_j^T x + \theta_j\right)$$

are dense in $C(I_n)$. In other words, given any $f \in C(I_n)$ and $\epsilon > 0$, there is a sum, $G(x)$, of the above form, for which

$$|G(x) - f(x)| < \epsilon \;\; \forall \;\; x \in I_n.$$

Proof. Please refer to Cybenko [21]. □

Also, it should be noted that by Lemma 1 of [21], *any continuous sigmoidal function is discriminatory.* We can thus prove:

Universal Approximation Theorem for SBAF. *The proposed function* $\mathrm{SBAF}_{k,\alpha}(x)$ *satisfies the universal approximation theorem.*

Proof. We observe that $\mathrm{SBAF}_{k,\alpha}(x)$ (for the choices $k = 1$ and $\alpha = 0$, see Sect. 8) is a continuous sigmoidal function and hence discriminatory. All the conditions of the approximation theorem are met. □

Thus, SBAF can be used with a feed-forward network with a single hidden layer containing a finite number of neurons to approximate a wide variety of linear and non-linear functions.

5.7 SBAF is not a probability density function [58]

Proof. Let us compute the integral below: $\int_I \frac{1}{1+kx^\alpha(1-x)^{1-\alpha}}$, where I is the interval containing $(-\infty,+\infty)$. We know from earlier calculations, that

$$y = \frac{1}{1 + kx^\alpha(1-x)^{1-\alpha}}, \tag{8}$$

and

$$\frac{dy}{dx} = \frac{y(1-y)}{x(1-x)} \cdot (x - \alpha). \tag{9}$$

Using the above in the integral and readjusting the limits of integration, we observe that $\int_{0,0} \frac{x(1-x)}{y(1-y)\cdot(x-\alpha)} dy \to 0$. Note, $0 \le f(x) \le 1, \forall x \in (0,1)$. Therefore the integral of $f(x)$ between 0 and 1 would also be less than or equal to 1. Equality will hold iff $f(x) = 1$, but this is not possible for any $x \in (0,1)$. □

SBAF is sampled from a probability density function (PDF). SBAF is not a PDF and we infer that it may oscillate. The next section contains a more pertinent insight, in clear agreement with the analysis, relating SBAF to regression under uncertainty.

6 Relating SBAF to binary logistic regression: Regression under uncertainty

Binary logistic regression builds a model to characterize the probability of Y_i (observed value of the binary response variable Y) given the values of the explanatory variable x_i in the following manner:

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \alpha_0 + \alpha_1 x_i, \tag{10}$$

where $\pi_i = P(Y_i = 1|x_i = x)$ denotes the probability that the response variable $Y_i = 1$ given the value $x_i = x$, and i stands for the index of the observed sample. The LHS of equation (10) is known as *logit*$(\cdot)$ or *log-odds.* The above formulation leads to the celebrated sigmoid activation function:

$$\pi_i = \frac{\exp(\alpha_0 + \alpha_1 x_i)}{1 + \exp(\alpha_0 + \alpha_1 x_i)}, \tag{11}$$

where the regression co-efficients α_0 and α_1 are to be determined.

Instead, if we formulate the logit as:

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = -\alpha\log(x_i) - (1-\alpha)\log(1-x_i) - \log(k), \tag{12}$$

where $0 < x_i < 1$, constants $k > 0$ and $0 \leq \alpha \leq 1$, then it leads to our proposed activation function:

$$\pi_i = \frac{1}{1 + kx_i^\alpha(1-x_i)^{1-\alpha}}. \tag{13}$$

Equation (12) can be understood as follows. Firstly, think of $\{x_i\}$ as probabilities of the observed explanatory variable X instead of the value itself. If X is a binary discrete random variable, then the quantities $-\log(x_i)$ and $-\log(1-x_i)$ would be the *self-information* of these two outcomes (measured in bits if the base of the logarithm is 2). The convex combination of these two self-information quantities is like an *average information quantity (Shannon entropy)*. In fact, the RHS is upper bounded by Shannon entropy (if $\alpha = x_i$ then RHS is the entropy of $\{x_i\}$).

Proof. Minimizing/Maximizing (13) is equivalent to maximizing/minimizing its inverse. We ignore the constant term 1, and assume that k is fixed. Then, the minima occurs where the derivative is 0:

$$\begin{aligned}\frac{d}{dx}\left(kx^\alpha(1-x)^{1-\alpha}\right) &= k\alpha x^{\alpha-1}(1-x)^{1-\alpha} - k(1-\alpha)x^\alpha(1-x)^{-\alpha} \\ &= kx^{\alpha-1}(1-x)^{-\alpha}\left(\alpha(1-x) - (1-\alpha)x\right) \\ &= kx^{\alpha-1}(1-x)^{-\alpha}(\alpha - x).\end{aligned}$$

Clearly, the optima of this, and therefore the optima of (13), occurs when $\alpha = x_i$. This is the critical point found by visual and theoretical analysis earlier. $\square$

The quantity $-\log(k)$ can be thought of as a bias term or the information content that is universally available (if we think of the RHS as average uncertainty instead of average information, then this quantity would be the irreducible uncertainty that is present in the environment, such as noise). Thus, under this scenario, the log-odds of classifying the binary response variable Y as 1 is a function of the average self-information of the observed explanatory variable.

This means that if the probability of observed explanatory variable was very close to zero, then x_i would be close to zero and the RHS would also be close to zero giving the log-odds ratio (LHS) also close to zero. If the probability of the observed explanatory variable was close to 0.5, then RHS would be very high and the log-odds ratio (RHS) would be close to 1. Now, as the probability increases towards 1, the RHS starts to reduce and log-odds ratio (LHS) also starts to reduce. This means that we trust the observed variable only if it has probability between 0 and 0.5. We do not trust high values (above 0.5).

Another way to interpret this is that as the uncertainty increases, then the neuron fires (activates). The neuron in this case continuously increases the magnitude of its response as the uncertainty increases. The term $-\log(k)$ is the ambient noise term, and the neuron can fire if this is high enough (greater than some preset threshold T). The task of regression here is to determine α and k such that it models the binary response variable Y as a function of the uncertainty of the observed variables (along with noise in the environment).

7 Functional properties of SBAF

7.1 Existence of a fixed point

Let us consider the first order differential equation [11]

$$\frac{dy}{dx} = \frac{y(1-y)}{x(1-x)} \cdot (x-\alpha) \tag{14}$$

which is a representation of the standard form:

$$\frac{dy}{dx} = f(x,y). \tag{15}$$

In order to prove the existence of a fixed point, we need to show that $f(x,y)$ is a Lipschitz continuous function. On differentiating y w.r.t x we obtain y as:

$$f(x,y) = \frac{dy}{dx} = \frac{y(1-y)}{x(1-x)} * (x-\alpha). \tag{16}$$

When $0 < x < 1$, we observe that $y < 1$. Moreover, when $x \to 0$, $y \to 1$ and when $x \to 1$, $y \to 1$. In our case, f is a continuous and differentiable function over the interval [0,1]. This implies f is bounded in (0,1). This further implies that the function follows the mean value theorem. That is, for some $c \in (0,1)$,

$$f'(c) = \frac{f(b)-f(a)}{b-a}.$$

On differentiating f w.r.t x we obtain the following equation

$$f'(x,y) = \frac{dy}{dx} = \frac{y(1-y)}{x(1-x)}. \tag{17}$$

Since we already know the representation of y in terms of x from (13), we can substitute the value of y and we obtain the following representation for f:

$$f'(x) = \frac{kx^{\alpha}(1-x)^{(1-\alpha)}}{(1+kx^{\alpha}(1-x)^{(1-\alpha)})^2} * \frac{1}{x(1-x)}.$$

Clearly, the right hand side is bounded between $(0,1)$ by some positive constant K. Using the above constraint in the mean value theorem specified, we obtain the following:

$$\begin{aligned} f'(c) = \frac{f(b)-f(a)}{-a} &<= K \\ f(b)-f(a) &\leq K(b-a) \\ |f(b)-f(a)| &\leq K. \end{aligned}$$

Since $a=0$ and $b=1$, the above equation is of the form

$$|f(b)-f(a)| \leq K|(b-a)|.$$

Therefore, f is a Lipschitz continuous and by Picard's theorem, the differential equation [59] has a fixed point and a unique solution in the form of the activation function, to be used in neural networks for classification.

7.2 Lipschitz continuity, contraction map and unique fixed point

In this section, we show that the fixed point is unique in the interval, thus ensuring that the solution to the DE, SBAF, is unique in that interval. We exploit the Banach contraction mapping theorem to achieve this goal. We consider the following DE

$$f(x,y) = \frac{dy}{dx} = \frac{y(1-y)}{x(1-x)} \cdot (x-\alpha) \tag{18}$$
$$y(x_0) = y_0.$$

Assume $f(x,y)$ to be continuous on $D = (x_0-\delta, x_0+\delta), (y_0-b, y_0+b)$. Then $\exists$ a solution in D. This gives rise to the solution as the activation function, y. Furthermore, if $f(x,y)$ is Lipschitz continuous in D, or in a region smaller than D, the solution thus found is unique.

Proof. (a) We first prove that $f(x,y)$ is Lipschitz continuous. We write $y' = \frac{dy}{dx} = f(x,y) = \frac{y(1-y)}{x(1-x)} * (x-\alpha)$.
Now,

$$|f(x,y_1) - f(x,y_2)| = \left| \frac{x-\alpha}{x(1-x)} \left(y_1(1-y_1) - y_2(1-y_2)\right) \right|.$$

Within $(0,1)$, $\left|\frac{x-\alpha}{x(1-x)}\right|$ is bounded. The expression blows up at $x = 0, 1$. Therefore,

$$|f(x,y_1) - f(x,y_2)| \leq k\left|y_1 - y_1^2 - y_2 + y_2^2\right|$$
$$|f(x,y_1) - f(x,y_2)| \leq k|y_1 - y_2 - (y_1-y_2)(y_1+y_2)|$$
$$|f(x,y_1) - f(x,y_2)| \leq k|(y_1-y_2)(1-y_1-y_2)|.$$

By construction, y_1 and y_2 are bounded by some positive constant $\xi \in (0,1)$. Therefore, the sum is bounded by 2ξ and $|1-(y_1+y_2)| \leq 1 \Rightarrow |f(x,y_1) - f(x,y_2)| \leq k|y_1 - y_2|$.
This implies $f(x,y)$ is Lipschitz continuous in y. We may now proceed to establish the existence of a unique fixed point i.e. unique solution (activation function) to the differential equation above.

(b) Let T be a contraction mapping, i.e. $T : X \rightarrow X$, where X is a complete metric space. Then T has a unique fixed point in X [60]. Moreover, let $x_0 \in X$, we define x_k by setting an iterative map, $x_{k+1} = T(x_k)$. Let us fix $d_0 = d(x_0, x_1)$. Then, by Lipschitz continuity

$$d(x_k, x_{k+1}) = d(T(x_{k-1}), T(x_k)) \leq \alpha_1 d(x_{k-1}, x_k)$$
$$d(x_k, x_{k+1}) \leq \alpha_1 d(T(x_{k-2}), T(x_{k-1}))$$
$$d(x_k, x_{k+1}) \leq \alpha_1^2 d((x_{k-2}), x_{k-1}))$$
$$\vdots$$
$$d(x_k, x_{k+1}) \leq \alpha_1^k d(x_0, x_1) = \alpha_1^k d_0.$$

Clearly, x_k is a Cauchy sequence. Since X is a complete metric space, $x_k \rightarrow x \in X$. Thus,

$$d(T(x_k), T(x)) \leq \alpha d(x_k, x) \rightarrow 0.$$

Thus, $T(x_k) \to T(x)$. But $T(x_k) = x_{k+1}$. Therefore, $T(x) \to x$, i.e. $T(x) = x$. Suppose $T(y) = y$ for $y \neq x$.

$$\begin{aligned} d(x,y) = d(T(x),T(y)) &\leq \alpha d(x,y) \\ d(x,y) &\leq \alpha d(x,y). \end{aligned}$$

This is possible only if $d(x,y) = 0 \Rightarrow x = y$. This implies that the fixed point is unique. □

We use the following lemma to complete the missing piece. f is continuous and Lipschitz w.r.t y on the defined domain. Then, there exists a unique fixed point of T in X. The function y (activation function) is the unique solution. We establish the fact that the activation function is unique in the interval (0,1).

7.3 Computation of the fixed point

Now that the existence of of the fixed point is established, we proceed to find it in the following manner. Let us consider the representation of y given by (1). By definition of fixed point we have $T(x) = y(x) = x$ at the fixed point, i.e.

$$\begin{aligned} x = \frac{1}{1 + kx^{\alpha}(1-x)^{1-\alpha}} &= T(x) \\ x\left[1 + kx^{\alpha}(1-x)^{1-\alpha}\right] &= 1 \\ x + kx^{\alpha+1}(1-x)^{1-\alpha} &= 1 \\ kx^{\alpha+1}(1-x)^{1-\alpha} &= 1 - x \\ kx^{\alpha+1}(1-x)^{-\alpha} &= 1. \qquad (19) \end{aligned}$$

Assume $k = 1$. Hence, (19) becomes

$$x^{\alpha+1}(1-x)^{-\alpha} = 1.$$

On applying log on both sides we obtain

$$\begin{aligned} \log\left(x^{\alpha+1}(1-x)^{-\alpha}\right) &= 0 \\ \log x^{\alpha+1} + \log(1-x)^{-\alpha} &= 0 \\ (\alpha+1)\log x - \alpha\log(1-x) &= 0 \\ \alpha\log x + \log x - \alpha\log(1-x) &= 0 \\ \alpha(\log x - \log(1-x)) + \log x &= 0 \\ \log\left(\frac{1-x}{x}\right)^{\alpha} &= \log x. \end{aligned}$$

Thus, $\alpha = \frac{\log x}{\log \frac{1-x}{x}}$.

7.4 Visual analysis of the fixed point

We use the formula from the above computation and visually represent the activation function, SBAF, the solution to the differential equation mentioned above. We observe that, for $K = 0.9$ onward, we obtain a stable fixed point. SBAF used for

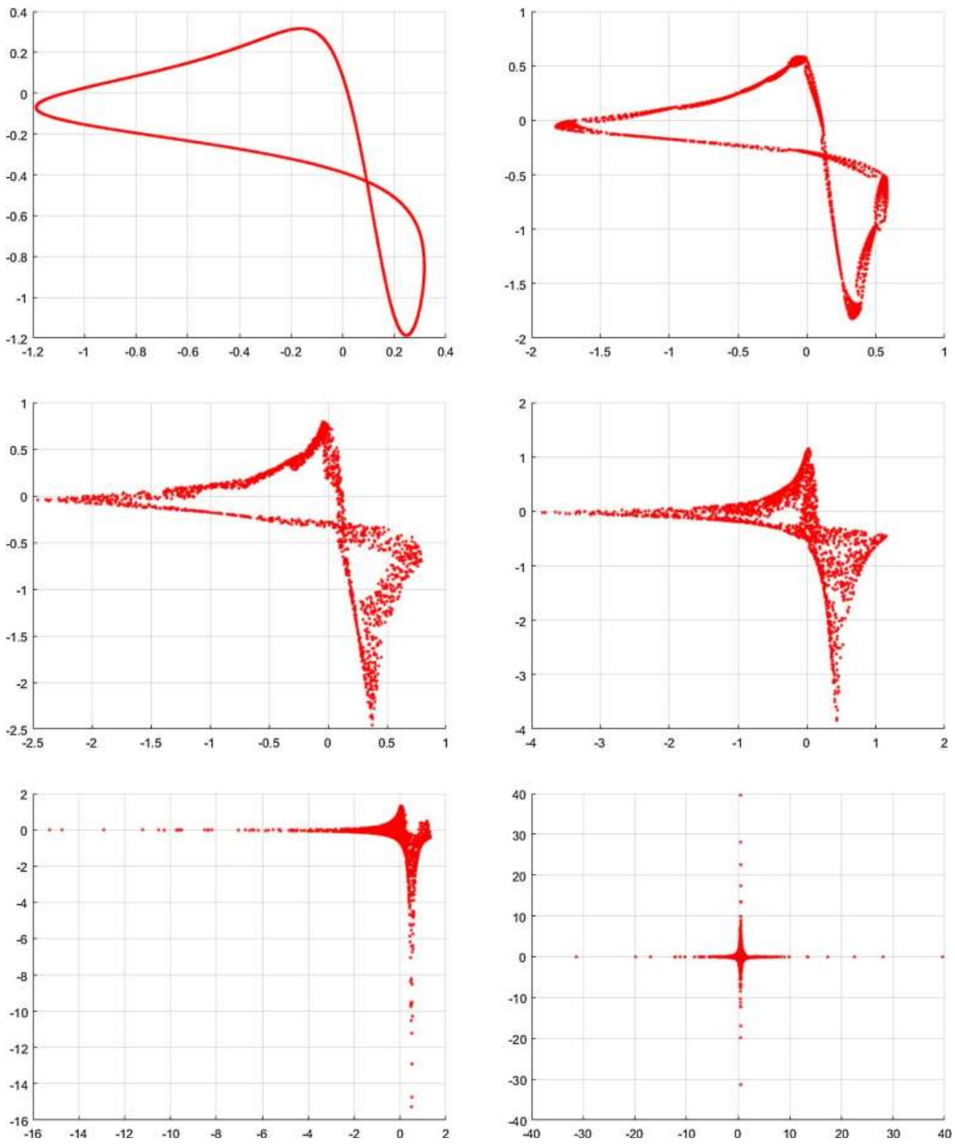


Fig. 14. First return maps of SBAF for $\alpha = 0.5$ and various values of $k < 0$ (top to bottom, left to right): $\{-2.0, -1.97, -1.90, -1.75, -1.5, -1.0\}$ indicating absence of a fixed point. The plot confirms absence of fixed points for all values of $k < 0$.

training the neural net is able to classify PHL-EC data with remarkable accuracy when K is very close to 1. Existence of a stable fixed point thus is a measure of classification efficacy, in this case.

We explore the non-linear dynamics of SBAF. At a fixed point, as noted above, we have:

$$kx^{\alpha+1}(1-x)^{-\alpha} = 1.$$

In the above expression for the fixed point, for all real values of α and when $0 < x < 1$, the left-hand side is always negative if $k < 0$. Thus, there cannot be any fixed point for $k < 0$ when $0 < x < 1$. When $k > 0$, it is possible to have a fixed point x^*. Below

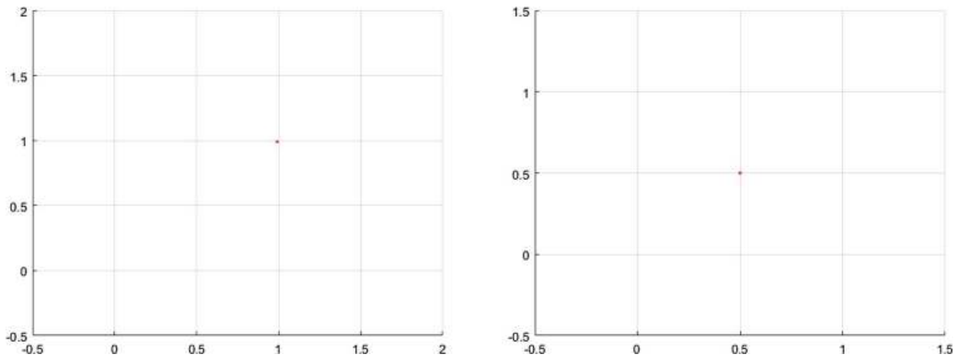


Fig. 15. First return maps of SBAF for $\alpha = 0.5$ and various values of $k > 0$: $\{0.1, 2.0\}$ indicating presence of a fixed point.

we plot the first return maps for the SBAF for a few cases with fixed $\alpha = 0.5$ and for different values of k (Figs. 14 and 15). We plot only the real values of x (since we have complex dynamics as x can become a complex number).

We can thus explicitly compute the values of k for which there exists a stable fixed point when $\alpha = 0.5$:

$$kx^{*(0.5+1)}(1 - x^*)^{-0.5} = 1,$$

which implies:

$$k = \frac{\sqrt{(1 - x^*)}}{x^* \sqrt{x^*}},$$

where we seek the fixed point x^* such that $0 < x^* < 1$, $x^* = \mathrm{SBAF}(x^*)$, and we also require for stability: $|\mathrm{SBAF}'(x^*)| < 1$. There are an infinite number of stable fixed points satisfying these conditions. Numerically, we found that x^* can be any value in the range $0 < x^* < 1$, and correspondingly, k takes values: $\infty > k > 0$ (lower the x^*, higher the k). For example, if the stable fixed point is $x^* = \frac{1}{\sqrt{2}}$ then $k = 0.91018$. When the fixed point varies from 0 to 1, the value of $\mathrm{SBAF}'(x^*)$ varies from -0.5 to 0.5 (thus $|\mathrm{SBAF}'(x^*)| < 1$, making it a stable fixed point).

We find it significant to mention that SBAF is inspired from a production function in economics [61,62] even though the adaptation is non-trivial and significantly more complex in structure than the Cobb–Douglas production function, CDPF [63–66]. Since CDPF is a production function [67] defined by k, α, β as labor and capital inputs, a negative value of k implies no quantity is produced, rather borrowing from market is necessary. This is important since it validates our choice of k in neural net training from an econometric argument. This is consistent with our assertion that the choices of parameters for optimal classification performance are not accidental!

As noted above, stable fixed points exists for a range of values of k depending on the value of x^* and α. When $\alpha = 0.5$, we found that there is a stable fixed point for every $0 < k < \infty$. However, classification with SBAF works optimally at $k = 0.91$ (corresponding $x^* \approx \frac{1}{\sqrt{2}}$). This is again consistent with our hypothesis that a stable fixed point will facilitate classification while chaos might occur otherwise. We have proven from the first order ODE that a fixed point exists and computed the fixed point in terms of α by fixing $k = 1$ and verified the same via simulation. We have also observed (but not reported here) that altering k values minimally within the stable fixed point range does not alter classification performance greatly. This reaffirms the confidence in the range of k values obtained from fixed point analysis. To sum up, SBAF is the analytical solution to the first order DE and has a fixed point! This fixed point analysis is computationally verified and applied in the classification task on the PHl-EC data, via a judicious choice of parameters.

7.5 Proof of global maxima for the activation

We now go back to an economics perspective of our activation function (or production function). In this section, we prove the existence of an optimum solution.

Let us consider the activation function:

$$y = \frac{1}{1 + kx^{\alpha}(1-x)^{1-\alpha}} \tag{20}$$

$\alpha + \beta = 1$, where $\alpha > 0$ and $\beta > 0$.

Let $S = x$ and $I = (1 - x)$ and $1 - \alpha = \beta$. For the constrained case of the above equation a critical point is defined in terms of the Lagrangian multiplier method. Suppose the constraint is $g = w_1S + w_2I - m$. Let the Lagrangian function for the optimization problem be:

$$\begin{aligned} L &= \frac{1}{1 + kx^{\alpha}(1-x)^{1-\alpha}} - \lambda(w_1S + w_2I - m) \\ L &= \frac{1}{1 + kS^{\alpha}I^{\beta}} - \lambda(w_1S + w_2I - m). \end{aligned} \tag{21}$$

On partially differentiating equation (2) with respect to S, I and λ we obtain the following first order constraints

$$\begin{aligned} \frac{\partial L}{\partial S} &= -yk\alpha S^{\alpha-1}I^{\beta} - \lambda w_1 = 0 \\ \frac{\partial L}{\partial I} &= -yk\beta S^{\alpha}I^{\beta-1} - \lambda w_2 = 0 \\ \frac{\partial L}{\partial \lambda} &= -(w_1S + w_2I - m) = 0. \end{aligned}$$

On differentiate again we obtain the second order condition:

$$\begin{aligned} \frac{\partial^2 L}{\partial \lambda^2} &= 0 \\ \frac{\partial^2 L}{\partial S^2} &= -yk\alpha^2 S^{\alpha-2}I^{\beta} \\ \frac{\partial^2 L}{\partial I^2} &= -yk\beta^2 S^{\alpha}I^{\beta-2} \\ \frac{\partial^2 L}{\partial SI} &= -yk\alpha\beta S^{\alpha-1}I^{\beta-1} \\ \frac{\partial L}{\partial \lambda S} &= -w_1 \\ \frac{\partial L}{\partial \lambda I} &= -w_2. \end{aligned}$$

For equation (1) to obtain a optimum max value it subject to the assumed constraint, it must satisfy the condition $\|M\| > 0$, where M is the bordered hessian matrix.

$$M = \begin{bmatrix} 0 & -w_1 & -w_2 \\ -w_1 & -yk\alpha^2 S^{\alpha-2} I^{\beta} & -yk\alpha\beta S^{\alpha-1} I^{\beta-1} \\ -w_2 & -yk\alpha\beta S^{\alpha-1} I^{\beta-1} & -yk\beta^2 S^{\alpha} I^{\beta-2} \end{bmatrix}$$

$$\begin{aligned} \|M\| &= w_1^2 yk\beta^2 S^{\alpha} I^{\beta-2} + w_1 w_2 yk\alpha\beta S^{\alpha-1} I^{\beta-1} + w_2^2 yk\alpha^2 S^{\alpha-2} I^{\beta} \\ &\quad - w_1 w_2 yk\alpha\beta S^{\alpha-1} I^{\beta-1}, \end{aligned}$$

$\|M\| = w_1^2 yk\beta^2 S^{\alpha} I^{\beta-2} + w_2^2 yk\alpha^2 S^{\alpha-2} I^{\beta}$,

$\|M\| > 0 \rightarrow$ the production function has global optima under CRS.

8 Approximating other activations

In this section, we show the k, α values for which SBAF approximates other activation functions.

- $k = 1, \alpha = 0$; SBAF becomes $\frac{1}{2-x}$ which is what we obtain in sigmoid when we restrict the Taylor series expansion at 0 of $\exp(-x)$ to first order!
- $k = 1, \alpha = 1$; SBAF becomes $\frac{1}{1+x}$ which upon binomial expansion (restricting to first order expansion assuming $0 < x < 1$) yields $y = 1 - x = 1 - \text{ReLU}$.

As noted earlier, these approximations can be seen in Figure 6 along the top and bottom horizontal edges, which correspond to $\alpha = 1$ and $\alpha = 0$ respectively.

8.1 ReLU approximate in the positive half

Consider the function $f(x) = kx^n$. We know that the ReLU activation function is $y = \max(0, x)$. We need to approximate the values n and k such that $f(x)$ approximates to the ReLU activation function over a fixed interval.

$$\text{Relative error} = \frac{\|f(x) - y\|}{\|y\|}.$$

Let the minimum tolerable error be $\epsilon < 10^{-3}$

$$\begin{aligned} \frac{\|f(x) - y\|}{\|y\|} &\leq \epsilon < 10^{-3} \\ \|f(x) - y\| &\leq 10^{-3}\|y\| \\ \|f(x)\| &\leq \|y\|(10^{-3} + 1) \\ \|f(x)\| &\leq 1.001\|y\| \\ \|f(x)\| &\leq 1.001\|y\|. \end{aligned}$$

Since $f(x) = kx^n$ approximates the positive half (i.e. when $x > 0$) of the ReLU activation function, $y = \max(x, 0)$, the value of y when $x > 0$ can be written as:

$$\|y\| = \|x\|.$$

Using this value in the error calculation we rewrite the error approximation as,

$$\begin{aligned} \|kx^n\| &\leq 1.001\|x\| \\ \|kx^{n-1}\| &\leq 1.001. \end{aligned}$$

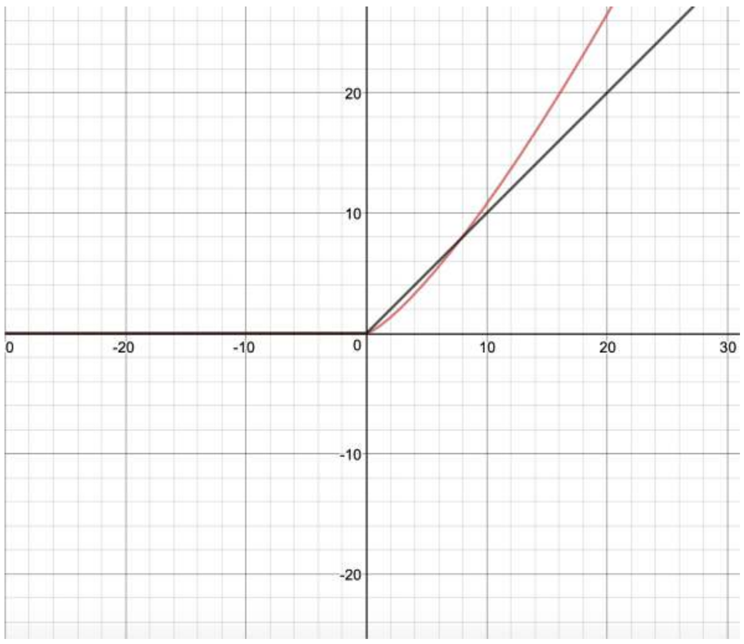


Fig. 16. ReLU approximation $y = kx^n$. In $[0, 10]$, the values $k = 0.54, n = 1.30$, correct to two decimal places, yield the least approximation error as detailed in the text. The values of k, n are used to train the network with A-ReLU. The two curves intersect at $(7.8, 7.8)$.

The above is an optimization problem i.e. min $\left\|kx^{n-1}\right\|$ subject to the constraints $k > 0, n > 1, -10 < x < 10$. We obtain the following bounds on k and n:

$$0 < k < 1; 1 < n < 2.$$

Therefore, we obtain the following continuous approximation of ReLU: $y = kx^n, x > 0$ when $0 < k < 1, 1 < n < 2, -10 < x < 10$, otherwise $y = 0$. More precisely, the approximation to the order of 10^{-3} is $k = 0.54, n = 1.3$ (Fig. 16).

8.2 Error approximation and estimation of k and n

Define

$$L_2 = \sum \|y_i - kx_i^n\|^2. \tag{22}$$

We need to minimize the least square error to approximate the function $f(x) = \|y - kx^n\|$ to the Relu activation function. This is a continuation in our efforts to find a continuous and differentiable approximation to ReLU.

Let us fix x between some fixed interval, say $1 < x < 10$. This choice is also justified by observing linear combinations of weights and inputs during the training where we observed the combination rarely surpassing $+3$ in the positive half plane. This means $x > 10$ would be least likely. Starting with some fixed value of k, say $k = 0.5$, we try to approximate the value of n such that the error in approximation is minimum. Algorithm 1 shows a simple way to achieve this.

Algorithm 1: Find k, n with minimum error.

```
1  min_error ← $\infty$;
2  min_k ← $-1$, min_n ← $-1$;
3  for $k \leftarrow 0.5$ to 1 by 0.01 do
4      for $n \leftarrow 1$ to 2 by 0.01 do
5          error ← 0;
6          for $x \leftarrow 1$ to 10 do
7              error ← sum $+(kx^n - x)^2$;
8          end
9          if error < min_error then
10             min_error ← error;
11             min_k ← $k$;
12             min_n ← $n$;
13         end
14     end
15 end
16 return $k, n$
```

$f(x)$ is minimum at $k = 0.53$ or $k = 0.54$ when $n = 1.3$. These are the parameter values used to train the network yielding better performance in classification, compared to ReLU.

8.3 Differentiability of $f(x)$

$$f(x) = \begin{cases} 0, & x \le 0 \\ kx^n, & x > 0. \end{cases}$$

We test for differentiability at $x = 0$. When $x \to 0^-$, $f'(x) = 0$. When $x \to 0^+$, $f'(x) = knx^{n-1} \to 0$. These derivatives are equal, and thus, the function is differentiable at $x = 0$. Moreover, the derivative is continuous.

Note: A-ReLU, $f(x) = kx^n$ gives rise to $f(x) = kx^2$. Please recall our discussion in Section 5.4 about non-monotonicity of SBAF and the family of activation functions SBAF generates. By inspection, we observe that when $n = 2$, A-ReLU generates a family of parabolic activation functions, with the degree of freedom on the same parameter k. Incidentally, this parabolic activation works well in practice for all the data sets we experimented in this manuscript, for the accepted values of k discussed in Section 7.4 and above.

8.4 Note about A-ReLU

On plotting the curve for the function, we obtain

$$f(x) = \begin{cases} 0, & x \le 0 \\ kx^n, & x > 0 \end{cases}$$

for different values of n and k, we have observed that the curve of $f(x)$ for $x > 0$ increases exponentially with increase in n. Since, the nature of the curve kx^n is non linear when $k \neq 0$, $k \neq 1$ and $n \neq 0$, $n \neq 1$ it is not meaningful to compute the absolute error for the entire range on the positive scale from 0 to ∞. Hence, we have fixed the range of x between 0 to ∞ on the positive scale and $-\infty$ to 0 on the negative scale. In Section 8.1, we have found the relative error is bounded by $\|x\|$.

In the chosen interval of $-\infty < x < 10$, we find that the minimum absolute error is $\approx 0.75 \pm 0.05$ when $k = 0.53$ or $k = 0.54$, and $n = 1.3$. This value is quite small compared to our chosen bound, 10. Hence, we can say that the function $f(x)$ is a good approximation to the ReLU activation function[3].

For a continuous domain, the squared error changes from a sum to an integral. We take this integral over $(0, t)$, for some t that we choose later. To minimize this, we take the partial derivatives with respect to k and n, and set them to 0. This yields two equations, with two unknowns. t can be thought of as the upper limit of the domain where the approximation is good.

$$
\begin{aligned}
f(k,n) &= \int_0^t (kx^n - x)^2 dx \\
&= \int_0^t \left(k^2x^{2n} + x^2 - 2kx^{n+1}\right) dx \\
&= k^2 \frac{t^{2n+1}}{2n+1} + \frac{t^3}{3} - \frac{2kt^{n+2}}{n+2} \\
\frac{\partial f}{\partial k} &= 2k\frac{t^{2n+1}}{2n+1} - \frac{2t^{n+2}}{n+2} = 0 \\
&\Rightarrow k\frac{t^{2n+1}}{2n+1} = \frac{t^{n+2}}{n+2} \\
&\Rightarrow \boxed{k = \left(\frac{2n+1}{n+2}\right) t^{1-n}} \\
\frac{\partial f}{\partial n} &= k^2 \left(\frac{2t^{2n+1}\ln(2n+1)}{2n+1} - \frac{2t^{2n+1}}{(2n+1)^2}\right) \\
&\quad - 2k\left(\frac{t^{n+2}\ln(n+2)}{n+2} - \frac{t^{n+2}}{(n+2)^2}\right) = 0 \\
&\Rightarrow \boxed{k\left(\frac{t^{2n+1}\ln(2n+1)}{2n+1} - \frac{t^{2n+1}}{(2n+1)^2}\right) = \frac{t^{n+2}\ln(n+2)}{n+2} - \frac{t^{n+2}}{(n+2)^2}.}
\end{aligned}
$$

Because it is difficult to a solution to this system of equations analytically, we simply plot these equations, fixing $t = 10$. In the graph below, the y-axis is k, and the x-axis is n. Note that the intersection of the two equations yields the trivial solution $k = n = 1$; however, this solution is not very interesting. Instead, we fix the value of k, and find the intersections with the two curves above. Figure 17 shows this plot.

8.5 Non-monotone activation revisited: Extension of 5.4

We argued in Section 5.4 about the non-necessity of strict monotonicity of activation functions. We mentioned that parabolic activations, which are not one-to-one functions, and therefore non-monotone can be derived from the generic class of activation functions proposed as SBAF family. Precisely, parabolic activation may be interpreted as an offshoot of A-RELU, in continuity of the discussion in this section. The theoretical discussion and results (Tab. 5) are now presented below. The general

[3] Please note, even if we do not restrict the function in the specified range mentioned above, we can work with the function itself as another activation function with no discontinuity at $x = 0$. We choose the value of n between 1 and 2 so that the derivative does not explode!

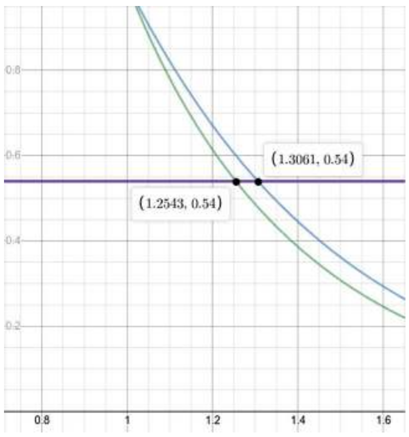


Fig. 17. Plot of the two curves derived analytically, showing the intersection with $k = 0.54$. The blue curve is the first one, and the green curve is the second.

parabolic function actually outperforms all of its counterparts and along with the A-ReLU unit, is one of the best performers.

8.5.1 Standard parabola

$$y = a * x^2 + b * x + c.$$

As mentioned in the Introduction above, we take a range of values for x and estimate values of a, b and c, for which the activation function will be the most effective. In out initial discussion, we discard c – in order to let the parabola pass through the origin and also to reduce the number of hyperparameters in the activation unit. Let us consider only a and b to tune for now (2 at a time), one particular result of optimal parameters is given below:

$$y = a * x^2 + b * x.$$

To find the optimal values of the 2 hyperparameters, we use the euclidean distance between our parabolic function and the standard ReLU function and minimize the same. Taking a range of input values as $(-t, t)$, the equation is as follows:

$$\min \int_0^t (a * x^2 + b * x - x)^2 dx + \int_{-t}^0 (a * x^2 + b * x)^2 dx.$$

For our experimentation, we take input values in the range $(-1, 1)$. Using this, we obtain the following range of values of hyperparameters a and b as $a = 0.60{-}0.65$, $b = 0.48{-}0.52$.

8.5.2 SBAF based parabola

The first analysis of a non-monotonic function includes a variation of the Saha Bora Activation Function:

$$y = \frac{1}{1 + k * x^{\alpha} * (1 - x)^{1-\alpha}}.$$

By using $\alpha = 1$ in the above equation, we get

$$f(x) = \frac{1}{1 + k * x} f(x) = (1 + k * x)^{-1}.$$

Now, take $k = 1$ and expanding the series, we get

$$\begin{aligned}(1+x)^{-1} &= 1 - x + x^2 - \dots \\ (1+x)^{-1} &= 1 - 2*(1/2)*x + x^2 \\ (1+x)^{-1} &= x^2 - 2*(1/2)*x + (1/2)^2 - (1/2)^2 + 1 \\ (1+x)^{-1}y &= (x - 1/2)^2 + 3/4.\end{aligned}$$

8.5.3 A-ReLU based parabola

$$f(x) = \begin{cases} k * x^{\mathrm{n}}, & \text{if } x \geq 0, \\ 0, & \text{if } x < 0. \end{cases}$$

Now here, we shall fix the value of n to 2 and mirror the function over the y-axis to obtain the following function:

$$f(x) = k * x^2.$$

Therefore, to sum up the previous two sections, the two parabolic functions considered along with the standard one are as follows:

$$\begin{aligned}f(x) &= 1 - x + x^2 \\ f(x) &= k * x^2.\end{aligned}$$

8.6 Functional properties of the parabolic activation functions

8.6.1 Derivative

Derivative of the function, as we know, is essential to be used in back-propagation. Both the functions SBAF based parabola, and AReLU based parabola are differentiable throughout their domain with the following derivatives:

$$\begin{aligned}\frac{dy}{dx}(1 - x + x^2) &= -1 + 2*x \\ \frac{dy}{dx}(k * x^2) &= 2*k*x.\end{aligned}$$

Similarly, a standard parabola would have the following derivative:

$$\frac{dy}{dx}(a*x^2 + b*x + c) = 2*a*x + b.$$

Looking at the functions, we can see that the derivatives are of the order of $O(x)$, that is, there is a possibility that the gradient could explode during backpropogation. Therefore, to avoid that, the dataset used during the tests has to be scaled appropriately. More on the Vanishing gradients and Exploding Gradients has been talked about in the subsequent subsections.

8.6.2 Existence of fixed point

What is a fixed point? A fixed point is a point which is unchanged when the function is applied on it. Mathematically, it refers to

$$f(x_0) = x_0.$$

If we can prove that the equation has a fixed point, this property helps in convergence and can be used in neural networks for classification. For the SBAF based parabola, as we are not tuning any hyperparameters, we get the following fixed point:

$$\begin{aligned}1 - x + x^2 &= x\\ 1 - 2 * x + x^2 &= 0\\ (1-x)^2 &= 0\\ x &= 1.\end{aligned}$$

Hence $x = 1$ is a fixed point for the SBAF based parabola.

For the A-ReLU based parabola $(k * x^2)$, we obtain the condition for the existence of a fixed point (Apart from the trivial case of $x = 0$) to be

$$k > 1/t$$

Proof.

$$\begin{aligned}k * x^2 &= x\\ x(k * x - 1) &= 0\\ x &= 1/k\\ k &= 1/t.\end{aligned}$$

Hence, we obtain the condition for the existence of a fixed point to be $k > 1/t$. This means that for the function to have a fixed point, k should be taken such that k is greater than $(1/t)$, where t is the range of the input being assumed. $\square$

As we assume the range to be within $1 (t = 1)$, we approximate the value of k by trying to minimize the square error between the standard ReLU activation and our novel activation in the positive half, i.e.

$$\min \int_0^t (k * x^2 - x)^2 dx.$$

Since $k > 1/t$ implies $k > 1$ ($t = 1$ in this case). For our tests, we obtained $k = 1.25$ and have used this value. SBAF Parabola vs. X graph and $1.25x^2$ vs. x graph is shown below (Fig. 18).

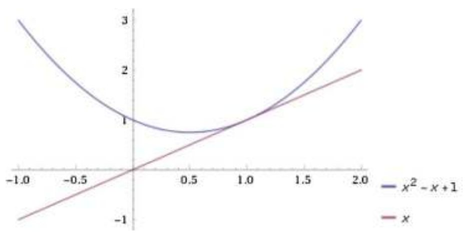
Similarly, for a generic parabola of the form

$$y = a * x^2 + b * x$$

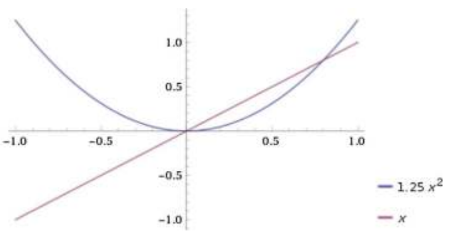
as taken in the previous subsection, the fixed point condition is as follows:

$$\begin{aligned}a * x^2 + b * x &= x\\ a * x^2 + (b-1) * x &= 0\\ a * x^2 &= (1-b) * x\\ a * x &= 1 - b\\ x &= (1-b)/a.\end{aligned}$$

Therefore, as long as our input range is greater than the above value, i.e. $|t| >= (1-b)/a$, a fixed point exists.

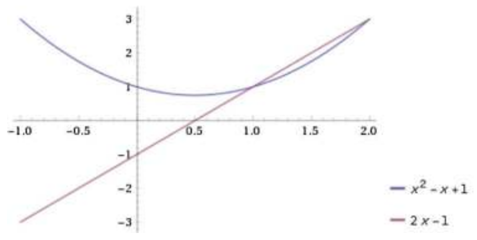


(a)

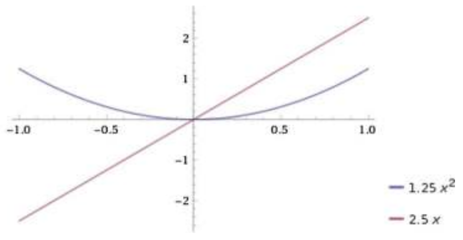


(b)

Fig. 18. Non-monotone activation units derived from SBAF and A-ReLU. (a) SBAF parabola: parabolic activation against input, derived from SBAF. (b) A-ReLU parabola: parabolic activation derived from A-ReLU with $k = 1.25$.



(a)



(b)

Fig. 19. (a) SBAF parabola: parabolic activation derived from SBAF. (b) ReLU parabola: parabolic activation derived from Approximate ReLU.

8.6.3 Exploding and vanishing gradients

Being an activation function, we cannot ignore the issue of exploding and vanishing gradient as we talked about before. As seen from the plots below, the probability of vanishing gradient occurs only at specific points in the function (we are addressing shallow networks at this point). In terms of exploding gradient, if we are able to limit the input values to a suitable range, we can control the exploding gradient issue. The SBAF parabola is more stable towards exploding gradient in the positive half while the ReLU parabola is more stable in the negative half. Plots for SBAF Parabola vs. its derivative and x^2 vs. its derivative are shown below (Fig. 19).

9 Network architecture

The architecture to explore the proposed activation functions is based on a multilayer perceptron that internally deploys back-propagation algorithm to train the network. Neurons are fully connected across all the layers and output of one layer is supplied as input to neurons of subsequent layer during the forward pass. Alternatively, the backward pass computes the error gradient and transmits it back into network in the form of weight-adjustments. Computation of gradients require calculating the derivatives of activation functions (SBAF and A-ReLU) which have been explained briefly in Algorithms 2 and 3.

The implementation of these algorithms is done in Python installed on an x64 based AMD E1-6010 processor with 4GB RAM. The github repository[4] stores the Python code for the SBAF and A-ReLU activations. The purpose of this exercise is

[4] https://github.com/mathurarchana77/A-RELUandSBAF

to demonstrate the performance of the activation functions used on variety of data sets. The Python implementation of these activation functions is done from scratch [68]. The whole architecture is implemented as a nested list, where the network is stored as a single outer list and multiple layers in the network are maintained as inner lists. A dictionary is used to store connection weights of the neurons, their outputs, the error gradients and other intermittent results obtained during back propagation of errors. The computation associated with the processing of neurons in the forward and backward pass are indicated in the algorithms below. The next sections explore the details involved in execution of these algorithms on different feature sets and investigates the performances by comparing them with the state-of-the-art activation functions.

Algorithm 2: Optimise weights and biases by using back propagation on SBAF.

```
Initialize all weights $w_{ij}$, biases $b_i$, $n\_epochs$, lr, k and $\alpha$;
for Each training tuple I in the database do
    for each unit j in input layer do
        $O_j \leftarrow I_j$;
    end
    The forward Pass;
    for each unit j in the hidden layer do
        $h_{jnet} \leftarrow \sum_i w_{ji} O_i + b_j$;
        $h_{jout} \leftarrow \frac{1}{1+k(h_{jnet})^{\alpha}(1-h_{jnet})^{(1-\alpha)}}$;
    end
    for each unit j in the output layer do
        $o_{jnet} \leftarrow \sum_i w_{ji} h_{iout} + b_j$;
        $o_{jout} \leftarrow \frac{1}{1+k(o_{jnet})^{\alpha}(1-o_{jnet})^{(1-\alpha)}}$;
    end
    The Backward pass;
    for each unit j in the output layer do
        $d \leftarrow \frac{o_{jout}(1-o_{jout})}{o_{jnet}(1-o_{jnet})}(o_{jnet} - \alpha)$;
        $E \leftarrow d.(T_j - O_j)$;
    end
    for each unit j in the hidden layer do
        $d \leftarrow \frac{h_{jout}(1-h_{jout})}{h_{jnet}(1-h_{jnet})}(h_{jnet} - \alpha)$;
        $E \leftarrow d \sum_k E_k W_{kj}$;
    end
    The weights update;
    for each weight $w_{ji}$ in network do
        $\triangle w_{ji} \leftarrow lr.E_j.h_{jout}$;
        $w_{ji} \leftarrow w_{ji} + \triangle w_{ij}$;
    end
    The bias update;
    for each weight $b_i$ in network do
        $\triangle b_i \leftarrow lr.E_j$;
        $b_i \leftarrow b_i + \triangle b_i$;
    end
end
```

Algorithm 3: Optimise weights and biases by using back propagation on A-ReLU.

```
Initialize all weights $w_{ij}$, biases $b_i$, $n\_epochs$, lr, k and n;
for Each training tuple I in the database do
    for each unit j in input layer do
        $O_j \leftarrow I_j$ ;
    end
    for each unit j in the hidden layer do
        $h_{jnet} \leftarrow \sum_i w_{ji} O_i + b_j$ ;
        $h_{jout} \leftarrow k.(h_{jnet})^n$ ;
    end
    for each unit j in the output layer do
        $o_{jnet} \leftarrow \sum_i w_{ji} h_{iout} + b_j$;
        $o_{jout} \leftarrow k.(o_{jnet})^n$;
    end
    for each unit j in the output layer do
        $d \leftarrow n.k^{\frac{1}{n}}.(o_{jout})^{\frac{n-1}{n}}$ ;
        $E \leftarrow d.(T_j - O_j)$;
    end
    for each unit j in the hidden layer do
        $d \leftarrow n.k^{\frac{1}{n}}.(h_{jout})^{\frac{n-1}{n}}$ ;
        $E \leftarrow d \sum_k E_k w_{kj}$;
    end
    for each weight $w_{ji}$ in network do
        $\triangle w_{ji} \leftarrow lr.E_j.h_{jout}$ ;
        $w_{ji} \leftarrow w_{ji} + \triangle w_{ij}$ ;
    end
    for each weight $b_i$ in network do
        $\triangle b_i \leftarrow lr.E_j$ ;
        $b_i \leftarrow b_i + \triangle b_i$;
    end
end
```

10 Data

The PHL-EC (University of Puerto Rico's Planetary Habitability Laboratory's Exoplanet Catalog) dataset [10,45] consists of a total of 68 features, 13 categorical and the remaining 55 are continuous. The catalog uses stellar data from the Hipparcos catalog [44,69] and lists 3771 confirmed exoplanets, out of which 47 are meso and psychroplanets and the remaining are non-habitable. The features included in catalog are atmospheric type, mass, radius, surface temperature, escape velocity, Earth Similarity Index, flux, orbital velocity, etc. The difference between The PHL-EC and other catalogs is that PHL-EC models some attributes when data are not available. This includes estimating surface temperature from equilibrium temperature as well as estimating stellar luminosity and pressure. The presence of observed and estimated attributes presents interesting challenges.

This paper uses the PHL-EC data set, of which an overwhelming majority are non-habitable samples. Primarily, PHL-EC class of labels of exoplanets are non-habitable, mesoplanets, and psychroplanets [10]. The class imbalance is observed in the ratio of thousands to one. Further, the potentially habitable planets (meso or psychroplanets) have their planetary attributes in a narrow band of values such that the margins between mesoplanets and psychroplanets are incredibly difficult to discern. This poses another challenge to the classification task.

The classes in the data are briefly described below:

(1) **Non-Habitable**: planets, mostly too hot or too cold, may be gaseous, with non-rocky surfaces. Such conditions do not favor habitability.
(2) **Mesoplanet**: generally referred to as Earth-like planets, the average global surface temperature is usually between 0 °C and 50 °C. However, Earth-similarity is no guarantee of habitability.
(3) **Psychroplanet**: these planets have mean global surface temperature between −50 °C and 0 °C. Temperatures of psychroplanets are colder than optimal for sustenance of terrestrial life, but some psychroplanets are still considered as potentially habitable candidates.

The data set also has other classes, though insignificant in number of samples for each class. These are thermoplanets, hypopsychroplanets and hyperthermoplanets. The tiny number of samples in each class makes it unsuitable for the classification task and these classes are therefore not considered for class prediction. Certain features such as the name of the host star and the year of discovery have been removed from the feature set as well. We filled the missing data by class-wise mean of the corresponding attribute. This is possible since the amount of missing data is about 1% of all the data. The online data source for the current work is available at the university website[5].

11 Experiments

The PHL-EC dataset has 3771 samples of planetary data for 3 classes of planets and 45 features (after pruning unnecessary features). As already mentioned, a Multi Layered Perceptron (MLP) is implemented at the core for classifying planets. The MLP internally utilizes gradient descent to update weights and biases during classification. The connection weights and biases are randomly initialized. The activation functions used in MLP are sigmoid, SBAF, A-ReLU and ReLU, and details for implementing SBAF and A-ReLU along with the computation of gradients in both cases is shared in Appendix C.2.

As a part of the experiments, different data sets are explicitly built by selecting certain combination of features from the original feature set. The idea behind doing this is to evaluate the performance of functions for a variety of feature sets. The following subsection illustrates various sets of data used on the activation functions and their results. In all these cases, the training set consists of 80% of samples and remaining 20% are used for testing. SBAF uses the hyper parameters, α and k, that are tuned during execution of code and best results were seen at $\alpha = 0.5$ and $k = 0.91$ (in agreement with the fixed point plots we observed, $k = 1$ does not alter classification performance). Similarly, for A-ReLU, k and n were set to 0.5 and 1.3 respectively (this is from the evidence by the approximation to ReLU-empirical observations match). This eliminates the need for parameter tuning. Tables 1–3 refer to the confusion matrix and classification results are shown in Tables 4 and 5. The accuracy, precision and recall are indicated for all classes of planets: Non habitable, Mesoplanet and Psychroplanet. Detailed analysis of these tables is discussed in Section 11.1. As seen from the tables, A-ReLU has outperformed ReLU in almost all the cases under investigation and SBAF has performed significantly better than the parent function, Sigmoid. Section 11.2 captures a comparison of SBAF, A-ReLU with another activation function, Swish.

[5] http://phl.upr.edu/projects/habitable-exoplanets-catalog/data/database

Table 1. Confusion Matrix obtained while executing SBAF using all 45 features; the three classes are non habitable, Mesoplanet and Psychroplanet; the confusion matrix shows all 745 non-habitable planets classified correctly; 5 Mesoplanets and 1 Psychroplanet is also labeled correctly by the network. Please note, all rocky exoplanets have been considered and therefore the numbers do not match up with the total count reported in introduction.

		Predicted		
		Non-Habitable	Mesoplanet	Psychroplanet
Actual	Non-habitable	745	0	0
	Mesoplanet	0	5	4
	Psychroplanet	0	0	1

Table 2. Confusion Matrix obtained while executing A-ReLU using No-Surface temperature dataset (Case 3), the three classes are non habitable, Mesoplanet and Psychroplanet. Please note that dataset is used post upsampling.

		Predicted		
		Non-Habitable	Mesoplanet	Psychroplanet
Actual	Non-habitable	333	0	0
	Mesoplanet	0	32	0
	Psychroplanet	0	0	74

Table 3. Confusion Matrix for A-ReLU using all feature dataset (Case 1), we observe that 1 non-habitable planet is misclassified as psychroplanet, and rest of the prediction are correct. Please note that the dataset is used post upsampling.

		Predicted		
		Non-Habitable	Mesoplanet	Psychroplanet
Actual	Non-habitable	343	0	1
	Mesoplanet	0	38	0
	Psychroplanet	0	0	76

11.1 Chosen feature sets

The different combination of features employed on the traditional and proposed activation functions is explored here. A case-by-case exploration of these feature sets with their performance comparison, is indicated in Tables 4 and 5. The first column of these tables reveals features, marked with their count and the case number. Remaining columns reflect class-wise accuracy, precision and recall of the 4 activation functions.

(1) To begin with, all features are used as input to the neural network, thus leading to 45 input and 3 output neurons. Since the inherent characteristics of each activation function are different, each one uses a different number of neurons in the hidden layer to reach convergence. For sigmoid, 20 hidden neurons are used while SBAF, A-ReLU and ReLU used 11 neurons in the hidden layer. Sigmoid gives the best accuracy at learning rate of 0.015, momentum of 0.001 and at 500 epochs, while SBAF, A-ReLU and ReLU gives the best results at 100 epochs keeping the other parameters same.

Table 4. Performance analysis of different Activation functions used in the study. First column indicates features that are given as input, along with their count. 3rd, 4th, 5th, 6th, and 7th columns show the performance of 5 different activation functions. Accuracy, precision and recall is indicated for every class of planet - Non habitable, Mesoplanet and Psychroplanet. A-ReLU has out performed ReLU, LeakyReLU and Sigmoid by significant difference in performance. Difference in performance is clearly visible in minority classes, Meso and psychro. Note, SBAF and A-ReLU didn't need parameter tuning, at all.

Different feature sets	Performance	Different Activation functions used in the study														
		Sigmoid			SBAF			Approx. Relu			Relu			Leaky ReLU		
		Non-H	Meso	Psychro	Non-H	Meso	Psychro	Non-H	Meso	Psychro	Non-H	Meso	Psychro	Non-H	Meso	Psychro
	Accuracy	1.	0.975	0.975	1.0	0.99	0.994	0.997	1.	0.997	1.0	0.99	0.99	0.994	0.5	0.857
All features (45)	Precision	1.0	0.943	0.932	1.0	1.0	0.2	1.	1.	0.987	0.988	0.998		1.0	0.25	0.857
(Case 1)	Recall	1.0	0.895	0.965	1.0	0.55	1.0	0.997	1.0	1.0	0.998	0.991	0.991	0.994	0.5	0.857
Restricted	Accuracy	0.758	0.741	0.798	0.987	0.854	0.847	1.0	1.0	1.0	0.985	0.834	0.830	0.937	1.0	0.286
Features (6)	Precision	0.719	0.525	0.808	1.	0.681	0.643	1.0	1.0	1.0	1.0	0.5	0.618	1.0	0.18	0.2
(Case 2)	Recall	0.864	0.583	0.384	0.978	0.223	0.943	1.0	1.0	1.0	0.974	0.149	0.925	0.937	1.0	0.286
Without Surface	Accuracy	0.998	0.998	0.997	0.997	0.924	0.927	1.0	1.0	1.0	1.0	1.0	1.0	0.98	1.0	0.67
Temperature (44)	Precision	0.994	0.995	1.	1.	0.484	0.783	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.33
(Case 3)	Recall	1.	1.	0.994	0.996	0.5	0.783	1.0	1.0	1.0	1.0	1.0	1.0	0.98	1.0	0.67
Planet MinMass	Accuracy	0.888	0.825	0.750	0.922	0.922	0.864	0.971	0.946	0.931	0.968	0.937	0.905	0.976	0.67	0.71
Mass and	Precision	0.974	0.534	0.547	1.	0.3	0.456	1.	0.619	0.673	0.996	1.	0.567	0.997	0.25	0.56
Radius (4) (Case 4)	Recall	0.807	0.397	0.814	0.904	0.130	0.912	0.965	0.590	0.846	0.965	0.090	0.974	0.976	0.67	0.71

Table 5. Performance analysis of different Activation functions used in the study. First column indicates input feature along with their count. 3rd, 4th, 5th and 6th column shows the performance of 4 different activation functions. Accuracy, precision and recall is indicated for every class of planet - Non habitable, Mesoplanet and Psychroplanet. A-ReLU has outshined significantly, the parent activation function ReLU in terms of performance. An anomaly is seen for case 8 where sigmoid performed marginally better than SBAF.

Different feature sets	Performance	Different Activation functions used in the study														
		Sigmoid			SBAF			Approx. Relu			Relu			Leaky ReLU		
		Non-H	Meso	Psychro	Non-H	Meso	Psychro	Non-H	Meso	Psychro	Non-H	Meso	Psychro	Non-H	Meso	Psychro
Radius	Accuracy	0.848	0.753	0.608	0.721	0.738	0.754	0.908	0.8311	0.807	0.890	0.861	0.795	0.813	0.5	0.57
and other	Precision	0.856	0.668	0.440	0.617	0.669	0.152	0.919	0.718	0.548	0.883	0.749	0.516	1	0.07	0.03
Star features (Case 5)	Recall	0.655	0.515	0.645	0.962	0.738	0.37	0.867	0.852	0.441	0.866	0.908	0.354	0.813	0.5	0.57
Planet Mass	Accuracy	0.846	0.706	0.58	0.859	0.868	0.881	0.997	1.	0.997	0.877	0.899	0.902	0.837	0.71	0.5
and other	Precision	0.764	0.55	0.337	0.914	0.442	0.307	1.	1.	0.987	0.878	0.554	0.0	0.997	0.116	0.045
Star features (Case 6)	Recall	0.78	0.65	0.27	0.906	0.628	0.177	0.997	1.	1.	0.980	0.580	0.0	0.837	0.71	0.5
Minimum	Accuracy	0.85	0.683	0.7	0.850	0.8007	0.857	0.925	0.850	0.868	0.864	0.893	0.864	0.88	1	0
Mass and other	Precision	0.711	0.75	0.532	1.0	0.190	0.520	0.938	0.269	0.571	0.857	0.0	0.558	1	0.045	0
Star features (Case 7)	Recall	0.925	0.07	0.85	0.798	0.2666	0.8837	0.961	0.233	0.558	0.980	0.0	0.558	0.88	1	0
Minimum Mass	Accuracy	0.958	0.8	0.808	0.858	0.77	0.716	0.904	0.799	0.839	0.901	0.796	0.790	0.87	1	0
P. Distance and	Precision	1.	0.785	0.649	0.911	0.35	0.38	0.894	0.506	0.666	0.893	0.000	0.478	1	0.05	0
Star features (Case 8)	Recall	0.875	0.55	0.925	0.846	0.106	0.746	0.958	0.636	0.349	0.9485	0.0	0.888	0.868	1	0

(2) A subset of features (restricted features) consisting of Planet Minimum Mass, Mass, Radius, SFlux Minimum, SFlux Mean, SFlux Maximum are used as input. All networks used 4 neurons in hidden layer to stabilize. Sigmoid converged at learning rate of 0.1 and momentum of 0.01. In parallel, SBAF A-ReLU and ReLU converged at learning rate of 0.08, momentum at 0.004 and number of epochs as 300. The performance of all the classifiers is reported in Table 4. It is evident from the table that the network is able to classify even the minority class samples (Mesoplanets and Psychroplanets) with fairly decent accuracy.

(3) It has already being shown that Surface Temperatures (ST) can distinguish habitable planets from non-habitable ones with a large degree of precision. Therefore, it becomes essential to ascertain if the proposed activation functions (SBAF and A-ReLU) as well as the already established ones (sigmoid and ReLU), can perform classification when ST is removed from feature set. To achieve this, the next set of features are those from which ST is removed. Thus, exoplanet features used as input are Zone Class, Mass Class, Composition Class, Atmosphere Class, Min Mass, Mass, Radius, Density, Gravity, Esc Vel, SFlux Min, SFlux Mean, SFlux Max, Teq Min (K) , Teq Mean (K), Teq Max (K), Surf Press, Mag, Appar Size, Period, Sem Major Axis, Eccentricity, Mean Distance, Inclination, Omega, HZD, HZC, HZA, HZI, ESI and Habitable. The features of parent Star that belong to feature set are Mass, Radius, Teff, Luminosity, Age, Appar Mag, Mag from Planet, Size from Planet, Hab Zone Min and Hab Zone Max. For SBAF and A-ReLU, the 44 featured data set is tuned at learning rate = 0.01, momentum = 0.001 and epochs = 300. The number of units in the hidden layer is 12 for all four activation functions. It is interesting to inspect that A-ReLU performs 100% classification for all three classes and this is at par with A-ReLU. However, sigmoid performs marginally better than SBAF.

(4) A unique combination of planetary attributes like Minimum Mass, Mass, Radius and Composition class is taken as input. The essence of selecting these features is to know whether the activation functions can perform classification by solely using Mass and mass related features. It becomes a challenging machine learning problem since the discrimination of planets is not at all clear by features such as mass of planets. Interestingly, for this kind of problem, A-ReLU performs better than the parent function ReLU, and SBAF outpaces Sigmoid with substantial difference in accuracy. The networks are tuned at learning rate of 0.2 and momentum of 0.03 at 400 epochs. It uses 3 neurons in the hidden layer.

(5) The following 4 cases (Cases 5–8) are set for an exclusive exploration, where exoplanets are classified using one feature from planets at a time along with multiple features of the parent star. At first, the planet's radius along with 6 star features (Mass, Radius, Teff, Luminosity, Hab Zone Min and Hab Zone Max) are fed as input to the four activation functions. With 4 neurons in the hidden layer, the learning rate and momentum for the network are tuned to 0.09 and 0.001. Results are achieved at 300 epochs, as shown in Table 5. The performance of A-ReLU is exceedingly better than the rest of the activation functions.

(6) Continuing on the same line of work, another feature set bearing planet's mass and previously used 6 star features are fed to the network. Hidden neurons are same in number as in the previous case. Here too, A-ReLU has performed better from its parent activation function, ReLU as well as from the other functions in terms of classification accuracy, precision and recall.

(7) Table 5 shows the performance of classification using Minimum mass as planet's feature and remaining parent star features as input keeping other arrangements same as the previous cases. Comparing accuracy in each activation function, A-ReLU has again performed better for all the 3 classes of planets.

(8) A slightly different grouping of feature is attempted here. Two planet features, mass and minimum distance computed from parent star, and the remaining 6 star features are used as input. For this particular combination of features, A-ReLU and ReLU performs at par and sigmoid performs better classification in comparison with the rest. This is the only case where an irregularity is seen in the performance.

The performance metrics shown in Tables 4 and 5 indicate that SBAF and A-ReLU outshine sigmoid and ReLU in almost all of the cases. Some more critical parameters in terms of training time, CPU utilization and memory requirements, for the execution of activation functions are reported in Table 6. SBAF and A-ReLU take the shortest time to reach convergence even though number of epochs are kept same. CPU utilization is minimum for A-ReLU followed by ReLU, SBAF and sigmoid. Memory consumption is balanced and almost equal for all the functions under study. It is worth mentioning that values reported in the table are obtained after taking the average of all 8 executions from above cases.

An investigation of the confusion matrix resulting from the execution of SBAF, is disclosed in Table 1. It is noted that, even though there are considerably large number of samples of non-habitable planets, SBAF classifies non-habitable and Psychroplanet unmistakably, with the accuracy of 1 and 0.994 for the respective classes. However, Mesoplanets are not flawlessly classified, (out of 9 samples, 5 were correctly labeled and rest of the 4 were mistakenly labeled as Psychroplanet), the reason is attributed to the class distribution in the data set. The number of samples of non-habitable, Mesoplanet and Psychroplanet are 3724, 17 and 30 respectively. Evidently, Mesoplanet are lowest in training samples and this leads to insufficiency in training the network for this class. Nevertheless, this can be handled by balancing the class samples in the training data (Tab. 7).

Even though our focus is not all on the performance of statistical machine learning or other learning algorithms on the PHL-EC data, we feel its necessary to report the performance of some of those briefly. This was a post-facto analysis (the realization dawned upon us much later) done to ascertain the efficacy of our methods compared to methods such as Gaussian Naive Bayes (GNB), Linear Discriminant Analysis (LDA), SVM, RBF-SVM, KNN, DT, RF and GBDT [17]. These methods did not precede the exploration into activation functions. We believe the readers should know and appreciate the complexity of the data and classification task at hand, in particular when hard markers (surface temperature and surface temperature related features which discriminate the habitability classes quite well). This also highlights the remarkable contribution of the proposed activation functions toward performance metrics. It is for this reason, we do not tabulate the results in detail but just state that for the specific cases (Cases 2, 4–8, “six” out of the total “eight” cases considered for our experiments) where “hard marker” features were removed, none of the popular machine learning methods mentioned above reached over 75% accuracy class-wise and 68% overall. This augurs well for the strength of our analysis presented in the manuscript. (Please refer different learning curves and MSE plots for various activation functions Figs. 20 and 21). Figure 20 indicates Sogmoid attains saturation at a small epoch value and this can be interpreted as the representational power of the activation function (since identical network architecture was used). If the activation function is more expressive, it will, in general, get a lower MSE when trained sufficiently long enough. Therefore, its not surprising that SBAF has the lowest MSE (and thus the highest representational power), since its shape can easily be changed. A-ReLU should not be very surprising either since the parameters render A-ReLU flexibility. Even though SBAF and A-ReLU has more parameters compared to sigmoid, their faster convergence is striking.

Table 6. Analysis of Training time, CPU utilization and Memory usage for Sigmoid, SBAF, A-ReLU and ReLU at epochs = 500. The reported values are averaged over all 8 executions mentioned above. The SBAF has the lowest training time followed by A-ReLU. CPU utilization is minimum for A-ReLU; SBAF has marginally better CPU utilization from the parent activation function sigmoid.

	Activation functions (Epochs = 500)				
	Sigmoid	SBAF	A-ReLU	ReLU	Leaky ReLU
Learning rate, momentum	0.01, 0.001	0.01, 0.01	0.01, 0.001	0.01, 0.01	0.01, 0.001
Time (s)	409.33	281.97	312.31	358.01	574.6
CPU Utilization (%)	54.6	53.6	42.8	48.4	41.81
Memory Usage (MB)	67.7	67.8	67.4	67.8	45.45

Table 7. Performance analysis of different Non-monotonic activation functions used in the study. Precision, Recall, F-Score and Accuracy is indicated for every class of planet - Mesoplanet, Non habitable and Psychroplanet.

Datasets	Performance	Non-monotonic Activation Functions											
		Generic Parabola (with a: 0.65, b: 0.5)			Absolute Value Function			ReLU Based Parabola			SBAF Based Parabola		
		Meso	Non-H	Psychro	Meso	Non-H	Psychro	Meso	Non-H	Psychro	Meso	Non-H	Psychro
All Features	Precision	100.0	99.8	71.43	77.78	99.59	75.0	80.0	99.59	75.0	88.89	99.8	66.67
	Recall	88.89	99.59	100.0	77.78	99.8	60.0	88.89	99.59	60.0	88.89	99.59	80.0
	F-Score	94.12	99.69	83.33	77.78	99.69	66.67	84.21	99.59	66.67	88.89	99.69	72.73
	Accuracy	99.8	99.4	99.6	99.21	99.4	99.4	99.4	99.21	99.4	99.6	99.4	99.4

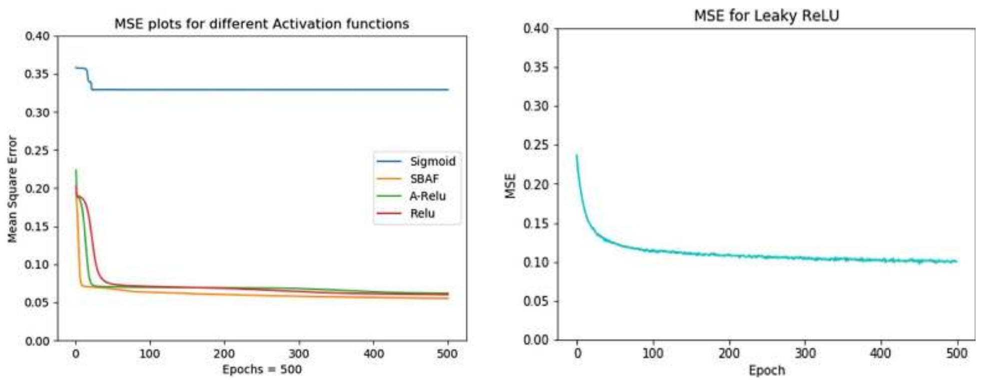


Fig. 20. MSE plots for all Activation functions executed for 500 epochs on Restricted features of PHL-EC data: sigmoid attains saturation at an early MSE value and fails to learn effectively when compared with SBAF, A-ReLU and ReLU.

11.2 A note on Swish activation

Swish activation function [70] is derived from ReLU with the intention of smoothing the discontinuity and singularity of ReLU. The function is known to work well, in practice, when deployed on deep neural nets. A-ReLU is also designed to overcome similar problems in ReLU. The difference however lies in theoretical validation of A-ReLU in approximating over-parametrized RelU networks. Swish presents empirical evidence alone. We presented theoretical justification of A-ReLU, to the extent of parameter selection, degree of approximation and other functional properties that will ensure it works well, even in shallow networks constrained by restricted features. As we observe, this is indeed the case. Please note, throughout the manuscript, we endeavored to justify cheap learning paradigm, supported by theoretical results. A-ReLu fits in the landscape as observed and Swish does not, apparently. Nonetheless, it is relevant to compare the two since both attempts to address the same issues in ReLU activation.

It is quite possible, over deep networks, Swish is able to generalize and extract features required to perform well. It does indeed [70]. However, as we observed, the expressive and generalization power of Swish activation is limited in shallow networks with constrained features. This impression might alter if theoretical investigations, beyond the scope of the current manuscript, are carried out in future. For example, in Cases 4–8 (constrained feature sets), accuracy/precision/recall scores of Swish activation on one of the two minority classes, Meso and Psychro are 0! (Tab. 8). The comparison also shows that Swish performs at par with SBAF and A-ReLU when dataset includes all 45 features. For case 3, when feature set is large, A-ReLU manages to get 100% accuracy, whereas SBAF and Swish showed satisfactory results in classifying Non-H, Meso and Psychro planets. Swish almost failed to classify Meso and Psychro planets in all the other cases (Cases 2, 4–8), even though SBAF and A-ReLU showed fair classification particularly for these planets. To derive the essence of this exercise, Swish performs at par when feature set is sufficiently large (Cases 1 and 3), but fails to identify Meso and Psychro when size of feature set shrinks and this is not the case with SBAF and A-ReLU. Figure 21 shows the validation curves and learning curve for SBAF shows a good fit for the data sets used. Neural network with A-ReLU shows a generalization gap between the training and validation loss curves which implies a specific case where in validation dataset is easier to predict than training data. However, when plotted with all features, a good fit is seen. Swish function creates over-fitted learning curves for the two datasets (Mass, radius data

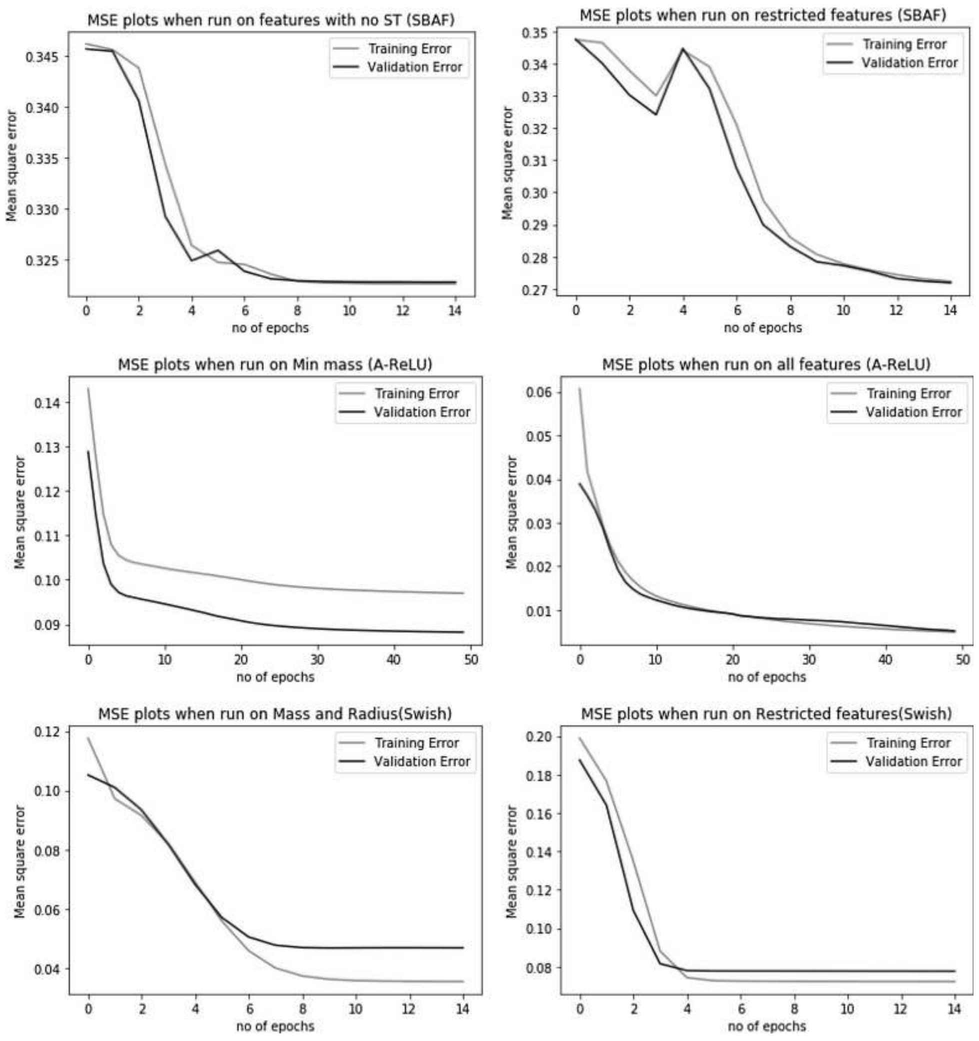


Fig. 21. Training error vs. validation error for SBAF, A-ReLU and Swish is shown using different data sets.

and restricted features data). Nonetheless, the training and validation loss decreases to a point of stability in all the plots shown here.

F1 Score: The F1 score is the harmonic mean of the recall and precision. It is a neat way to describe performance of classifiers, particularly in cases where data are imbalanced. For the purpose of experimentation, we computed F1 scores for Swish, SBAF and A-ReLU activation functions. Using case numbers from Table 8, for Cases 2, 4, 7 and 8, Swish reported an F1 score of zero for mesoplanets and simultaneously, a zero was observed for psychroplanets for Cases 5–7. In the remaining cases, the score of Swish is at par with SBAF but interestingly, A-ReLU gives highest F1 scores in all cases, as against SBAF and Swish.

Table 8. Comparative analysis of SWISH activation function with SBAF and A-ReLU is presented in this table.

Different feature sets	Performance	Comparison of activation functions with Swish								
		SWISH			SBAF			Approx. Relu		
		Non-H	Meso	Psychro	Non-H	Meso	Psychro	Non-H	Meso	Psychro
	Accuracy	1.	0.95	0.95	1.0	0.99	0.994	0.997	1.	0.997
All features (45)	Precision	1.0	0.86	0.80	1.0	1.0	0.2	1.	1.	0.997
(Case 1)	Recall	1.0	0.52	0.96	1.0	0.55	1.0	0.997	1.0	1.0
Restricted	Accuracy	0.98	0.83	0.82	0.987	0.854	0.847	1.0	1.0	1.0
Features (6)	Precision	1.0	0.0	0.59	1.	0.681	0.643	1.0	1.0	1.0
(Case 2)	Recall	0.97	0.0	1	0.978	0.223	0.943	1.0	1.0	1.0
Without Surface	Accuracy	1	0.97	0.97	0.997	0.924	0.927	1.0	1.0	1.0
Temperature (44)	Precision	1	0.74	1.	1.	0.484	0.783	1.0	1.0	1.0
(Case 3)	Recall	1.	1.	0.85	0.996	0.5	0.783	1.0	1.0	1.0
Planet MinMass	Accuracy	0.97	0.93	0.90	0.922	0.922	0.864	0.971	0.946	0.931
Mass and	Precision	1.0	0.0	0.55	1.	0.3	0.456	1.	0.619	0.673
Radius (4) (Case 4)	Recall	0.96	0.0	1.0	0.904	0.130	0.912	0.965	0.590	0.846
Radius	Accuracy	0.792	0.751	0.791	0.721	0.738	0.754	0.908	0.8311	0.807
and other	Precision	0.709	0.621	0.0	0.617	0.669	0.152	0.919	0.718	0.548
Star features (7) (Case 5)	Recall	0.896	0.786	0.0	0.962	0.738	0.37	0.867	0.852	0.441
Planet Mass	Accuracy	0.803	0.889	0.902	0.859	0.868	0.881	0.997	1.	0.997
and other	Precision	0.801	0.588	0.0	0.914	0.442	0.307	1.	1.	0.987
Star features(7) (Case 6)	Recall	0.997	0.095	0.0	0.906	0.628	0.177	0.997	1.	1.
Minimum	Accuracy	0.740	0.893	0.846	0.850	0.8007	0.857	0.925	0.850	0.868
Mass and other	Precision	0.740	0.0	0.0	1.0	0.190	0.520	0.938	0.269	0.571
Star features(7) (Case 7)	Recall	1.0	0.0	0.0	0.798	0.2666	0.8837	0.961	0.233	0.558
Minimum Mass	Accuracy	0.694	0.796	0.811	0.858	0.77	0.716	0.904	0.799	0.839
P. Distance and	Precision	0.666	0.0	0.527	0.911	0.35	0.38	0.894	0.506	0.666
Star features(8) (Case 8)	Recall	0.984	0.0	0.301	0.846	0.106	0.746	0.958	0.636	0.349

12 Another piece in the puzzle: Unsupervised learning to group exoplanets

A significant portion of our discussion on habitability revolved around machine classification for exoplanets, and for good reason. As pointed out earlier, classification of objects is equivalent to predicting class labels based on existing (training) class labels. Thus, the efficacy of such algorithms/methods depends on quality of training data. Classification methods fall under supervised learning. We ask the following question: what do we anticipate if the grouping of exoplanets is accomplished by unsupervised learning methods, i.e. clustering? How do we frame the problem so that the solution helps us in finding the right habitable candidates without their trained class labels? Clustering deals with a class of problems where we are not given the target labels with the data. Essentially, we only have the data vector, X as our training examples. The only thing we can really do is find some patterns in the data. Cluster analysis, or clustering, is one such class of methods. In clustering, we try to find reasonable "clusters" of data–data that is grouped together in some meaningful way. What constitutes "meaningful" decides what clustering algorithm we use. If we decide to follow this path of unsupervised learning, what are the features/attributes we use to arrive at the clusters we wish to obtain? We attempt to answer these questions here.

We present the following hypothesis: Number of potentially habitable planets (tagged as mesoplanets or psychroplanets) in PHL-EC are far less compared to the number of non-habitable planets. Therefore, finding and organizing these mesoplanets or psychroplanets to a group/cluster amounts to determining anomalies in the data set, since the fraction of these planets in comparison to non-habitable ones is significantly lesser. We will use the same feature set previously employed in classification except for the fact that we will do away with the labels-mesoplanets, psychroplanets and non-habitable planets as the clustering/anomaly detection method will not possess the knowledge of class labels during detection. This is the hallmark of unsupervised learning methods. In anomaly detection based clustering, it is assumed that the anomaly will be part of a sparse or small cluster, and normal data instances belong to the large or dense cluster. Such a technique depends on a cluster-based anomaly

Table 9. Hausdorff distances of different attributes.

Features	Hausdorff distance
Period	0.3221775035782434
Surface temperature	0.09311046300418734
Escape velocity	0.06300384960911469
Gravity	0.02891863077548809
Eccentricity	0.021038884550142307
Surface pressure	0.016395465088041485
Density	0.01466598293926538
Radius	0.01396646644180649
Mass	0.008171181718606813
Flux	0.007764217430292616

score assigned to each data instance. Based on this score, anomaly data are detected. The score is computed as a distance between the data point and the centroid of the cluster. The method relies heavily on the knowledge of the number of clusters in the data-set. This specific clustering technique can be compared with other clustering techniques where planets are clustered based on similarities/dissimilarities.

12.1 Attribute selection

An important challenge in the dataset is that a total of about 1% of data is missing and in order to overcome this, we performed imputation with R using MICE package. The MICE package in R helps you impute missing values with plausible data values. These plausible values are drawn from a distribution specifically designed for each missing data point. Following this, a clustering technique can be used on the processed data set. The selection of appropriate attributes is very crucial for any analysis. We plot the distribution of attributes of habitable and non-habitable exoplanets and try to find a pattern in their distribution. We have made use of Hausdorff distance metric to find the distance between the distribution of attributes of habitable and non-habitable planets. Hausdorff distance between two distribution A and B is defined as following,

$$d_H(A,B) = \max_{\forall a \in A} \min_{\forall b \in B} |a-b|. \tag{23}$$

For every point a in A, we find the distance of the nearest point in B, store this in a set, and then find the maximum of all these distances (Tab. 6).

Data points in the same cluster should have similar properties, while the data points in different clusters should have dissimilar properties. Clustering helps us find the relationship between the data points by observing what clusters they fall into. We have made use of various clustering methodologies such as spectral clustering, anomaly detection and PSO (Particle Swarm Optimization) based clustering (Tab. 9).

12.2 Spectral clustering

Spectral clustering is used to identify communities of nodes in a graph based on the edges connecting them. Spectral clustering uses information from the eigenvalues of special matrices built from the graph or the data set. In order to get the eigenvalues, we need to build an adjacency matrix of the graph and convert the matrix to Degree matrix. Subsequently, we construct the Laplacian graph (L is the Degree matrix-Adjacency matrix). This matrix is normalized for mathematical efficiency. To reduce

Table 10. Habitable Planets in Earth's Cluster for spectral clustering.

Planet names
GJ 667 C e
HD 40307 g
HD 283869 b
Kapteyn b
Kepler-62 e
Kepler-62 f
Kepler-283 c
Kepler-296 f
Kepler-440 b
Kepler-442 b
Kepler-452 b
Kepler-1229 b
Kepler-1410 b
Kepler-1544 b
KOI-4427 b
tau Cet e

the dimensions first, the eigenvalues and the respective eigenvectors are calculated. If the number of clusters is k, then the first eigenvalues and their eigenvectors are stacked into a matrix such that the eigenvectors form columns of the matrix. Clustering is performed on this reduced data. The number of zero eigenvalues corresponds to the number of connected components in our graph. We obtained the following habitable planets in our earth's cluster, based on this method (Tab. 10).

12.3 Clustering via anomaly detection

We have made use of Isolation forest [71] method for anomaly detection [72,73]. This method is fundamentally different and efficient means of outlier detection from the commonly used basic distance and density measures. It isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. A normal point will be more densely clustered while an anomalous point will be far away from the densely distributed points. Thus, while randomly partitioning the domain space, the anomalous points will be detected in lesser number of partitions than a normal point. Smaller number of partitions means distance from the root node is smaller. An anomaly score is required for any anomaly detection method. The anomaly score for an instance x is given by:

$$s(x,n) = 2^{-(h(x)/c(n))} \tag{24}$$

where $E(h(x))$ is the average of the path lengths from the Isolation forest and $c(n)$ is given by,

$$c(n) = 2H(n-1) - (2(n-1)/n). \tag{25}$$

The anomaly score lies between 0 and 1. For our analysis, it produces a total of 66 anomalies, out of which 18 are classified as habitable by the PHL-EC dataset. Following are the 18 planets (Tab. 11).

Table 11. Anomaly Score of Habitable Planets.

Planet name	Anomaly score
HD 40307 g	0.6618491281
HD 283869 b	0.5876162072
Kepler-22 b	0.6979305999
Kepler-62 e	0.6085267624
Kepler-62 f	0.6939491374
Kepler-174 d	0.6932244356
Kepler-298 d	0.5842785633
Kepler-440 b	0.5830443768
Kepler-442 b	0.6263227795
Kepler-443 b	0.646733081
Kepler-452 b	0.7317805784
Kepler-1090 b	0.6754203354
Kepler-1540 b	0.6644749057
Kepler-1544 b	0.6564644722
Kepler-1552 b	0.6654079436
Kepler-1638 b	0.6839070943
tau Cet e	0.6337382861
earth	0.7680744097

12.4 Particle swarm optimization

Particle swarm optimization (PSO) is a biologically inspired metaheuristic for finding the global minima of a function. The population of particles keeps track of its current position and the best solution it has encountered, called pbest. Each particle also has an associated velocity that traverses the search space. The swarm keeps track of overall best solution, called gbest. Each iteration of swarm updates the velocity of the particle towards its pbest and gbest values. The minimization function is quantization error which is a metric of error introduced by moving each point from its original position to its associated quantum point. In clustering, we often measure this error as the root-mean-square error of each point(moved to the centroid of its cluster).

When a particle finds a location that is better than the previous locations, it updates this location as the new current best for the particle i. It is no longer required to update current best solutions when the objective no longer improves or after a certain number of iterations. Here x_i and v_i are position vector and velocity vector respectively. The new velocity is found by the following formula:

$$v_\mathrm{i} \leftarrow w.v_\mathrm{i} + \lambda_\mathrm{g} u_\mathrm{g}(\mathrm{gbest} - p_\mathrm{i}) + \lambda_\mathrm{p} u_\mathrm{p}(\mathrm{pbest}_\mathrm{i} - p_\mathrm{i}) \tag{26}$$

$$p_\mathrm{i} \leftarrow p_\mathrm{i} + v_\mathrm{i}. \tag{27}$$

12.5 PSO based clustering

Here *f(x)* is a function to be minimized which is also called as the fitness function, where x is an n-dimensional array. Algorithm 4 outlines the approach to minimizing *f(x)* using PSO and Algorithm 5 assigns cluster labels. A set of particles are randomly initialized with position and velocity. The position of the particle corresponds to its associated solution. Here each particle has cluster centroids which is the solution and pbest score calculated using the fitness function. The pbest position that corresponds to the minimum fitness is selected to be the gbest position of the swarm.

On each iteration, the algorithm updates the velocity and position of each particle. For each particle, it picks two random numbers u_g, u_p from a uniform distribution, $U(0,1)$ and updates the particle velocity. Here, w is the inertial weight and λ_g, λ_p are the global and particle learning rates. If the new position of the particle corresponds to a better fit than its pbest, the algorithm updates pbest to the new position. Once the algorithm has updated all particles, it updates gbest to the new overall best position. The function g() assigns cluster labels to each data point and returns an array of cluster labels.

We provide n-dimensional array of data and gbest centroids as parameter to function g(). Then we calculate the distance between data points and the centroids and store it in the distance array. Distance array corresponding to each centroid is stored in the global distance array. The data point is assigned to a cluster with minimum distance.

Algorithm 4: Algorithm for GA-PSO.

```
input  : An $n$-dimensional array of data $x$
output: An array of cluster labels
1  for $j \leftarrow 1$ to $max\_iterations$ do
2      for each particle $i \leftarrow 1$ to $n\_particles$ do
3          $u_p, u_g \approx U(0,1)$
4          $v_i \leftarrow w.v_i + \lambda_g u_g(\text{gbest} - p_i) + \lambda_p u_p(\text{pbest}_i - p_i)$
5          $p_i \leftarrow p_i + v_i$
6          if $f(p_i) < f(\text{pbest}_i)$ then
7              $\text{pbest}_i \leftarrow p_i$
8          end
9      end
10     Sort the particles based on fitness value
11     Crossover the top 25 percent particles
12     Mutation(particles)
13     for each particle $i \leftarrow 1$ to $n\_particles$ do
14         if $f(\text{pbest}_i) < f(\text{gbest})$ then
15             $\text{gbest} \leftarrow \text{pbest}_i$
16         end
17     end
18 end
19 cluster $\leftarrow$ g($x$, gbest)
20 return cluster
```

Cluster 4 (Fig. 22) is marked as the earth's cluster as seen in the above graph. There are 298 planets in the earth's cluster out of which 12 are mesoplanets (surface temperature 0–50 degree Celsius) and psychroplanets (surface temperature in the range of -50–0 degree Celsius). This brings us to an interesting proposition. Assuming the clustering/anomaly detection method yields reasonably good results (depends on anomaly score or purity of clusters), can we use these results to cross-validate earlier conclusions drawn from supervised methods described in Sections 4–9? If indeed, we can, then we have a robust list of potentially habitable exoplanets curated from two completely orthogonal approaches- one that does rely on training data and the other which does not! This could have far reaching connotations

Algorithm 5: Algorithm for clustering.

```
input  : An $n$-dimensional array of data $x$ and gbest centroids
output: An array of cluster labels
1  for each gbest_centroids c do
2      for each data x do
3          for each param $i \leftarrow 1$ to n_params do
4              $d \leftarrow d + (x_i - c_i)^2$
5          end
6          distance.append(d)
7      end
8      global_distance.append(distance)
9  end
10 global_distance ← transpose(global_distance)
11 cluster ← argmin(global_distance)
12 return cluster
```

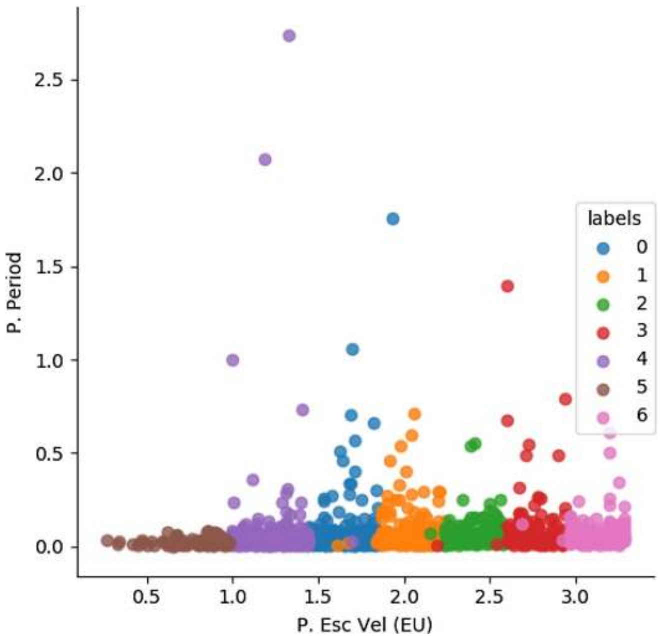


Fig. 22. Planet's escape velocity vs. period.

pending more detailed investigation in to clustering techniques applied on exoplanets (Tab. 12).

13 Discussion and conclusion

As we come to terms with the constantly increasing number of discovered exoplanets and the possibility that stars with planets are a rule rather than an exception, characterizing exoplanets in terms of planetary parameters, types and populations is a reasonably feasible task to accomplish. Once classified with near-perfect accuracy (which we have shown through our methods) we could then associate the classified and potentially habitable planets with indices quantifying habitability

Table 12. Habitable Planets in Earth's Cluster for PSO based clustering.

Planet names
GJ 273 b
GJ 667 C e
GJ 667 C f
GJ 3323 b
K2-72 e
Kepler-62 f
Kepler-438 b
Kepler-442 b
Kepler-1229 b
Kepler-1652 b
Proxima Cen b
Ross 128 b

potential. This is also important in understanding the formation pathways of exoplanets. However, complete appraisal of the potential habitability needs the knowledge of multiple planetary parameters which, in turn, requires hours of expensive telescope time. Automated classification of exoplanets, thus, becomes critical as it helps prioritising the planets to look at and eventually help build a quick screening tool for evaluating habitability perspectives from observed properties. The automated process is not intended to substitute the painstaking process of science that demands observation and analysis time. But, the classification methods developed are a much needed follow-up to look for the biosignatures and complement the spectroscopic studies. These biosignatures are atmospheric gases that only living organisms produce in abundance. It can be oxygen, ozone, methane, carbon dioxide or, better, their combinations [3,4]. Plenty of upcoming space missions, PLATO, Euclid and JWST dedicated to the search of life in the Universe are planned. In order to facilitate the search in a less expensive manner, we need to make a list of potential/preferred candidates obtained by the methods of exoplanet classification detailed in the manuscript. This ensures the quest to complete within finite time. It is estimated that one in five solar-type stars and approximately half of all M-dwarf stars may host an Earth-like planet in the habitable zone (HZ). As Borucki et al. [5], Batalha et al. [6], and Petigura et al. [7] pointed out, evidence from Kepler's data shows that there could be as many as 40 billion such planets in our Galaxy alone. However, the sheer volume of task that demands parsing through information containing pentabytes of data may be a luxury we should try to avoid. Obtaining the spectra of a small planet around a small star is difficult, and even a large-scale expensive space mission (such as e.g. JWST) may be able to observe only about a hundred stars over its lifetime [74]. Thus, the task of shortlisting a reasonable list of potentially habitable exoplanets is of paramount importance. The grand ideas of classifying exoplanets, computing habitability scores/indices and automatic grouping of the exoplanets need to converge at some level. The different components are significant to investigate without having one to supersede the other. This brings us to the importance behind development of activation functions (SBAF family) as a key tool in classification.

The motivation of SBAF is derived from the idea of using $kx^{\alpha}(1-x)^{1-\alpha}$ to maximize the width of the two separating hyperplanes (similar to separating hyperplanes in the SVM as the kernel has a global optima) when $0 \leq \alpha \leq 1$. This is equivalent to the CDHS formulation when CD-HPF is written as $y = kx^{\alpha}(1-x)^{\beta}$ where $\alpha + \beta = 1, 0 \leq \alpha \leq 1, 0 \leq \beta \leq 1$, k is suitably assumed to be 1 (CRS condition), and the representation ensures global maxima (maximum width of the separating hyperplanes) under such constraints [12,25]. The new activation function to be used for training a neural network for habitability classification boasts of an optima. Evidently, from the graphical simulations presented earlier, we observe less flattening of the function and therefore the formulation is able to tackle local oscillations more easily as compared to the more generally used sigmoid function. Moreover, since $0 \leq \alpha \leq 1, 0 \leq x \leq 1, 0 \leq 1 - x \leq 1$, the variable term in the denominator of SBAF, $kx^{\alpha}(1-x)^{1-\alpha}$ may be approximated to a first order polynomial. This may help us in circumventing expensive floating point operations without compromising the precision. We have seen evidence of these claims, theoretically and from implementation point of view, in the preceeding sections.

Habitability classification is a complex task. Even though the literature is replete with rich and sophisticated methods using both supervised [75] and unsupervised learning methods, the soft margin between classes, namely psychroplanet and mesoplanet makes the task of discrimination incredibly difficult. A sequence of recent explorations by Saha et al. [25] expanding previous work by Bora et al. [12] on using Machine Learning algorithm to construct and test planetary habitability functions with exoplanet data raises important questions. The 2018 paper [25] analyzed the elasticity of the Cobb–Douglas Habitability Score (CDHS) and compared its performance with other machine learning algorithms. They demonstrated the robustness of their methods to identify potentially habitable planets [76] from exoplanet data set. Given our little knowledge on exoplanets and habitability, these results and methods provide one important step toward automatically identifying objects of interest from large data sets by future ground and space observatories. The variable term in SBAF, $kx^{\alpha}(1-x)^{1-\alpha}$ is inspired from a history of modeling such terms as production functions and exploiting optimization principles in production economics [61,77–79]. Complexities/bias in data may often necessitate devising classification methods to mitigate class imbalance, Mohanchandra et al. [80] to improve upon the original method, Vapnik and Chervonenkis [81], Corinna and Vladimir [82] or manipulate confidence intervals [83]. However, these improvisations led the authors to believe that, a general framework to train in forward and backward pass may turn out to be efficient. This is the primary reason to design a neural network with a novel activation function. We used the architecture to discriminate exoplanetary habitability [9,10,44,45,84,85].

We had to consider the ramifications of the classification technique in astronomy. Hence, we try to classify exoplanets based on *one feature of the exoplanet at a time along with multiple features of the parent star.* Note, we did not consider surface temperature, which is hard marker. An example of this is as follows:

1. **Attributes of a sample planet**:
   – Radius only
   with attributes of the Parent Star such as Mass, Radius, Effective temperature, Luminosity, Inner edge of star's habitable zone and Outer edge of star's habitable zone with
2. **Class attributes**:
   – Thermal habitability classification label of the planet.

Similarly, we present results of classification when we include the exoplanet's mass, instead of the radius; and when we include the exoplanet's minimum mass instead of the radius. Reiterating, we use only one planetary [86] attribute in a classification run.

Machine classification on habitability is a very recent area. Therefore, the motivation for contemplating such a task is beyond doubt. However, instead of using "black-box" methods for classification, we embarked upon understanding activation functions and their role in Artificial Neural Net based classification. Theoretically, there is evidence of optima and therefore absence of local oscillations. This is significant and helps classification efficacy, for certain. In comparison to gradient boosted classification of exoplanets [17,25], our method achieved more accuracy, a near perfect classification. This is encouraging for future explorations into this activation function, including studying the applicability of Q-deformation and maximum entropy principles. Even without habitability classification or absence of any motivation, further study of the activation function seems promising.

Our focus has shifted from presenting and compiling the accuracy of various machine learning and data balancing methods to developing a system for classification that has practical applications and could be used in the real world. The accuracy scores that we have been able to accomplish show that with a reasonably high accuracy, the classification of the exoplanets is being done correctly. The performance of the proposed activation functions on pruned features is simply remarkable for two reasons. The first being, the pruned feature set does not contain features which account for hard markers. This makes the job hard since one would expect a rapid degradation in performance when the hard markers such as surface temperature and all its related features are removed from the feature set before classification [50]. We note, when an exoplanet is discovered, surface temperature is one of the features Astrophysicists use to label it. If surface temperature cannot be measured, it is estimated. Even if the surface temperature cannot be measured, our activation functions make strong enough classifiers to predict labels of exoplanet samples, dispensing away the need for estimating it. This implies that whenever an exoplanet is newly discovered, their thermal habitability classes can be estimated using our approach. This significant information could be useful for other missions based on methods that would try following up the initial observation. Hence, samples that are interesting from a habitability point of view could gain some traction quite early.

Habitability indices need to reconcile with the machine learning methods that have been used to automatically classify exoplanets. Such cross-matching may dispel doubts about relying too much on one number (ESI/CDHS/PHI type) and help make our estimation efforts more robust. The purpose of machine classification of exoplanets is to rationalize the indexing to some extent and investigate if there are conflicting results. It is not easy to discover meeting ground of two fundamentally different approaches (one of which is CDHS based indexing of exoplanets) that lead to a similar conclusion about an exoplanet. While habitability indices provides a numerical indicator, machine classification bolsters the proposition by telling us automatically which class of habitability an exoplanet belongs to. Bora et al. explain that a CDHS close to 1 indicates a greater chance of habitability. The performance of machine classification is evaluated by class-wise accuracy. The accuracies achieved are remarkably high, and at the same time, it is observed that the values of the CDHS for the sample of potentially habitable exoplanets which are considered are also close to 1. Therefore, the computational approaches map Earth similarity to classification based indicator of habitability. Imagine, an intersection of these results with the findings from an unsupervised grouping of exoplanets (Sect. 12), without relying on class labels from the catalog! This is remarkably non-trivial.

Future work could focus on using adaptive learning rates [87] by fixing Lipschitz loss functions. This may help us investigate if faster convergence is achieved helping us fulfill the larger goal of parsimonious computing.

Funding: This work was supported by the Science and Engineering Research Board (SERB)-Department of Science and Technology (DST), Government of India (project reference number SERB-EMR/2016/005687). The funding source was not involved in the study design, writing of the report, or in the decision to submit this article for publication. NN would like to thank Tata Trusts for financial support.

Author contribution statement

Snehanshu Saha authored the activation functions derived from the first principles and showed several results pertaining to the origin and behavior of the activation functions. HE authored three sections in the main text and a major part of the appendices. Nithin Nagaraj was responsible for interpreting and exploiting chaos theory to explain the functional behavior of the activation. He also proved a couple of results regarding regression under uncertainty and Universal Approximation. Archana Mathur implemented the rigorous, bare-bones experimentation of the activation functions and along with Rahul Yedida, built the Symnet pipeline which runs the activations in the neural networks. Sneha H.R did some background empirical investigation on classical machine learning algorithms for comparison and helped in implementing unsupervised method for cross-matching exoplanets.

Appendix A: What is classification?

Classification is segregation of incoming data into 2 or more groups. The majority of this discussion will proceed only about *binary classification* – when there are only two classes. It is easy to adapt a binary classification algorithm to multiple classes.

Let y be the output variable; in binary classification, $y \in \{0, 1\}$. For classification, we set our hypothesis function to

$$h(x) = g\left(\Theta^T x\right)$$

where g is some function. A very popular choice for this function is the *sigmoid function*, which is also called the *logistic function*. The logistic function is defined as:

$$g(z) = \frac{1}{1 + e^{-z}}\cdot$$

The logistic function is bounded between 0 and 1. This makes it very convenient to use it for classification, because if the output of our hypothesis function is above 0.5, we predict 1; otherwise, we predict 0.

We start with an initial guess for Θ, and tweak it so that the cost function $J(\Theta)$ is minimum.

A.1 Cost function and gradient descent

Our hypothesis function $h(x)$ is non-linear, so the least-squares cost function becomes non-convex. Instead, we derive the maximum likelihood estimates.

We have,

$$P(y = 1|x) = h(x)$$

and thus,

$$P(y = 0|x) = 1 - h(x).$$

Given this, we can combine the above to obtain:

$$P(y|x) = h(x)^y \left(1 - h(x)\right)^{1-y}.$$

The likelihood function is then given by,

$$L(\Theta) = P(y|x;\Theta) = \prod_i P\left(y^{(i)}|x^{(i)};\Theta\right).$$

We instead look at the log-likelihood function:

$$l(\Theta) = \sum_{i=1}^{m} y^{(i)} \log\left(h\left(x^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h\left(x^{(i)}\right)\right).$$

The principle of maximum likelihood then states that we should choose Θ so that the likelihood (or in our case, the log-likelihood) is the maximum. This leads to our cost function:

$$J(\Theta) = \frac{-1}{m}\sum_{i=1}^{m} y^{(i)} \log\left(h\left(x^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h\left(x^{(i)}\right)\right).$$

This function is now convex, and can be optimized using gradient descent. To find the derivative, we first find the derivative of the sigmoid function.

$$\begin{aligned}
\frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right) &= \frac{-\left(-e^{-z}\right)}{\left(1+e^{-z}\right)^2} \\
&= \frac{e^{-z}}{\left(1+e^{-z}\right)^2} \\
&= \frac{1}{1+e^{-z}}\cdot\frac{e^{-z}}{1+e^{-z}} \\
&= g(z)\left(1-\frac{1}{1+e^{-z}}\right) \\
&= g(z)\left(1-g(z)\right).
\end{aligned}$$

Then,

$$\begin{aligned}
\frac{\partial}{\partial\theta_j}J(\Theta) &= \frac{\partial}{\partial\theta_j}\left(\frac{-1}{m}\sum_{i=1}^{m} y^{(i)}\log\left(h\left(x^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h\left(x^{(i)}\right)\right)\right) \\
&= \frac{-1}{m}\sum_{i=1}^{m}\left(\frac{y^{(i)}}{h\left(x^{(i)}\right)}h'\left(x^{(i)}\right) + \frac{1-y^{(i)}}{1-h\left(x^{(i)}\right)}\left(-h'\left(x^{(i)}\right)\right)\right) \\
&= \frac{-1}{m}\sum_{i=1}^{m}\left(\frac{y^{(i)}}{h\left(x^{(i)}\right)}h\left(x^{(i)}\right)\left(1-h\left(x^{(i)}\right)\right)\left(x_j^{(i)}\right)\right. \\
&\quad\left. - \frac{1-y^{(i)}}{1-h\left(x^{(i)}\right)}h\left(x^{(i)}\right)\left(1-h\left(x^{(i)}\right)\right)\left(x_j^{(i)}\right)\right) \\
&= \frac{-1}{m}\sum_{i=1}^{m}\left(y^{(i)}\left(1-h\left(x^{(i)}\right)\right)\left(x_j^{(i)}\right) - \left(1-y^{(i)}\right)h\left(x^{(i)}\right)\left(x_j^{(i)}\right)\right) \\
&= \frac{-1}{m}\sum_{i=1}^{m}\left(y^{(i)}\left(1-h\left(x^{(i)}\right)\right) - \left(1-y^{(i)}\right)h\left(x^{(i)}\right)\right)x_j^{(i)} \\
&= \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - y^{(i)}h\left(x^{(i)}\right) - h\left(x^{(i)}\right) + y^{(i)}h\left(x^{(i)}\right)\right)x_j^{(i)}
\end{aligned}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}.$$

A.2 The Newton–Raphson method

Suppose we have a function $f : \mathbb{R} \mapsto \mathbb{R}$. Suppose we are interested in finding a value x such that $f(x) = 0$. We start with some initial guess, say x_0. For the Newton–Raphson method, the subscript indicates which *iteration* we are currently on. The Newton–Raphson method performs the following update:

$$x = x - \frac{f(x)}{f'(x)}.$$

We can use this to maximize or minimize a function. We note that at the maxima or minima of a function, $f'(x) = 0$. Thus, we can use Newton's method in the same way as described above, using $f'(x)$ instead. We can use this to maximize our log-likelihood function, and thus our update rule is

$$\Theta = \Theta - \frac{l'(\Theta)}{l''(\Theta)}.$$

We note that for logistic regression, Θ is a vector and not a real number, so we modify the equation accordingly. We define $\nabla_\Theta J(\Theta)$ as the derivative of the cost function with respect to Θ, where Θ is a vector. For the denominator, we need something to represent the second derivative of the cost function. This is the *Hessian*. Each entry of the Hessian is defined as

$$H_{ij} = \frac{\partial^2 l(\Theta)}{\partial\theta_i \partial\theta_j}.$$

Since we need the double derivative in the denominator, and division is not defined for matrices, we use the inverse, to get the final update rule (here, Θ is a vector).

$$\Theta = \Theta - H^{-1}\nabla_\Theta l(\Theta).$$

Here, H is of dimensions $(n+1, n+1)$. While Newton's method takes fewer steps to find the minimum, it does require finding and then inverting the Hessian, and then multiplying with another matrix. This can be very expensive for large values of n. When we use the Newton–Raphson method instead of gradient descent in logistic regression, it is called *Fisher scoring*.

A.3 Softmax regression

Softmax regression is based on the multinomial distribution.

A.3.1 Exponential family distributions

There is a class of distributions called *exponential family distributions*. A distribution belongs to this class of distributions if it can be written in the form[6]:

$$P(y;\eta) = b(y)\exp\left(\eta^T T(y) - a(\eta)\right).$$

[6] CS229 taught by Andrew Ng at Stanford University, https://see.stanford.edu/course/cs229

- η is called the *natural parameter*. When proving that a certain distribution (such as the Gaussian) belongs to the exponential family, it becomes convenient to change the parameters of the original distribution to new parameters. The new natural parameters are different for different distributions. In the Gaussian example, the original parameters were μ and σ^2, but the natural parameters are $\frac{\mu}{\sigma^2}$ and $\frac{-1}{2\sigma^2}$.
- $T(y)$ is the *sufficient statistic*. A sufficient statistic of a *sample* of data about a parameter is a *statistic* that provides as much information as we can possibly get about that parameter from that data. For example, given a sample of data distributed according to the Gaussian, it can be proved using maximum likelihood estimation that the sample mean is the best estimate of the population mean. Thus, for a Gaussian distribution, the sample mean is the sufficient statistic of the population mean.
- $a(\eta)$ is called the *log partition function*. The quantity $e^{-a(\eta)}$ normalizes the distribution so that the area under the curve is 1. This has the additional property that its first derivative yields the mean, and its second derivative yields the variance.

Many common distributions such as the Binomial, Gaussian, Weibull, and Poisson are examples of distributions that belong to the exponential family. Pertinent to our discussion, the *multinomial distribution*, which is an extension of the Bernoulli distribution, also belongs to this class.

A.3.2 Generalized linear models (GLMs)

Generally, we can have a model that assumes that the output variable follows a distribution that belongs to the exponential family of distributions. Such a model is called a *generalized linear model*. Apart from the distribution of the output variable belonging to the exponential family, GLMs have some other assumptions:

- Given input x, we would like to predict $\mathbb{E}[T(y)|x;\Theta]$.
- The natural parameter $\eta = \Theta^T x$. If, as in the Gaussian example, η is a vector, then we have $\eta_i = \Theta_i^T x$.

A.3.3 Deriving the softmax regression model

We work towards a (generalized linear) model for multi-class classification. Since we have, in general, k classes, the predictor variable follows a *multinomial* distribution, that is, $y \in \{1, 2, \ldots, k\}$.

Our $k-1$ parameters are $\phi_1, \phi_2, \ldots, \phi_{k-1}$, and our kth "parameter" is

$$\phi_k = 1 - (\phi_1 + \phi_2 + \ldots + \phi_{k-1}).$$

Thus, we have $k-1$ parameters. Now we need an exponential family distribution. We work on $T(y)$, and define the *vectors* $T(y)$ as follows:

$$T(1) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad T(2) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \ldots T(k-1) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \quad T(k) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Since $T(y)$ is a vector, we denote the ith element by $T(y)_i$. With this framework in place, we can prove that the multinomial distribution is a member of the exponential family of distributions.

$$\begin{aligned}
P(y;\phi) &= \phi_1^{T(y)_1}\phi_2^{T(y)_2}\dots\phi_k^{T(y)_k}\\
&= \phi_1^{T(y)_1}\phi_2^{T(y)_2}\dots\phi_k^{1-\sum_{i=1}^{k-1}T(y)_i}\\
&= \exp\left(T(y)_1\log(\phi_1)+T(y)_2\log(\phi_2)+\ldots+\left(1-\sum_{i=1}^{k-1}T(y)_i\right)\log(\phi_k\right)\\
&= \exp\left(T(y)_1\log\left(\frac{\phi_1}{\phi_k}\right)+T(y)_2\log\left(\frac{\phi_2}{\phi_k}\right)+\ldots+T(y)_{k-1}\log\left(\frac{\phi_{k-1}}{\phi_k}\right)+\log(\phi_k)\right).
\end{aligned}$$

This is in the form of an exponential family distribution, with

$$\begin{aligned}
b(y) &= 1\\
a(\eta) &= -\log(\phi_k)\\
\eta &= \begin{bmatrix}\log\left(\frac{\phi_1}{\phi_k}\right)\\ \log\left(\frac{\phi_2}{\phi_k}\right)\\ \vdots\\ \log\left(\frac{\phi_{k-1}}{\phi_k}\right)\end{bmatrix}.
\end{aligned}$$

Now, note that $\eta_i = \log\left(\frac{\phi_i}{\phi_k}\right)$. For convenience, we define $\eta_k = \log\left(\frac{\phi_k}{\phi_k}\right) = \log 1 = 0$.

We define $\Theta_k = 0$ so that $\eta_k = \log 1 = 0$.

$$\begin{aligned}
e^{\eta_i} &= \frac{\phi_i}{\phi_k}\\
\phi_k e^{\eta_i} &= \phi_i\\
\phi_k\sum_{i=1}^{k} e^{\eta_i} &= \sum_{i=1}^{k}\phi_i = 1.
\end{aligned}$$

This yields,

$$\phi_k = \frac{1}{\sum_{i=1}^{k} e^{\eta_i}}.$$

And therefore,

$$\phi_i = \frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}}.$$

Hence, given a vector η with k elements, we have a function that considers this vector as input and gives us *another vector*, ϕ, also with k values, defined by the above equation, subject to the conditions

$$\begin{aligned}
\sum_{i=1}^{k}\phi_i &= 1\\
\phi_i &> 0.
\end{aligned}$$

This function is a generalization of the logistic function, and is called the *softmax function.* Finally, we use the last assumption, which we restate here:

$$\eta_i = \Theta_i^T x \qquad \forall i = 1, 2, \ldots, k-1.$$

Each Θ_i is a vector of $(n+1)$ dimensions. And thus, our final sof tmax regression model is

$$P(y = i|x; \Theta) = \frac{e^{\Theta_i^T x}}{\sum_{j=1}^{k} e^{\Theta_j^T x}}.$$

The model will output a vector:

$$\begin{aligned} h(x) &= \mathbb{E}[T(y)|x; \Theta] \\ &= \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{k-1} \end{bmatrix} \\ &= \begin{bmatrix} P(y = 1|x; \Theta) \\ P(y = 2|x; \Theta) \\ \vdots \\ P(y = k-1|x; \Theta) \end{bmatrix}. \end{aligned}$$

Notice that the above only outputs a $(k-1)$ dimension vector. We can find

$$P(y = k|x; \Theta) = \phi_k = 1 - \sum_{i=1}^{k-1} \phi_i.$$

A.3.4 Finding the optimal parameter values

To find the optimal values of Θ to fit the data, we use maximum likelihood estimation. We start with the log-likelihood function.

$$\begin{aligned} \ell(\Theta) &= \sum_{i=1}^{m} \log P\left(y^{(i)}|x^{(i)}; \Theta\right) \\ &= \sum_{i=1}^{m} \log \prod_{j=1}^{k} \left(\frac{e^{\Theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\Theta_l^T x^{(i)}}} \right)^{T(y^{(i)})_j} \\ &= \sum_{i=1}^{m} \sum_{j=1}^{k} \log \left(\frac{e^{\Theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\Theta_l^T x^{(i)}}} \right)^{T(y^{(i)})_j}. \end{aligned}$$

To find the derivative of this, we first find the derivative of the softmax function.

$$\frac{\partial}{\partial \Theta_p} \left(\frac{e^{\Theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\Theta_l^T x^{(i)}}} \right) = \frac{\partial}{\partial \Theta_p^T x^{(i)}} \left(\frac{e^{\Theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\Theta_l^T x^{(i)}}} \right) \cdot \frac{\partial}{\partial \Theta_p} \left(\Theta_p^T x^{(i)} \right)$$

$$= \frac{\frac{\partial}{\partial \Theta_p^T x^{(i)}}\left(e^{\Theta_j^T x^{(i)}}\right)\sum_{l=1}^k e^{\Theta_l^T x^{(i)}} - e^{\Theta_j^T x^{(i)}} \cdot \frac{\partial}{\partial \Theta_p^T x^{(i)}}\left(\sum_{l=1}^k e^{\Theta_j^T x^{(i)}}\right)}{\left(\sum_{l=1}^k e^{\Theta_l^T x^{(i)}}\right)^2}$$
$$\cdot \frac{\partial}{\partial \Theta_p^T}\left(\Theta_p^T x^{(i)}\right)$$
$$= \frac{[p=j]e^{\Theta_j^T x^{(i)}}\sum_{l=1}^k e^{\Theta_l^T x^{(i)}} - e^{\Theta_j^T x^{(i)}} \cdot e^{\Theta_p^T x^{(i)}}}{\left(\sum_{l=1}^k e^{\Theta_l^T x^{(i)}}\right)^2} \cdot x^{(i)}$$
$$= \left(\frac{[p=j]e^{\Theta_j^T x^{(i)}}}{\sum_{l=1}^k e^{\Theta_l^T x^{(i)}}} - \frac{e^{\Theta_j^T x^{(i)}}}{\sum_{l=1}^k e^{\Theta_l^T x^{(i)}}} \cdot \frac{e^{\Theta_p^T x^{(i)}}}{\sum_{l=1}^k e^{\Theta_l^T x^{(i)}}}\right) \cdot x^{(i)}$$
$$= ([p=j]s_j - s_j s_p) \cdot x^{(i)}$$
$$= s_j([p=j] - s_p)x^{(i)}.$$

Continuing,

$$\begin{aligned}\frac{\partial}{\partial \Theta_p}\left(\sum_{j=1}^k \log s_j\right) &= \sum_{j=1}^k \frac{\partial}{\partial \Theta_p}(\log s_j)\\ &= \sum_{j=1}^k \frac{1}{s_j} \cdot \frac{\partial s_j}{\partial \Theta_p}\\ &= \sum_{j=1}^k \frac{1}{s_j} \cdot s_j([p=j] - s_p)x^{(i)}\\ &= \sum_{j=1}^k ([p=j] - s_p)x^{(i)}.\end{aligned}$$

Next,

$$\begin{aligned}\frac{\partial}{\partial \Theta_p}\left(\sum_{j=1}^k \log(s_j)^{T(y^{(i)})_j}\right) &= \frac{\partial}{\partial \Theta_p}\sum_{j=1}^k T(y^{(i)})_j \log s_j\\ &= T(y^{(i)})_j \sum_{j=1}^k ([p=j] - s_p)x^{(i)}.\end{aligned}$$

At this point, we note that $T(y^{(i)})_j = 1$ only when $y^{(i)} = j$, and so we can use the Iverson notation. This will be useful for us later:

$$\frac{\partial}{\partial \Theta_p}\left(\sum_{j=1}^k \log(s_j)^{T(y^{(i)})_j}\right) = x^{(i)}\left(\sum_{j=1}^k [y^{(i)} = j] \cdot [p=j] - s_p \sum_{j=1}^k [y^{(i)} = j]\right).$$

Simplifying and computing the gradient of the log-likelihood yields[7]:

[7] Ben's answer on Cross Validation, https://stats.stackexchange.com/a/353342/212844

$$\frac{\partial}{\partial \Theta_p}\ell(\Theta) = \frac{\partial}{\partial \Theta_p}\left(\sum_{i=1}^{m}\sum_{j=1}^{k}\log(s_j)^{T(y^{(i)})_j}\right)$$
$$= \sum_{i=1}^{m} x^{(i)}([y^{(i)} = p] - s_p).$$

Subsequently, we use gradient descent to find a minima.

A.3.5 Regularization

The consequence of running gradient descent be that the model will sometimes overfit by trying to fit a curve through noise. Overfitting can be a result of multiple reasons including trying to fit to a model that is too complex, choosing initial parameter values wrong, among other reasons. What is common when models overfit, though, is that the parameters tend to become very large (in magnitude). Note that our model is trying to minimize the cost function. Therefore, an obvious way to minimize this overfitting problem is to include the coefficients (parameters) themselves in the cost function. We use regularization while doing this. Consider a linear regression model, where we add the term $\lambda\sum_{i=0}^{n}|\theta_i|$ to the cost function. Thus, we obtain the following:

$$J(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{i=0}^{n}|\Theta_i|.$$

The λ term controls how much we want to regularize the parameters. The above is called lasso regression. We could also use L_2 regularization. When we do that, the resulting model is called ridge regression. In machine learning, the term weight decay has caught on better, because these regularization methods effectively cause the parameter values (or "weights") to go down (or "decay"), resulting in a model that does not overfit. Concretely, the cost function is

$$J(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{i=0}^{n}\Theta_i^2.$$

We added the factor 2 in the denominator for the same mathematical convenience as for the ordinary least-squares cost function.

A.4 Likelihood and posterior: Two important tools in machine inference and classification

Maximum Likelihood Estimation, as seen in regression, and Maximum-A-Posterior are extremely useful tools in pointwise estimates and classification of objects. Let $D = x_1, x_2, \ldots .x_n$ be the training data and L be the likelihood function, assumed to be Gaussian. Then,

$$L(\Theta|D) = \prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \tag{A.1}$$
$$L = (\mu, \sigma|x_1, x_2, \ldots .x_n) = L(\mu, \sigma|x_1)\, L(\mu, \sigma|x_2) \ldots L(\mu, \sigma|x_n) \tag{A.2}$$

where $L = (\mu, \sigma|x_1, x_2, \ldots .x_n)$ is $L(\Theta|D)$.

MLE estimation can be computed by taking derivative with respect to μ (σ constant) and with respect to σ (μ constant). Taking log on both sides of equation (23)

$$\begin{aligned}
\ln L(\Theta|D) &= \ln\left(\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}\right)\\
&= \ln\left(\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x_1-\mu)^2}{2\sigma^2}}\right)+\ldots+\ln\left(\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x_n-\mu)^2}{2\sigma^2}}\right)\\
&= -\frac{1}{2}\ln\left(2\pi\sigma^2\right)-\frac{(x_1-\mu)^2}{2\sigma^2}\\
&= -\frac{1}{2}\ln(2\pi)-\ln(\sigma)-\frac{(x_1-\mu)^2}{2\sigma^2}+\ldots\\
&= -\frac{n}{2}\ln(2\pi)-nln(\sigma)-\sum_{i=1}^{n}\frac{(x_1-\mu)^2}{2\sigma^2}.
\end{aligned}$$

Computing $\frac{\partial L}{\partial \mu}$ of the above equation and equating it with 0

$$\begin{aligned}
\frac{\partial L}{\partial \mu} &= 0-0+\sum_{i=1}^{n}\frac{x_i-\mu}{\sigma^2}\\
0 &= \sum_{i=1}^{n}\frac{x_i-\mu}{\sigma^2}\\
\mu &= \frac{\sum_{i=1}^{n}x_i}{n}.
\end{aligned}$$

μ is equal to $\bar{x}$ which is the mean of the sample. This implies sample mean is a good enough estimate of the population mean. Finding the derivative with respect to σ and equating with 0, we obtain

$$\begin{aligned}
\frac{\partial L}{\partial \sigma} &= -\frac{n}{\sigma}+\sum_{i=1}^{n}(x_i-\mu)^2\sigma^{-3}\\
0 &= -\frac{n}{\sigma}+\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{\sigma^3}\\
\sigma^2 &= \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n}\\
\sigma &= \sqrt{\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n}}.
\end{aligned}$$

This implies that sample standard deviation is a good estimate of population standard deviation.

Maximum a posterior Probability (MAP). Given, $D = \{x_1, x_2 \ldots, x_n \mid x_i \epsilon R^n\}$; assume a joint distribution $p(D, \theta)$ where θ is random variable. The goal is to choose good θ for D,

$$\begin{aligned}
\theta_{\text{MAP}} &= \text{argmax } p(\theta \mid D)\\
\theta_{\text{MLE}} &= \text{argmax } p(D \mid \theta).
\end{aligned}$$

Pros:

(i) Easy to compute and interpret; $p(D,\theta) = p(\theta \mid D)p(\theta)$ where $p(\theta \mid D)$ is the Maximum Likelihood estimation(MLE) and $p(\theta)$ is the prior distribution on the data. MAP interpolates between MLE and prior probability.

(ii) MAP avoids overfitting. MLE often overfits the data.

Cons:

(i) It is a points estimate, which means there is no representation of uncertainty in θ.

(ii) We must assume prior probability on θ.

MAP for a mean of a univariate Gaussian. We assume data D, such that $D = \{x_1, x_2 \ldots, x_n \mid x_i \epsilon R^n\}$; where θ is random variable of normal distribution with mean μ and variance as 1. Here, $x_1, x_2 \ldots, x_n$ are conditionally independent given θ. This means that:

$$p(x_1, x_2 \ldots .x_n \mid \Theta) = \prod_{i=1}^{n} p\left(x_i \mid \Theta\right).$$

MAP estimate is given by:

$$\theta_{\text{MAP}} = \text{argmax}\ p(\theta \mid D) = \text{argmax}\ \frac{p\left(D \mid \Theta\right) p\left(\Theta\right)}{p\left(D\right)}. \tag{A.3}$$

The above equation uses the Bayes Rule given by

$$p(\theta \mid D) = \frac{p\left(D \mid \Theta\right) p\left(\Theta\right)}{p\left(D\right)}.$$

Considering just the numerator of the above equation for taking log (since the denominator is a normalizing factor and will be consistent for different posteriors), we get

$$\theta_{\text{MAP}} = \text{argmax}\left(\log\ p\left(D \mid \Theta\right) p\left(\Theta\right)\right)$$
$$\theta_{\text{MAP}} = \text{argmax}\left(\log\ p\left(D \mid \Theta\right)\right) + \left(\log\ p\left(\Theta\right)\right).$$

In order to maximise the above equation, we find the derivative of $(\log\ p\left(D \mid \Theta\right)) + (\log\ p\left(\Theta\right))$ with respect to θ and equate it to zero. The derivative of the first component i.e. $(\log\ p\left(D \mid \Theta\right))$ is given by

$$\frac{\partial \log\ p\left(D \mid \Theta\right)}{\partial \theta} = \frac{\left(\sum_{i=1}^{n} x_i - n\Theta\right)}{\sigma^2}$$

and derivative of the second component $(\log\ p\left(\Theta\right))$ is given by

$$\begin{aligned} \frac{\partial \log\ p\left(\Theta\right)}{\partial \Theta} &= \frac{\partial}{\partial \Theta} \log\left(\frac{e^{-\frac{1}{2}(\Theta-\mu)^2}}{\sqrt{2\pi}}\right) \\ &= \frac{\partial}{\partial \Theta}\left(-\frac{1}{2}\log 2\pi - \frac{1}{2}\left(\theta - \mu\right)^2\right) \\ &= \mu - \Theta. \end{aligned}$$

Hence the derivative of (log $p(D \mid \Theta)) + (\log\ p(\Theta))$ with respect to θ is

$$= \frac{\left(\sum_{i=1}^{n} x_i - n\Theta\right)}{\sigma^2} + \mu - \Theta.$$

Equating it to zero, we obtain the posterior estimate as the following:

$$0 = \frac{\left(\sum_{i=1}^{n} x_i - n\Theta\right)}{\sigma^2} + \mu - \Theta$$

$$\left(\frac{n}{\sigma^2} + 1\right)\Theta = \frac{1}{\sigma^2}\sum_{i=1}^{n} x_i + \mu$$

$$\Theta_{\text{MAP}} = \left(\frac{n}{n+\sigma^2}\right)\bar{x} + \left(\frac{\sigma^2}{n+\sigma^2}\right)\mu.$$

The above equation suggests that Θ_{MAP} is a convex combination of prior/population mean (μ) and sample mean of the data D $\bar{x}$. Based on the above equation, the following points can be deduced,

- Θ_{MAP} is equal to μ when $\sigma^2 = 0$ (NO variance).
- As n and σ^2 vary, we obtain a range of values between sample mean and prior mean.
- When $n = 0$ (NO data), it indicates $\Theta_{\text{MAP}} \sim \mu$.
- When $n \to \infty$, it indicates $\Theta_{\text{MAP}} \sim \bar{x}$. When the data are huge, MAP estimator is nothing but the sample mean.

A.4.1 MAP as an illustration in habitability classification

Consider an example in which a new exoplanet has been discovered and there are two alternative hypothesis in context with its habitability (1) exoplanet is habitable and, (2) the exoplanet is non-habitable. Based on the parameters reported for the exoplanet, there could be two possible outcomes: $\oplus$ (habitable) and $\ominus$ (non-habitable). Based on the prior information available from the catalogues, there are only 0.008 planets which are habitable. Moreover, from the entire population of discovered exoplanets that are habitable, 98% are correctly labeled. Correspondingly, the exoplanets that are confirmed as non-habitable, 97% are correctly labeled. Summarizing and calculating the prior and posterior probabilities:

(1) habitable planet(h) – (+ class), non-habitable planet ($\bar{h}$) – (− class)
(2) $P(h) = 0.008$, $P(\bar{h}) = 0.992$,
(3) $P(\oplus \mid h) = 0.98$, $P(\ominus \mid h) = 0.02$,
(4) $P(\oplus \mid \bar{h}) = 0.03$, $P(\ominus \mid \bar{h}) = 0.97$.

Bayes theorem:

$$P(h \mid D) = \frac{P(D \mid h)\, P(h)}{P(D)}.$$

Now we know that,

$$h_{\text{MAP}} = \text{argmax} P(D \mid h)\, P(h).$$

Suppose for the newly discovered planet, investigation indicates a positive outcome (i.e. planet is confirmed to be habitable). Does that mean that the exoplanet is actually habitable? In order to find this, compute the posterior probability, for each hypothesis in H (h, $\bar{h}$),

$$P(h \mid \oplus) = P(\oplus \mid h)P(h) = 0.98 * 0.008 = 0.00784$$
$$P(\bar{h} \mid \oplus) = P(\oplus \mid \bar{h})P(\bar{h}) = 0.003 * 0.992 = 0.02976$$

where $P(h \mid \oplus)$ shows posterior probability of exoplanet being correctly labeled as habitable and $P(\bar{h} \mid \oplus)$ calculates the probability that the exoplanet is wrongly labeled as habitable (it is actually non-habitable). Once the posterior probabilities of both hypothesis are computed, output the hypothesis h_{MAP} with the highest posterior probability. As per the calculation shown above, $P(\oplus \mid \bar{h})P(\bar{h})$ has larger value (0.02976). Hence, even though the exoplanet is reported as habitable, it is non-habitable.

Note on MLE and MAP: MAP can be interpreted as forming and changing opinions via an evolutionary process. We may have an opinion or belief that decides in forming initial decision until that gets modified by new evidence gathered by interactions or experience (likelihood on data) from data. For example, let us consider the role of swing votes in choosing a candidate during an election. If you are an ideologue, there is no likelihood estimation interacting with your past beliefs to change your posterior prediction. There, the likelihood function plays no role (i.e. no data $\sim \bar{x} = 0$ i.e. $\Theta_{\text{MAP}} \sim \mu$ (prior mean). The different scenarios for using MLE or MAP are summarized below:

- In the absence of prior, MLE is used.
- In case of flat prior, MLE $\sim$ MAP.
- Use MLE if we have an abundance of data.
- In the absence of data, we may just use prior mean.

Parameter Estimation in Regression by MLE. MLE on linear regression: $y = Xb + \varepsilon$ where X is the data matrix and b are the regression coefficients to determine; ε is the Gaussian noise. Need to show that,

$$b = \left(X^T X\right)^{-1} X^T y.$$

$(X^T X)^{-1}$ is pseudo-inverse. Let's look at the general least square optimization problem: minimize the residuals $\sum_{i=1}^{n} \left(y_i - a_0 X_i - a_1\right)^2$. Writing the residuals to n-vector form, we obtain

$$\begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \cdots \\ \epsilon_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{pmatrix} - \begin{bmatrix} 1 & X_1 \\ 1 & X_2 \\ \cdots & \cdots \\ 1 & X_n \end{bmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$$
$$\epsilon = y - Xb.$$

Therefore, the multiple regression problem i.e. least square minimization of the residuals becomes,

$$\left\|\epsilon^2\right\| = \left\|\epsilon^T \epsilon\right\| = (y - Xb)^T (y - Xb).$$

The objective is to minimize $(y - Xb)^T (y - Xb)$ by taking its derivative with respect to b and equating it to 0. Rearranging the terms, we obtain

$$\begin{aligned}
&= (y - Xb)^T (y - Xb)\\
&= \left(-(Xb)^T + y^T\right)(y - Xb)\\
&= -b^T X^T y + b^T X^T X b + y^T y - y^T X b.
\end{aligned}$$

We know that $X^T y = \left(y^T X\right)^T$. Keeping this in the above equation,

$$\begin{aligned}
&= -b^T X^T y + b^T X^T X b + y^T y - b^T \left(y^T X\right)^T\\
&= -b^T X^T y + b^T X^T X b + y^T y - b^T X^T y\\
&= y^T y - 2b^T X^T y + b^T X^T X b.
\end{aligned}$$

Taking its derivative with respect to b,

$$= \frac{\partial}{\partial b}\left(y^T y - 2b^T X^T y + b^T X^T X b\right). \tag{A.4}$$

We know that,

$$\frac{\partial}{\partial b}\left(b^T c b\right) = \left(c + c^T\right) b.$$

Assume $c = X^T X$, which means $c^T = c$ and so, $(c + c^T) = 2c$. Keeping this in above equation,

$$\begin{aligned}
\frac{\partial}{\partial b}\left(b^T c b\right) &= 2cb\\
\frac{\partial}{\partial b}\left(b^T X^T X b\right) &= \left(2X^T X b\right).
\end{aligned}$$

Keeping the value in equation to find derivative,

$$\begin{aligned}
&= \frac{\partial}{\partial b}\left(y^T y - 2b^T X^T y + b^T X^T X b\right)\\
&= \left(-2X^T y + 2X^T X b\right)\\
2X^T X b &= 2X^T y\\
b &= \left(X^T X\right)^{-1} X^T y.
\end{aligned}$$

Regression coefficients, $b = \left(X^T X\right)^{-1} X^T y$, can thus be obtained by MLE.

A.5 Evaluating classification models

Having used a classification model, we also need to know how to evaluate its performance. We begin by stating some definitions. The ground truths of a dataset are the class labels in the dataset. These are the true values of the classes (this also applies to regression: the ground truth is the actual Y value for each training example). Suppose we use a train-test split. We train the model on the training set, and make it predict on the remaining 30% so that we can see how well it performs on data it has never seen before. Given that our model predicts on the test set, the true positives are the examples where the model predicts a positive class, and the ground truth for that example is also positive. False positives are where the model predicts true, but the ground truth is false. False positives in statistics are called Type I errors. True negatives and false negatives (called Type II errors) can be thought about in a similar way.

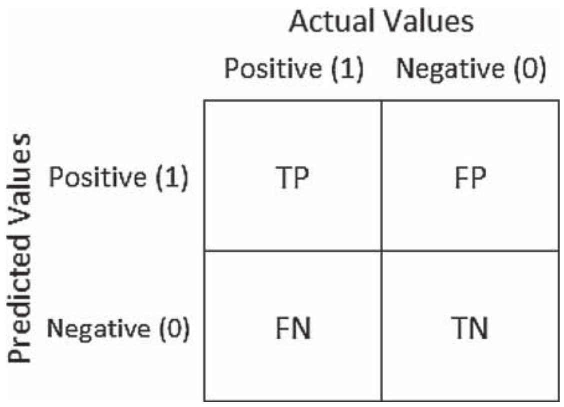


Fig. A.1. A typical Confusion matrix. Source: https://tiny.cc/confusionmatrix.

A.5.1 The confusion matrix and metrics

The confusion matrix is simply a matrix of the counts of everything we defined above. Figure A.1 shows how this is constructed.

Given the confusion matrix, we can now begin to define the metrics to judge our classifiers.

$$\begin{aligned}\text{Accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}\\ \text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}\\ \text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}\\ \text{F1 Score} &= \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.\end{aligned}$$

Accuracy is pretty intuitive, and the others are described below:

- Precision answers the question, "What fraction of positives predicted by the model were right?"
- Recall answers the question, "What fraction of actual positive classes did the model get right?"
- The F1 score is the harmonic mean of the recall and precision. It is a neat way to sum things up.

Another commonly quoted metric is the receiver operating characteristic (ROC) curve. This is a curve drawn with the true positive rate (recall) on the y-axis, and the false positive rate ($\frac{\text{FP}}{\text{FP}+\text{TN}}$) on the x-axis. This is done only with binary classification tasks. Figure A.2 shows an example.

To draw this, recall that the logistic regression model (for binary classification) gives us a probability as an output, rather than a class directly. We use a threshold value (say 0.5) to decide the class. Call this threshold value as T. By varying T systematically, we obtain many values of true positive rate and true negative rate, and we can join all these points to form a smooth curve. This curve is the ROC curve. The line $y = x$ is what we get if the true positive rate was always equal to the false positive rate. We want the ROC curve to be as far from it as possible, towards the top left corner. This is because at the diagonal line, the model is basically just guessing the class. Clearly, the ideal ROC is one that just goes up the y-axis till the top-left corner, and then goes along the top of the graph. This however, is almost never doable, since it implies that the algorithm absolutely never goes wrong at all.

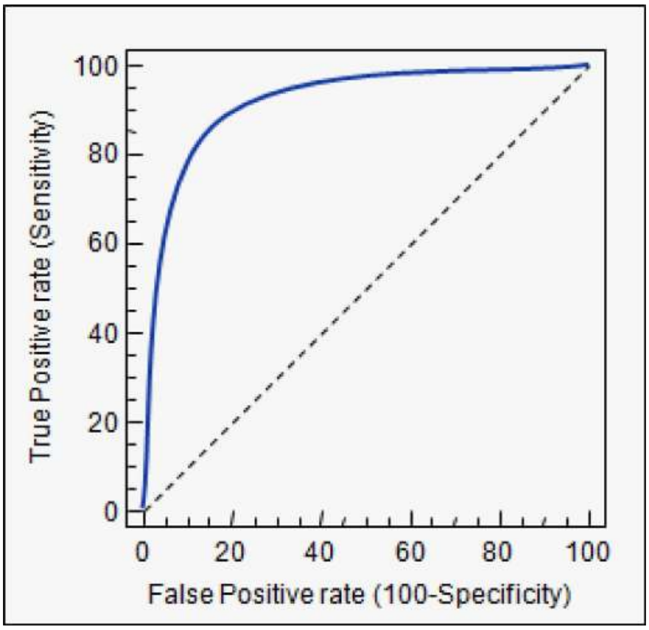


Fig. A.2. The ROC curve. Source: http://qr.ae/TUpjCN.

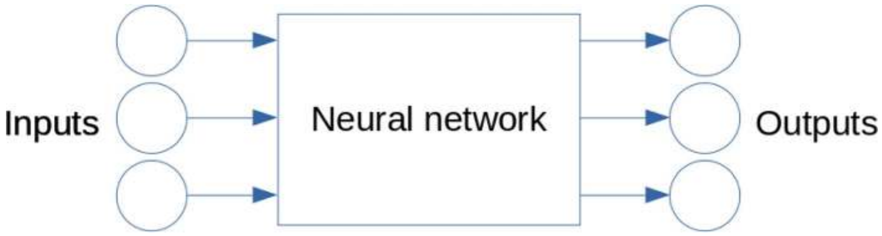
Appendix B: Neural networks: A comprehensive introduction

Neural networks are quickly becoming omnipresent for many tasks, including image recognition, text summarization and synthesis, and speech recognition. There are a couple of reasons why they are become so popular in recent years. First, we have a lot more data these days than earlier, and this means learning algorithms have more to learn from. Secondly, computers are more powerful now, and hardware support in the form of GPUs makes neural networks significantly quicker to train. Finally, a new “activation function” called ReLU (rectified linear unit) made neural networks significantly better at a lot of tasks.

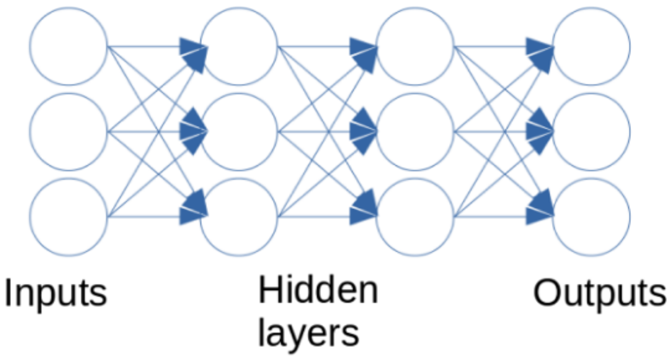
Because of the depth of the field and the pace at which research in “deep learning”, as it is called, is progressing, it is practically impossible to concisely discuss all of neural networks in anything less than a book.

B.1 Introduction

At a very high level, neural networks are a black box. They take in some inputs, and yield outputs that are extraordinarily accurate.



This is a high-level diagram of a neural network. It needs some data to train. Later, we can provide some inputs and expect highly accurate outputs. The circles are typically called *nodes* or *neurons*. The idea is to simulate what goes on in the human brain. Arrows in diagrams like this represent weighted connections. We now unfold that box in the middle.



Inside, you simply have more neurons. These neurons are organized in *layers*, with the connections connecting neurons in one layer to neurons in the next. Images like the one above give the impression that every neuron is necessarily connected to every other. This certainly need not be the case. In fact, we could also have connections jump over layers – we call these skip-connections. However, beyond simply mentioning that, we will not discuss it further.

We have a layer that collects inputs. These input layer nodes do something, and pass the results on to so-called *hidden layers*. Those hidden layers in turn do something, and pass the results forward, until we get outputs. In theory, we could customize what every neuron in every layer does; in practice, that becomes cumbersome, and we only do such customization layer-wise, meaning that all neurons in one layer will perform the same operation.

What does each neuron do? If you see the figure above, each neuron receives several inputs from weighted connections. We specifically used the term weighted connections, because the inputs are not treated equally. A neuron will first add up all the inputs that it gets, but it will perform a weighted sum. After performing a weighted summation, the neuron ends up with a single number. It then computes a function of that number, and the result of that is the neuron's final output – the one that it broadcasts to whatever it happens to be connected to at future layers. Loosely, a neuron does this:

$$f(x_1, x_2, \ldots, x_n) = g(w_1x_1 + w_2x_2 + \ldots + w_nx_n) = g\left(w^T x\right).$$

Those w terms are called the *weights*. It is those weights that we learn using gradient descent. The function g is called the *activation function*. This name is partly historical. In the earlier days of neural networks, this function gave an output of 1 if the weighted sum was higher than a set threshold, and gave output 0 otherwise, and the neuron was "activated" if the weighted sum was above that threshold.

So, given inputs, the input layer neurons will forward the inputs to the first hidden layer, which compute a weighted sum, compute the *activations* (the results of the activation function), and pass these to the second hidden layer. The neurons in this layer will in turn do the same thing, and so on until we get outputs. This process is called *forward propagation*, and is the first step used in gradient descent while training a neural network.

We we will use the following notation. $g(\cdot)$ will still represent the activation function; but since the activation used can be different at each layer, we will be explicit about that, and write $g^{[l]}(\cdot)$ to denote the activation at layer l. We will not actually care about the outputs of individual neurons; rather, we will look at the outputs of an entire layer (which will of course, be a vector). We will represent the weighted sums computed by the neurons at a layer by $z^{[l]}$, and the outputs (the activations) by $a^{[l]}$. The inputs will simply be denoted by the vector x, and to simplify things, we let $a^{[0]} = x$. At each layer, the weights form a matrix, where the first row corresponds to the weights of the outputs from the first neuron, the second row corresponds to the weights of the outputs from the second neuron, and so on. We will represent this by $W^{[l]}$. We will denote the number of layers by L, and the number of neurons at layer l by $n^{[l]}$. The number of layers in the above diagram are three, because the layer of inputs are not counted as an actual layer (since those neurons are not really doing anything).

At a given layer l, the computations performed are:

$$z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}$$
$$a^{[l]} = g(z^{[l]}).$$

This is because $W^{[l]}$ has dimensions $(n^{[l]}, n^{[l-1]})$, and $a^{[l]}$, $z^{[l]}$, and $b^{[l]}$ have the dimensions $(n^{[l]}, 1)$.

Let us revisit to forward propagation now. We first compute weighted sums, now represented as a matrix multiplication, and then compute the activations from those weighted sums. In neural networks, we explicitly represent the constant term – we call this constant term the *bias*. Typically, all the neurons in a layer use the same value of the bias, so $b^{[l]}$ is technically a real number, but because the result of the multiplication $W^{[l]}a^{[l-1]}$ is a vector, we simply create a vector of the same dimensions, where every value is that identical bias.

B.2 Backpropagation

The parameters of a neural network are the weights and the biases.

We cannot directly compute the gradients with respect to the weights in the first few layers. Because of the way that we computed the outputs, left to right, we have to compute the gradients in the reverse order – from right to left – and that gives this step its name. We now discuss how we compute the gradients. Essentially, we simply use the chain rule.

$$\frac{\partial L}{\partial W^{[L]}} = \frac{\partial L}{\partial a^{[L]}}\frac{\partial a^{[L]}}{\partial z^{[L]}}\frac{\partial z^{[L]}}{\partial W^{[L]}}.$$

Similarly, we can continue and compute the gradients with respect to the penultimate layer:

$$\frac{\partial L}{\partial W^{[L-1]}} = \frac{\partial L}{\partial a^{[L]}}\frac{\partial a^{[L]}}{\partial z^{[L]}}\frac{\partial z^{[L]}}{\partial a^{[L-1]}}\frac{\partial a^{[L-1]}}{\partial z^{[L-1]}}\frac{\partial z^{[L-1]}}{\partial W^{[L-1]}}.$$

We simply continue this way till we hit the first layer. Consider the problem of binary classification. We will use the binary cross-entropy loss as before. For now, we assume that every activation function is the sigmoid function.

$$\frac{\partial L}{\partial a^{[L]}} = \frac{\partial}{\partial a^{[L]}}\left(-y\log a^{[L]} - (1-y)\log\left(1 - a^{[L]}\right)\right)$$

$$
\begin{aligned}
&= -\frac{y}{a^{[L]}} + \frac{1-y}{1-a^{[L]}} \\
&= \frac{-y + ya^{[L]} + a^{[L]} - ya^{[L]}}{a^{[L]}\left(1-a^{[L]}\right)} \\
&= \frac{a^{[L]} - y}{a^{[L]}\left(1-a^{[L]}\right)}.
\end{aligned}
$$

That was the first term. The second term is computed as follows

$$
\begin{aligned}
\frac{\partial a^{[L]}}{\partial z^{[L]}} &= \frac{\partial}{\partial z^{[L]}} \frac{1}{1+\exp(-z^{[L]})} \\
&= a^{[L]}\left(1-a^{[L]}\right).
\end{aligned}
$$

Our next term is also easy to compute:

$$
\begin{aligned}
\frac{\partial z^{[L]}}{\partial a^{[L-1]}} &= \frac{\partial}{\partial a^{[L-1]}} \left(W^{[L]}a^{[L-1]} + b^{[L]}\right) \\
&= W^{[L]}.
\end{aligned}
$$

The next term is the same as for the last layer, so we will not repeat that calculation. Finally,

$$
\begin{aligned}
\frac{\partial z^{[L-1]}}{\partial W^{[L-1]}} &= \frac{\partial}{\partial W^{[L-1]}} \left(W^{[L-1]}a^{[L-2]} + b^{[L-1]}\right) \\
&= a^{[L-2]T}.
\end{aligned}
$$

From these, we get

$$\frac{\partial L}{\partial W^{[L]}} = \left(a^{[L]} - y\right) a^{[L-1]T}.$$

Moving on, we have

$$\frac{\partial L}{\partial b^{[L]}} = \left(a^{[L]} - y\right)$$

and

$$\frac{\partial L}{\partial W^{[L-1]}} = \left(a^{[L]} - y\right) W^{[L]} a^{[L-1]} \left(1 - a^{[L-1]}\right) a^{[L-2]T}.$$

It turns out that the right form for this is:

$$\frac{\partial L}{\partial W^{[L-1]}} = \left(W^{[L]T}\left(a^{[L]} - y\right)\right) \times \left(a^{[L-1]}\left(1 - a^{[L-1]}\right)\right) a^{[L-2]T}.$$

The $\times$ symbol denotes element-wise multiplication, not a matrix multiplication. More generally, we have

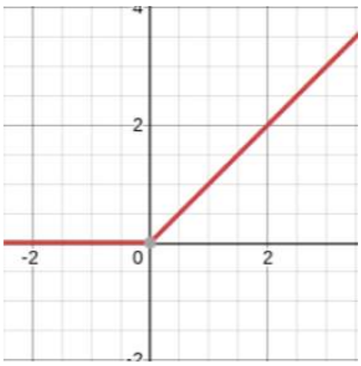
$$
\begin{aligned}
\frac{\partial L}{\partial z^{[l]}} &= \frac{\partial L}{\partial a^{[l]}} \times g^{[l]\prime}(z^{[l]}) \\
\frac{\partial L}{\partial W^{[l]}} &= \frac{\partial L}{\partial z^{[l]}} a^{[l-1]T} \\
\frac{\partial L}{\partial a^{[l-1]}} &= W^{[l]T} \frac{\partial L}{\partial z^{[l]}} \\
\frac{\partial L}{\partial b^{[l]}} &= \frac{\partial L}{\partial z^{[l]}}.
\end{aligned}
$$

B.3 Activation functions

In practice, modern neural networks do not use sigmoid in all the layers – typically, only the last layer uses sigmoid activations, and only when the problem is binary classification. Similarly, when our problem is multi-class classification, we use the softmax activation function.

The ReLU is defined as

$$g(x) = \max(0, x).$$



We cannot just use a simple linear activation function, $g(x) = x$. After two layers,

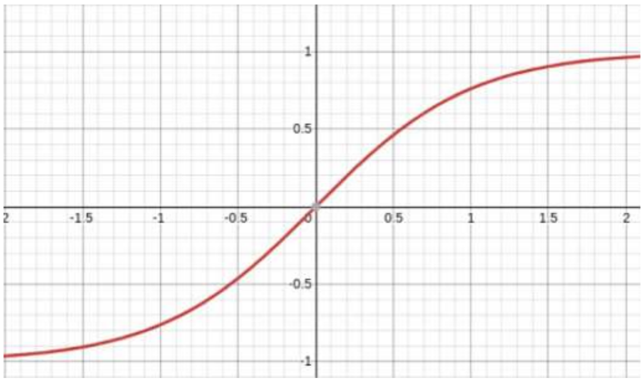
$$\begin{aligned} z^{[1]} &= W^{[1]}x + b^{[1]} \\ a^{[1]} &= W^{[1]}x + b^{[1]} \\ z^{[2]} &= W^{[2]}a^{[1]} + b^{[2]} \\ a^{[2]} &= W^{[2]}a^{[1]} + b^{[2]} \\ &= W^{[2]}\left(W^{[1]}x + b^{[1]}\right) + b^{[2]} \\ &= W^{[2]}W^{[1]}x + W^{[2]}b^{[1]} + b^{[2]}. \end{aligned}$$

The first term is simply some new matrix, say W, times x. And the second term is a vector whose dimensions equal $(n^{[2]}, 1)$, the same as $b^{[2]}$. So even with two layers, we end up with the equivalent of just one layer.

The ReLU activation is not differentiable at 0, which is a problem since we need to compute its gradient. During implementation, we arbitrarily set it to either 0 or 1.

There are other activation functions as well:

– $g(x) = \tanh(x)$ is sometimes used instead of sigmoid.



– Another activation function sometimes used is the Exponential Linear Unit (ELU). It is defined as
$$g(x;\alpha) = \begin{cases} x; & x \geq 0 \\ \alpha(e^x - 1); & x < 0. \end{cases}$$

B.3.1 Vanishing gradient problem

Consider the sigmoid activation function. Specifically, we look at its derivative.

$$g'(x) = g(x)(1 - g(x)).$$

When x gets large, $g(x) \to 1$, and therefore $1 - g(x) \to 0$, and we have $g'(x) \to 0$. Similarly, when x is very small (that is, high in magnitude, but negative), then $g(x) \to 0 \Rightarrow g'(x) \to 0$. This is a problem. But there is another problem. The maximum value of the gradient can be found as follows:

$$\begin{aligned} f(x) &= g(x)(1 - g(x)) \\ &= g(x) - g^2(x) \\ f'(x) &= g'(x) - 2g(x)g'(x) = 0 \\ \Rightarrow g'(x) &= 2g(x)g'(x) \\ \Rightarrow g(x) &= \frac{1}{2}. \end{aligned}$$

Therefore, the maximum value of the gradient is $\frac{1}{4}$. As we perform backpropagation, layer by layer, these gradients will get multiplied and thus get smaller and smaller. The deeper the network, the smaller the gradients, and the bigger this problem becomes. This is called the *vanishing gradients problem.* This is precisely why networks using only the sigmoid activation have trouble learning well.

ReLU does not suffer from this problem technically, since its gradients are either 0 or 1. But it has its own curious problem, called the *dying ReLU problem.* In this situation, all the inputs to a particular neuron are such that the output is 0, and so the gradient is also 0 and this neuron never really learns (since during weight update, the gradient is 0). This is rare, but happens rarely and can render a network less effective.

To make sure this never happens, an activation function called the *leaky ReLU* was proposed. Rather than force the output to 0 on the negative side, we allow a small amount to "leak". So when $x < 0$, $g(x) = 0.01x$, and we could use any constant instead of 0.01.

B.3.2 Replicated weight neural network (RWNN)

Artificial Neural networks (ANN) consist of several layers – input, hidden and output, as shown in Figure 2. These layers are designed to forward and back propagate error such that the difference between the target and output get minimized after certain number of epochs. An epoch uses same training samples and update weights. On the contrary, an iteration runs over different set of training samples every time. Thus, a batch of epochs is called an iteration. Usually, the network trains itself by updating an initial random choice of weights through back propagation. We propose a novel scheme, RWNN to train the network. It is built with randomly initialized weights where learning rate and number of neurons across layers are kept same as in ANN.

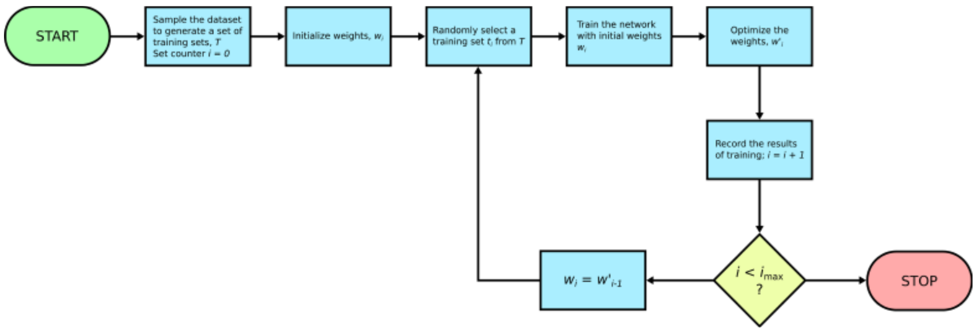

START
Sample the dataset to generate a set of training sets, T Set counter $i = 0$
Initialize weights, w_i
Randomly select a training set t_i from T
Train the network with initial weights w_i
Optimize the weights, w'_i
Record the results of training; $i = i + 1$
$i < i_{max}$?
STOP
$w_i = w'_{i-1}$

Fig. B.1. Architecture pipeline for replicated weight neural network. (For a better readability please refer to the online version).

The first iteration generates a balanced training set by random-oversampling and trains the network. This is identical to the original vanilla feed forward neural net. However, the optimized weights obtained from first iteration are utilized in training network during the second iteration. Precisely, this is how RWNN is different from a standard ANN architecture. ANN updates weights in every iteration (which any other iterative method should do anyway). The essence is that already optimized network (not completely optimized though, since it was trained using few samples from majority class) is passed down to next level of training with the new set of data samples. The cascading effect is observed when the same network is tuned in successive iterations with entirely new data set generated on-the-fly while weights get tuned in each iteration. Accuracy is recorded at every iteration and the whole process is carried out till a fixed number of iterations are completed.

Most neural network architectures use random initialization of edge weights and the iterative change of the weights to improve the overall performance of a neural network. Instead of random re-initialization of weights in every iteration, RWNN uses an optimized set of weights and update the set of weights accordingly. In other words, the method rushes convergence by avoiding random re-initialization of weights in every iteration and setting a threshold on iterations rather than on error (MSE) (Fig. B.1).

B.3.3 Synthetic oversampling using Parzen Window estimation

Artificial oversampling [50] is an approach to handle the effects of bias due to a class with a large sample size [51] in data. Typically, people try to artificially add samples to the classes with lesser number of samples, namely the mesoplanet and the psychroplanet classes. Essentially, the paradigm of choice is to analyze the class-wise distribution of the data and to generate reasonable samples which reflect the general properties of their respective classes.

In this approach, we estimate the density of the data by approximating a distribution function empirically – we do not assume that the numeric values in the data samples are drawn from a standard probability distribution. The method we use for this is known as *window estimation*. This approach was developed by Rosenblatt and Parzen [58,88]. In a broader sense, window estimation is a method of kernel density estimation (KDE). In our work, we use this to augment the samples of the different classes in the data and we choose the features that we require in different experimental scenarios.

It is also possible to augment variables from the top 85% of the features based on their importance as determined by random forest classifiers [89]. The experiments using KDE are done in the same manner as the experiments using undersampling.

B.3.4 Generative adversarial networks

Generative adversarial networks (GANs) are a powerful framework of a pair of competing neural networks where the generative network imitates the input and discriminating network differentiates between authentic and synthetic samples. They are popularly used for unsupervised learning for tasks as image generation from descriptions or rough sketches, getting high resolution images from low resolution ones etc. The two networks contest with each other, where the generator creates realistic images and the discriminator forces the generator to perform better.

It can also be used a semi-supervised learning technique where the discriminator learns the target function based on both labeled and unlabeled generated data. A semi-supervised classifier takes a small portion of labeled data and a larger amount of unlabeled data and uses both to learn an inferred function, which can further be used to classify new points. This can enhance learning of cluttered data, which have a large diversity. GANs are not frequently used for classification of numeric/text data; however, it is a method worth exploring in binary classification of astronomical objects.

Based on the conservative set of features in each experimental setup, described in Sections 4.1 and 10, objects from the three classes are cluttered. By not considering surface temperature and associated attributes, we lose the clear demarcation between classes and that makes the classification task we aim to accomplish quite challenging.

B.3.5 A novel neural network pipeline

Using the RWNN architecture and GANs, [53] a "fusion-neural net approach" may be proposed where RWNN is applied in conjunction with GANs to achieve desirable performance metrics. The pipeline is as follows:

1. *Classification between non-habitable, mesoplanet, and psychroplanet classes*: First, we employ RWNN to separate the non-habitable class, which is the majority class. At this stage, although a high accuracy is achieved in discriminating between the non-habitable class on one side, and the mesoplanet and psychroplanet classes on the other side, the discrimination between the classes of mesoplanets and psychroplanets is not perfect.
2. *Classification between mesoplanet, and psychroplanet classes*: Once the non-habitable class has been separated out, we use a GAN to distinguish between the mesoplanet and psychroplanet samples. For this, the generator function *generates* samples within the network with some amount of induced noise, and the discriminator tries to best separate the classes. The effects of the two components of the GAN put together amount to a robust discrimination between the mesoplanet and the psychroplanet classes.

The entire pipeline of the fusion network is shown in Figure B.2 and the configuration of the GAN is shown in Figure B.3.

B.3.6 Training the GAN for categorical semi-supervised learning

Standard GANs turn the discriminator into a classifier that is only useful for determining whether a given data point x belongs to the input data distribution or not. For classification, we aim to train a discriminator that classifies the data into k categories by assigning a label y to each input sample x. In a standard GAN, the discriminator is used to determine whether a given data point belongs to the distribution of the

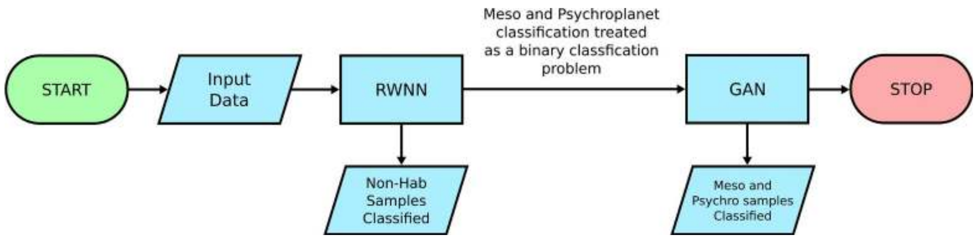

START
Input Data
RWNN
Meso and Psychroplanet classification treated as a binary classfication problem
GAN
STOP
Non-Hab Samples Classified
Meso and Psychro samples Classified

Fig. B.2. Fusion neural-net approach for classification: First tier of the network classifies between habitable and non-habitable planets; second tier architecture uses GAN to discriminate among mesoplanets and psychroplanets (classes within the habitable class): the fused network produces perfect classification when Min Mass, P. distance and star properties are used as features (see Tab. 2).

input sample. In order to use GANs for classification, the discriminator should not only model data distribution but also learn the feature representation for separating the data into classes. Using it as a categorical classifier requires the objective to accomplish the following:

1. Instead of the discriminator returning a probability of a sample being real or synthetic, it should provide class assignments to all the encountered samples, while remaining uncertain of class assignments made to synthetic generated data.
2. Instead of generating samples replicating the input dataset, each synthetic sample should belong to precisely one class with a high certainty; the number of synthetic samples generated in the neural network should be balanced across all the classes [52].

The probability of an example x belonging to one of the k mutually exclusive classes can be modelled using a soft-max assignment based on the discriminator output. The generator produces synthetic samples with random noise, which are used along with the authentic samples to train the discriminator in batches. This forces the generator to replicate the original data, which contributes to training samples for improving classification. By providing feedback to the generator to improve the replication of input, the discriminator is transformed into a classifier which has been trained both by labeled input data and unlabeled generated data in mini-batches. The discriminator still identifies individual examples as real data or generated data, but it is now able to use the other examples in the mini-batch to enhance the coordination between gradients [53]. We compute statistics when discriminator trains on the real batch, followed by the ones obtained by training on a synthetic batch. Training batch-wise mitigates problem of mode collapse by looking at multiple examples in a combination rather than in isolation. The Figure B.3 presents an overview of the architecture of the networks.

In addition to trying the fusion-net architecture on the data, GANs as classifiers can be tested in a stand-alone experiment where two classes are *habitable*, which comprise of the mesoplanet and psychroplanet samples together, and non-habitable samples. The fusion-net architecture and GAN would constitute Phase I of experiments. Phase II shall highlight limitations of standard activation functions in the backpropagation part of vanilla feed-forward neural network and suggest improvements. The outcome of this phase is particularly interesting in the context of the problem. We reiterate that, such architectural innovations are usually not backed by hard Mathematical proofs as human neural networks are too complex to mimic. To be precise, it is often, not possible to explain why a particular architecture such as the one described here, succeeds or fails!

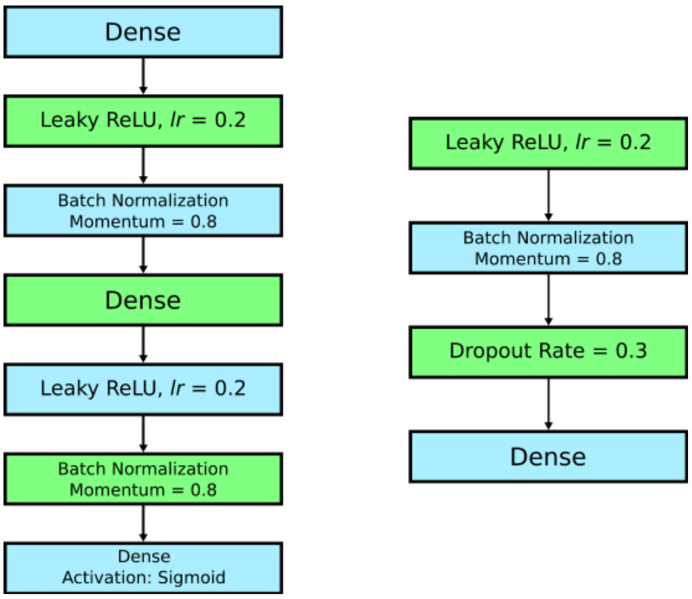

Dense
Leaky ReLU, lr = 0.2
Batch Normalization Momentum = 0.8
Dense
Leaky ReLU, lr = 0.2
Batch Normalization Momentum = 0.8
Dense Activation: Sigmoid
Leaky ReLU, lr = 0.2
Batch Normalization Momentum = 0.8
Dropout Rate = 0.3
Dense

Fig. B.3. Layers of the Generator and Discriminator in our GAN model with batch-size of 128.

B.4 Disadvantages of neural networks

B.4.1 Speed

Neural networks are slow to train. Larger neural networks can take days to train, even on the best hardware. Neural networks are almost always trained on GPUs (Graphics Processing Units), rather than the CPU of the computer. While CPUs have about 4–8 cores, GPUs can have up to thousands of cores. Of course, the cores on a CPU are individually much more powerful than GPU cores.

The idea behind using GPUs for training is simple: to train a neural network, you need to do lots of matrix multiplications. If you give a large matrix multiplication to a CPU, it takes time, because even though its cores are much faster and more powerful, there still are only 4–8 cores. A GPU, on the other hand, has thousands of cores that can very quickly do matrix multiplications.

B.5 Interpretability

With more “classical” machine learning models like linear regression, we have coefficients that are relatively easy to interpret in terms of the original variables of your data. With neural networks, it is much harder to figure out why it works even when it does. There has been a lot of work on interpretability in neural networks. A lot of these work on visualizing the loss surface[8], or as in the seminal paper by Zeiler and Fergus[9], visualizing the activations at each layer of the network.

[8] H. Li et al., Visualizing the loss landscape of neural nets, in *Advances in Neural Information Processing Systems.* (2018)

[9] M.D. Zeiler, R. Fergus, *Visualizing and understanding convolutional networks. European conference on computer vision.* (Springer, Cham, 2014)

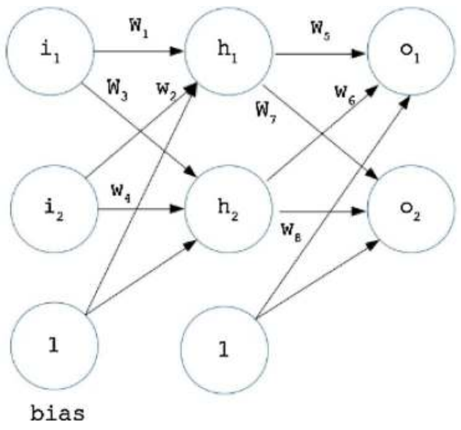


Fig. C.1. A simple neural network.

Appendix C: Neural networks with SBAF and A-ReLU

A neural network is an interconnection of neurons arranged in hierarchy and predominantly used to perform predictions and classification on a dataset. Commonly, the input is given to the network over and over again to tune the network a little each time, so that when the new inputs are given, the network can predict its outcome. To explain how neural network works, we will run through a simple example of training a small network shown in figure below. Keeping its configuration simple, we have kept 2 nodes in input, hidden and output layer and have chosen SBAF and A-ReLU as activation functions to demonstrate the working of back propagation [90].

C.1 Basic structure

Let us assume the nodes at input layer are i_1, i_2, at hidden layer h_1, h_2 and at output layer o_1, o_2. To start off, we assign some initial random numbers to weights and biases and move on with a forward pass demonstrated in next subsection (Fig. C.1).

C.2 The forward pass using SBAF and A-ReLU

This section computes the activation of neurons at hidden and output layers by using SBAF and A-ReLU. We start with the first entry in the data set of two inputs i_1 and i_2. The forward pass is a linear product of inputs and weights added with a bias. Calculating the total input at h_1.

$$h1_{\text{net}} = w_1 \cdot i_1 + w_2 \cdot i_2 + b_1$$

$$h2_{\text{net}} = w_3 \cdot i_1 + w_4 \cdot i_2 + b_1.$$

Use SBAF to calculate the activation's of hidden neuron h_1 by the formula, $y = \frac{1}{1+kx^{\alpha}(1-x)^{1-\alpha}}$.

$$h1_{\text{out}} = \frac{1}{1 + k(h1_{\text{net}})^{\alpha}(1 - h1_{\text{net}})^{1-\alpha}}$$

$$h2_{\text{out}} = \frac{1}{1 + k(h2_{\text{net}})^{\alpha}(1 - h2_{\text{net}})^{1-\alpha}}.$$

In parallel, if we use A-ReLU to compute the activation's of hidden neuron h_1, the formula is $y = kx^n$ if $x > 0$ else $y = 0$, therefore

$$h1_{\text{out}} = k(h1_{\text{net}})^n$$

$$h2_{\text{out}} = k(h2_{\text{net}})^n$$

assuming $h1_{\text{net}} > 0$ and $h2_{\text{net}} > 0$.
Repeat the process for neurons at output layer.

$$o1_{\text{net}} = w_5 \cdot h1_{\text{out}} + w_6 \cdot h2_{\text{out}} + b_2$$

$$o2_{\text{net}} = w_7 \cdot h1_{\text{out}} + w_8 \cdot h2_{\text{out}} + b_2.$$

While using SBAF, the activation of neurons are

$$o1_{\text{out}} = \frac{1}{1 + k(o1_{\text{net}})^{\alpha}(1 - o1_{\text{net}})^{1-\alpha}}$$

$$o2_{\text{out}} = \frac{1}{1 + k(o2_{\text{net}})^{\alpha}(1 - o2_{\text{net}})^{1-\alpha}}.$$

For A-ReLU, the activation are

$$o1_{\text{out}} = k(o1_{\text{net}})^n$$

$$o2_{\text{out}} = k(o2_{\text{net}})^n$$

assuming $o1_{\text{net}} > 0$ and $o2_{\text{net}} > 0$.

Since we initialized the weights and biases randomly, the outputs at the neurons are off-target. Conclusively, we need to compute the difference and propagate it back to the network. The next subsection computes the error gradient by using back propagation and adjust the weights to improve the network. Calculating the errors,

$$\text{Error} = \text{Error}_{o1} + \text{Error}_{o2}$$

$$\text{Error}_{o1} = \frac{1}{2}\left(o1_{\text{target}} - o1_{\text{out}}\right)^2$$

$$\text{Error}_{o2} = \frac{1}{2}\left(o2_{\text{target}} - o2_{\text{out}}\right)^2.$$

C.3 The backward pass for both SBAF and A-ReLU

This part deals with computing the error margins, the error resulted because the weights are randomly initialized. Therefore, the weights need adjustments so that the error can be decreased during predictions. Calculating the change of weights in done in two steps. The rate of change in error with respect to weights in computed in first step. In the second, the weights are updated by subtracting a portion of error gradient from weights. Similar to the forward pass, the backward pass is also computed layer-wise, but in the reverse mode.

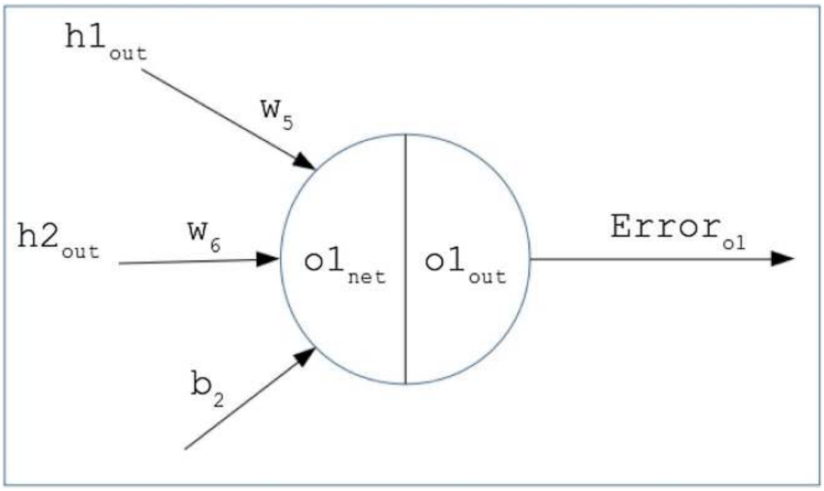


Fig. C.2. A simple neuron at the output layer.

C.3.1 At output layer

Moving backwards, lets consider weight w_5 that needs to be updated. To find the error gradient with respect to w_5, i.e. $\frac{\partial E_T}{\partial w_5}$ we use the chain rule shown in the equation below (here E_T is the total error at both neurons of output).

$$\boxed{\frac{\partial E_T}{\partial w_5} = \frac{\partial E_T}{\partial o1_{\text{out}}} \cdot \frac{\partial o1_{\text{out}}}{\partial o1_{\text{net}}} \cdot \frac{\partial o1_{\text{net}}}{\partial w_5}.}$$

Taking each component one at a time on the RHS of the equation, we first derive $\frac{\partial E_T}{\partial w_5}$ (Fig. C.2):

$$\boxed{\begin{aligned} E_T &= E_{o1} + E_{o2} \\ E_T &= \frac{1}{2}\left(o1_{\text{target}} - o1_{\text{out}}\right)^2 + \frac{1}{2}\left(o2_{\text{target}} - o2_{\text{out}}\right)^2 \\ \frac{\partial E_T}{\partial o1_{\text{out}}} &= 2 \cdot \frac{1}{2}\left(o1_{\text{target}} - o1_{\text{out}}\right) \cdot (-1) + 0 \\ \frac{\partial E_T}{\partial o1_{\text{out}}} &= -\left(o1_{\text{target}} - o1_{\text{out}}\right). \end{aligned}} \tag{C.1}$$

Next we compute the derivative of the activation functions in terms of $\frac{\partial o1_{\text{out}}}{\partial o1_{\text{net}}}$. We are using SBAF and A-ReLU, and dervatives of both the functions are available. First, while using SBAF:

$$\boxed{\begin{aligned} o1_{\text{out}} &= \frac{1}{1 + k(o1_{\text{net}})^{\alpha}(1 - o1_{\text{net}})^{1-\alpha}} \\ \frac{\partial o1_{\text{out}}}{\partial o1_{\text{net}}} &= \frac{o1_{\text{out}}\left(1 - o1_{\text{out}}\right)}{o1_{\text{net}}(1 - o1_{\text{net}})} \cdot (\alpha - o1_{\text{net}}). \end{aligned}} \tag{C.2}$$

While using A-ReLU,

$$\boxed{\begin{aligned} o1_{\text{out}} &= k.(o1_{\text{net}})^n \\ \frac{\partial o1_{\text{out}}}{\partial o1_{\text{net}}} &= n.k^{\frac{1}{n}}.(o1_{\text{out}})^{\frac{n-1}{n}}. \end{aligned}} \tag{C.3}$$

Finally the third component of the chain rule $\frac{\partial o1_{\text{net}}}{\partial w_5}$ (this is common for both SBAF and A-ReLU),

$$\boxed{\begin{aligned} o1_{\text{net}} &= w_5 \cdot h1_{\text{out}} + w_6 \cdot h2_{\text{out}} + b_2 \\ \frac{\partial o1_{\text{net}}}{\partial w_5} &= h1_{\text{out}}. \end{aligned}} \tag{C.4}$$

The error gradient with respect to w_5 for SBAF is derivable by putting (21) and (22) and (24) together in $\frac{\partial E_T}{\partial w_5}$,

$$\frac{\partial E_T}{\partial w_5} = -\left(o1_{\text{target}} - o1_{\text{out}}\right) \cdot \frac{o1_{\text{out}}\left(1 - o1_{\text{out}}\right)}{o1_{\text{net}}(1 - o1_{\text{net}})} \cdot \left(o1_{\text{net}} - \alpha\right) \cdot h1_{\text{out}}.$$

Likewise the other gradients are also computed as,

$$\frac{\partial E_T}{\partial w_6} = -\left(o1_{\text{target}} - o1_{\text{out}}\right) \cdot \frac{o1_{\text{out}}\left(1 - o1_{\text{out}}\right)}{o1_{\text{net}}(1 - o1_{\text{net}})} \cdot \left(o1_{\text{net}} - \alpha\right) \cdot h2_{\text{out}}$$

$$\frac{\partial E_T}{\partial w_7} = -\left(o2_{\text{target}} - o2_{\text{out}}\right) \cdot \frac{o2_{\text{out}}\left(1 - o2_{\text{out}}\right)}{o2_{\text{net}}(1 - o2_{\text{net}})} \cdot \left(o2_{\text{net}} - \alpha\right) \cdot h1_{\text{out}}$$

$$\frac{\partial E_T}{\partial w_8} = -\left(o2_{\text{target}} - o2_{\text{out}}\right) \cdot \frac{o2_{\text{out}}\left(1 - o2_{\text{out}}\right)}{o2_{\text{net}}(1 - o2_{\text{net}})} \cdot \left(o2_{\text{net}} - \alpha\right) \cdot h2_{\text{out}}.$$

Correspondingly, the error gradient with respect to w_5 for A-ReLU is derived by keeping (21), (23) and (24) together,

$$\frac{\partial E_T}{\partial w_5} = -n.k^{\frac{1}{n}}\left(o1_{\text{target}} - o1_{\text{out}}\right) \cdot \left(o1_{\text{out}}\right)^{\frac{n-1}{n}} \cdot h1_{\text{out}}.$$

Likewise the other gradients are also computed as,

$$\frac{\partial E_T}{\partial w_6} = -n.k^{\frac{1}{n}}\left(o1_{\text{target}} - o1_{\text{out}}\right) \cdot \left(o1_{\text{out}}\right)^{\frac{n-1}{n}} \cdot h2_{\text{out}}$$

$$\frac{\partial E_T}{\partial w_7} = -n.k^{\frac{1}{n}}\left(o2_{\text{target}} - o2_{\text{out}}\right) \cdot \left(o2_{\text{out}}\right)^{\frac{n-1}{n}} \cdot h1_{\text{out}}$$

$$\frac{\partial E_T}{\partial w_8} = -n.k^{\frac{1}{n}}\left(o2_{\text{target}} - o2_{\text{out}}\right) \cdot \left(o2_{\text{out}}\right)^{\frac{n-1}{n}} \cdot h2_{\text{out}}.$$

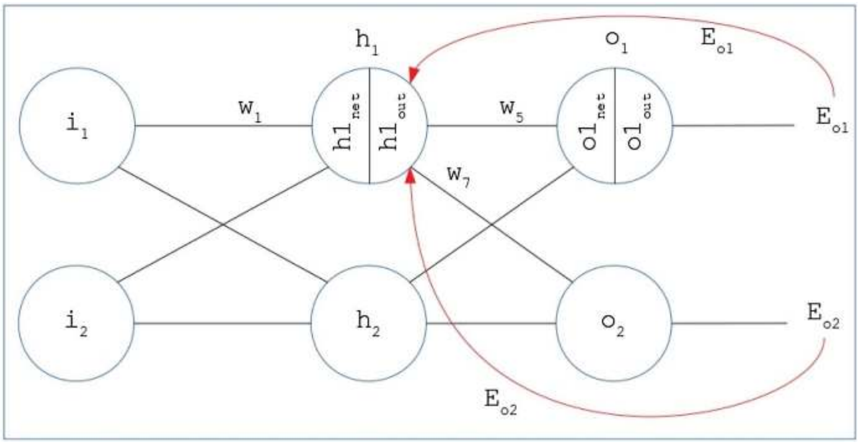
Both activation functions, the weights are adjusted as:

$$w_5^{new} = w_5 - \eta \cdot \frac{\partial E_T}{\partial w_5} \tag{C.5}$$

where η is the learning rate.

C.3.2 Hidden layer

Continuing the backward pass, this part demonstrate the computation of error gradients with respect to weights that are connecting input and hidden layer. Once the gradients are found, the weights can be updated by the formula used previously. Thus, we need to derive $\frac{\partial E_T}{\partial w_1}$ and correspondingly the other gradients can be obtained.



We need to find $\frac{\partial E_T}{\partial w_1}$.
Apparently, if E_T is the total error at output layer, then

$$\frac{\partial E_T}{\partial w_1} = \frac{\partial E_{o1}}{\partial w_1} + \frac{\partial E_{o2}}{\partial w_1}.$$

Computing both the additive terms by using chain rule,

$$\frac{\partial E_{o1}}{\partial w_1} = \frac{\partial E_{o1}}{\partial o1_{\text{out}}} \cdot \frac{\partial o1_{\text{out}}}{\partial o1_{\text{net}}} \cdot \frac{\partial o1_{\text{net}}}{\partial h1_{\text{out}}} \cdot \frac{\partial h1_{\text{out}}}{\partial h1_{\text{net}}} \cdot \frac{\partial h1_{\text{net}}}{\partial w_1} \tag{C.1}$$

$$\frac{\partial E_{o2}}{\partial w_1} = \frac{\partial E_{o2}}{\partial o2_{\text{out}}} \cdot \frac{\partial o2_{\text{out}}}{\partial o2_{\text{net}}} \cdot \frac{\partial o2_{\text{net}}}{\partial h1_{\text{out}}} \cdot \frac{\partial h1_{\text{out}}}{\partial h1_{\text{net}}} \cdot \frac{\partial h1_{\text{net}}}{\partial w_1}. \tag{C.2}$$

Finding the multiplicative terms of equation (1) (please note that this is with reference to SBAF. For A-ReLU, a similar procedure is followed),

$$\begin{aligned}
\frac{\partial E_{o1}}{\partial o1_{\text{out}}} &= -(o1_{\text{target}} - o1_{\text{out}})\\
\frac{\partial o1_{\text{out}}}{\partial o1_{\text{net}}} &= \frac{o1_{\text{out}}(1 - o1_{\text{out}})}{o1_{\text{net}}(1 - o1_{\text{net}})} \cdot (\alpha - o1_{\text{net}})\\
\frac{\partial o1_{\text{net}}}{\partial h1_{\text{out}}} &= w_5 \left(\because o1_{\text{net}} = w_5 \cdot h1_{\text{out}} + w_6 \cdot h2_{\text{out}} + b_2 \text{ and } \frac{\partial o1_{\text{net}}}{\partial h1_{\text{out}}} = w_5\right)\\
\frac{\partial h1_{\text{out}}}{\partial h1_{\text{net}}} &= \frac{h1_{\text{out}}(1 - h1_{\text{out}})}{h1_{\text{net}}(1 - h1_{\text{net}})} \cdot (\alpha - h1_{\text{net}})\\
\frac{\partial h1_{\text{net}}}{\partial w_1} &= i_1 \left(\because h1_{\text{net}} = w_1 i_1 + w_2 i_2 + b_1 \text{ and } \frac{\partial h1_{\text{net}}}{\partial w_1} = i_1 + 0\right).
\end{aligned}$$

Similarly, computing all the components of (2),

$$\begin{aligned}
\frac{\partial E_{o2}}{\partial o2_{\text{out}}} &= -(o2_{\text{target}} - o2_{\text{out}})\\
\frac{\partial o2_{\text{out}}}{\partial o2_{\text{net}}} &= \frac{o2_{\text{out}}(1 - o2_{\text{out}})}{o2_{\text{net}}(1 - o2_{\text{net}})} \cdot (o2_{\text{net}} - \alpha)\\
\frac{\partial o2_{\text{net}}}{\partial h1_{\text{out}}} &= w_7 \left(\because o2_{\text{net}} = w_7 h1_{\text{out}} + w_8 h2_{\text{out}} + b2\right).
\end{aligned}$$

We already know the values of $\frac{\partial h1_{\text{out}}}{\partial h1_{\text{net}}}$ and $\frac{\partial h1_{\text{net}}}{\partial w_1}$. Similar calculations hold valid for computing gradient of A-ReLU.

Adding up everything,

$$\frac{\partial E_T}{\partial w_1} = \frac{\partial E_{o1}}{\partial w_1} + \frac{\partial E_{o2}}{\partial w_2}.$$

Adjusting the weight

$$\boxed{w_1^{new} = w_1 - \eta \cdot \frac{\partial E_T}{\partial w_1}.}$$

Likewise, the remaining error derivatives, $\frac{\partial E_T}{\partial w_2}$, $\frac{\partial E_T}{\partial w_2}$, $\frac{\partial E_T}{\partial w_3}$, and $\frac{\partial E_T}{\partial w_4}$ can be computed in the similar manner for SBAF as well as for A-ReLU. Their corresponding weights are adjusted by using the same weight-update formula.

Appendix D: Details of data augmentation methods

In Section 4.2.2, a technique to estimate the probability density function of a sequence of random variables, called KDE (using Parzen-window estimation), was briefly described. We elaborate on the method and validate the method by performing a density estimation on random numbers generated from several known analytic continuous and discrete distributions. As a part of this exercise, we have presented goodness-of-fit scores for the estimated distributions. In addition to that, we have also plotted the graphs of the data points for a visualization of the density (generated using the standard analytic distributions as well as a Parzen-window estimate of those distributions).

We provide evidence of the working of Parzen-window estimation for different standard continuous and discrete probability distributions.

D.1 An analytical exposition of density exploration

Let $X = x_1, x_2, \ldots, x_n$ be a sequence of independent and identically distributed multivariate random variables having d dimensions. The window function used is a variation of the uniform kernel defined on the set R^d as follows:

$$\phi(u) = \begin{cases} 1 & u_j \leq \frac{1}{2} \quad \forall j \in \{1, 2, \ldots, d\} \\ 0 & \text{otherwise.} \end{cases} \tag{D.1}$$

Additionally, another parameter, the edge length vector $h = \{h_1, h_2, \ldots h_d\}$, is defined, where each component of h is set on a heuristic that considers the values of the corresponding feature in the original data. If f_j is the column vector representing some feature $j \in X$ and

$$\begin{aligned} l_j &= \min\{(a-b)^2 \quad \forall a, b \in f_j\} \\ u_j &= \max\{(a-b)^2 \quad \forall a, b \in f_j\}, \end{aligned} \tag{D.2}$$

the edge length h_j is given by,

$$h_j = c\left(\frac{u_j + 2l_j}{3}\right) \tag{D.3}$$

where c is a scale factor.

Let $x' \in R^d$ be a random variable at which the density needs to be estimated. For the estimate, another vector u is generated whose elements are given by:

$$u_j = \frac{x_j{}' - x_{ij}}{h_j} \quad \forall j \in \{1, 2, \ldots, d\}. \tag{D.4}$$

The density estimate is then given by the following relationship:

$$p(x') = \frac{1}{n \prod_{i=1}^{d} h_i} \sum_{i=1}^{n} \phi(u). \tag{D.5}$$

D.2 Generating synthetic samples from the estimated empirical distribution

Traditionally, random numbers are generated from an analytic density function by inversion sampling. However, this would not work on a numeric density function unless the quantile function is numerically approximated by the density function. In order to avoid this, a form of rejection sampling has been used.

Let r be a d-dimensional random vector with each component drawn from a uniform distribution between the minimum and maximum value of that component in the original data. Once the density, $p(r)$ is estimated by equation (D.5), the probability is approximated to:

$$Pr(r) = p(r) \prod_{j=1}^{d} h_j. \tag{D.6}$$

To either accept or reject the sample r, another random number is generated from a uniform distribution within the range $[0, 1)$. If this number is greater than the probability estimated by equation (D.6), then the sample is accepted. Otherwise, it is rejected.

For the PHL-EC dataset, synthetic data were generated for the mesoplanet and psychroplanet classes using this method by taking $c = 4$ for mesoplanets and $c = 3$ for psychroplanets (in Eq. (D.5)). 1000 samples were then generated for each class using rejection sampling on the density estimate. In this method, the bounding mechanism was not used and the samples were drawn out of the estimated density. Here, the top 85% of the features by importance were considered to estimate the probability density; the values of the remaining features were copied from the naturally occurring data points and shuffled between the artificially augmented data points. The advantage of using this method is that it may be used to estimate a distribution which resembles more closely the actual distribution of the data. However, this process is a little more complex than assuming a standard probability distribution in the data. Nonetheless, this is an inherently unassuming method and can accommodate distributions in data which are otherwise difficult to describe using the commonly used methods for describing the density of data.

D.3 Parzen-Window estimation of continuous random variables

For the tests involving univariate continuous random variables, standard distributions were used with the location parameter set to 0 and the scale set to 1. The distributions used in their univariate forms were:

(1) Normal
(2) Cauchy
(3) Laplace
(4) Rayleigh

Table D.1. Mean squared error for continuous standard distributions.

Distribution	c value	MSE ($\times 10^{-5}$)
Normal (univariate)	0.2	0.8164
Normal (multivariate)	0.23	3.3946
Cauchy	0.0002	1.7011
Laplace	0.07	3.3099
Rayleigh	0.15	4.3095

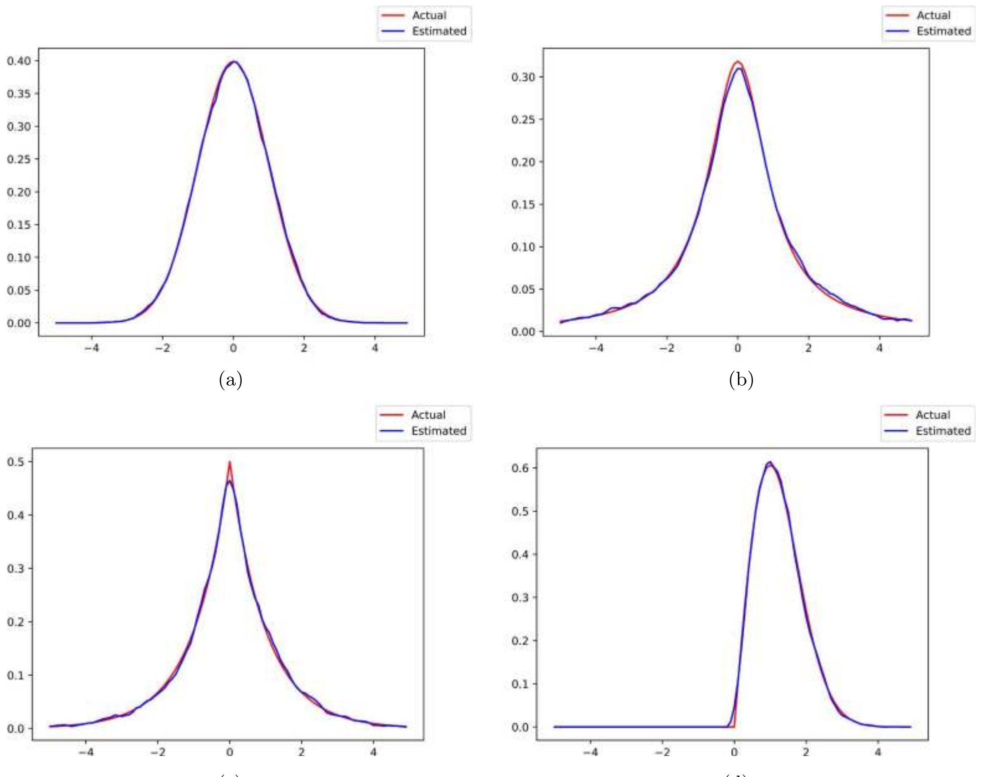


Fig. D.1. Estimates for univariate continuous distributions. (a) Normal. (b) Cauchy. (c) Laplace. (d) Rayleigh.

For multivariate continuous random variables, however, the test was performed on the normal density for which the mean and covariance were explicitly set to the following values:

$$\mu = \{0.5, -0.2\}$$
$$Z = \begin{bmatrix} 2.0 & 0.3 \\ 0.3 & 0.5 \end{bmatrix}. \tag{D.7}$$

For each distribution, 10 000 samples were generated and then used to estimate the original analytic distribution function. For the univariate case, the actual and estimated probability density were calculated over the range -5 to 5 by stepping at 0.1 and for multivariate, the same was done over the range $(-1, -1)$ to $(1, 1)$ by stepping at 0.1. The mean squared error was calculated and has been presented

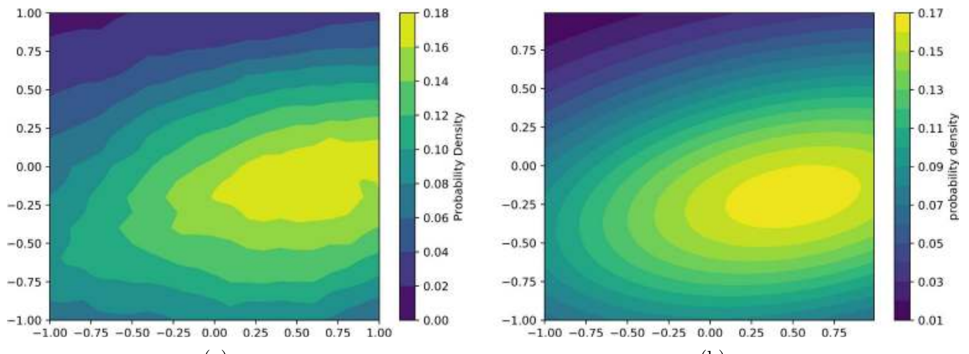


Fig. D.2. Multivariate normal density estimation in two dimensions. (a) Estimated. (b) Actual.

in Table D.1. The graphs of these calculated values are presented in Figure D.1 for univariate data and Figure D.2 for multivariate data.

In the cases of both discrete and continuous distributions, it is evident that KDE's performance is reasonable for generating data points. A small MSE or Chi-squared score is desirable as this represents a small deviation of artificially generated points from the actual distribution of the data. Thus, in experiments which require the estimation of the distribution of data, KDE is a candidate whose efficacy may be compared with the common methods of assuming a probability density using well established probability densities.

Appendix E: Introduction to chaos theory: Relevance to activation function and neural networks

Nonlinear systems abound in nature. Deterministic Chaos is the apparent random-like complex behaviour arising out of "simple" (typically low dimensional) non-linear deterministic bounded systems. The flapping of the wings of a butterfly in Kanyakumari can cause a snowstorm in Kashmir – popularly known as the butterfly effect, is the hallmark of chaos. This metaphor captures the *sensitive dependence on initial conditions* exhibited by chaotic complex systems which makes such deterministic systems unpredictable. Edward Lorenz, in his seminal 1963 paper [91] that introduced the bizarre yet fascinating world of nonlinear systems, summarized Chaos as – *When the present determines the future, but the approximate present does not approximately determine the future.* Thus, round-off errors in numerical computation and modeling of natural systems suffer from the butterfly effect making long term prediction practically impossible.

Chaos can generically be found in bounded deterministic non-linear systems, especially found in mathematical models of natural, biological, chemical, social and in even artificial human-made systems. Fluid flow, heart rate variability, firing of neurons in the brain, weather and climate, stock market, road traffic, social network dynamics and even decimal expansions of number systems (just a small list, but no means exhaustive) – all exhibit the tell tale signs of chaos. Chaos theory – the systematic scientific study of such nonlinear systems is a highly interdisciplinary theory which incorporates a host of rigorous mathematical/analytical as well as computational techniques such as recurrence plots, Poincaré maps, self-simiarlity, fractals, symbolic dynamics, non-linear differential equations (flows) and discrete time maps, bifurcation diagrams, lyapunov exponents computation, etc. [92–95].

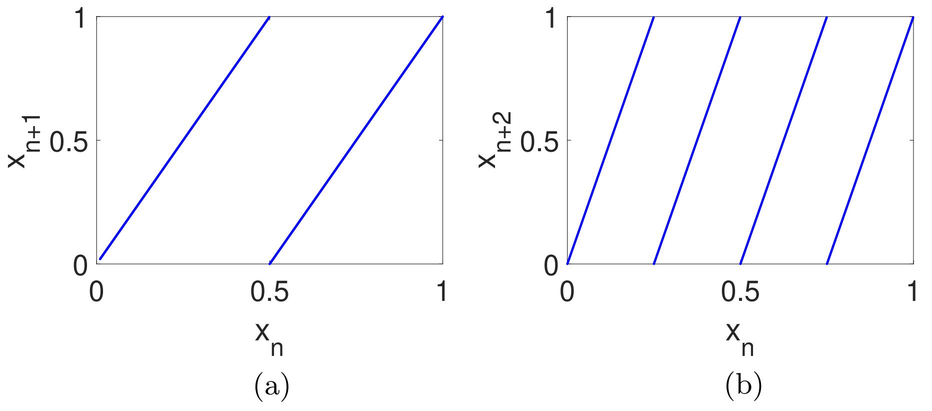


Fig. E.1. Chaotic Binary Map: (a) First-return map. (b) Second-return map.

The rich and complex behaviour of simple and innocent looking nonlinear dynamical systems includes dense and infinite periodic orbits, quasi-periodic and even an uncountably infinite of non-periodic or wandering trajectories, strange chaotic and non-chaotic attractors (sometimes concomitantly), attracting and repelling periodic orbits, etc.

We first describe a prototypical chaotic system with that displays nearly the full repertoire of rich and exotic properties of chaos and then highlight the relevance of chaos in learning, activation functions and neural networks.

E.1 A gentle introduction to chaos via the Binary Map

Arguably, the simplest deterministic system that exhibits chaos is the 1D nonlinear map known as the Binary Map or $T_2(x) : [0, 1) \mapsto [0, 1)$ and defined as:

$$\begin{aligned} T_2(x) &= 2x, & 0 \leq x < \frac{1}{2}, \\ &= 2x - 1, & \frac{1}{2} \leq x < 1. \end{aligned}$$

Figure E.1 shows the first-return map (x_{n+1} vs. x_n) and second-return map (x_{n+2} vs. x_n) of the Binary map (also known as Bernoulli Shift Map [92,96]) $T_2(x)$. Starting from an initial value x_0 and applying the Binary map iteratively leads to a trajectory (infinitely long): $\{T_2^0(x_0) \rightarrow T_2^1(x_0) \rightarrow T_2^2(x_0) \rightarrow T_2^1(x_0) \rightarrow \ldots \rightarrow T_2^n(x_0) \ldots\}$ or $\{x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \ldots x_n \rightarrow \ldots\}$. If one were to just record whether at every iteration x_i belongs to the left half of the interval $[0, 0.5)$ (record a symbol "0") or the right half $[0.5, 1)$ (record a symbol "1"), then one would get an infinitely long binary sequence[10] corresponding to the initial value x_0. This sequence is nothing but the binary expansion of the real number x_0 (hence the name Binary map). The Binary map satisfies all the required properties of chaos (Devaney's list [93]):

- Bounded, nonlinear deterministic map.
- Countably infinite number of dense periodic orbits. These correspond to the set of all rational numbers since their binary expansions are either terminating (infinite terminating zeros) or periodic/eventually periodic (recurring binary expansions).

[10] This sequence is called the symbolic seuqence of the trajectory corresponding to x_0.

- Uncountably infinite number of nonperiodic or wandering trajectories. These correspond to the set of all irrational numbers since their binary expansions are nonrepeating.
- Sensitive dependence on initial value. This can be understood intuitively as follows. We can approximate any irrational number (a non-periodic trajectory) by a rational number (a periodic trajectory) within an arbitrarily chosen accuracy ($\epsilon > 0$). Thus, a slight change in the initial value leads to a very qualitative behaviour (as seen in the structure of the trajectory). Sensitive dependence on initial value can be quantified by lyapunov exponent [92] which is positive for chaos. For the binary map, the lyapunov exponent is $= \ln(2)$.
- Topological transitivity. Given any two non-zero lebesgue measure sets, there always exists an initial value which when iterated a finite number of times reaches the second set.

E.2 Fixed points and their stability

For any chaotic nonlinear 1D map, fixed points are defined as those initial values which satisfy the equation: $f(x^*) = x^*$. Consider another 1D chaotic map, the logistic map: $x_{n+1} = 4x_n(1 - x_n)$ where $0 \leq x_n \leq 1$, n stands for the iteration number (Fig. E.2 depicts the sensitive dependence on initial values). All trajectories lie on a thin manifold – a parabola (see Fig. E.2). Solving for the fixed points for this map, $x^* = x_{n+1} = x_n$, we get $x^* = 0$ and $x^* = \frac{3}{4}$.

We can ask the following question: is the fixed point *stable*? By stability, what we mean is the following notion – if we slightly perturb the fixed point, do the trajectories converge to the fixed point or diverge away. In other words, how does neighbouring points of the fixed point behave under the map? Do their trajectories get *attracted* to the fixed point or *repelled*. Consequently, stable fixed points are called *attracting points* and unstable ones are called *repellers*. The mathematical condition for stability [92] is given by: $|\frac{df}{dx}|_{x=x^*} < 1$. It can be verified that for the logistic map, both $x^* = 0$ and $x^* = \frac{3}{4}$ are *unstable* fixed points.

Periodic points – are those initial values which yield periodic trajectories when iterated on the given map. A point is said to have a period m if $f^m(x^*) = x^*$ and $f^i(x^*) \neq x^*, \forall i = 1, 2, \ldots, m-1$. For example, a period two point would satisfy $f^2(x^*) = x^*$ and $f(x^*) = x^*$. These would be fixed points on the second-return map. For the Binary map (Fig. E.1b), they lie at the intersection of the map and the line $x_{n+2} = x_n$. However, two of these fixed points would correspond to period-0 points which we have to ignore. The remaining two fixed points correspond to period-2 points. Thus, to find all period-m points, we can find the fixed points of the m-th return map (discounting all those fixed points which correspond to the divisors of m since they would be periodic with a lesser period length). Stability criteria are easily extended to these periodic orbits as well [92].

E.3 Chaos and learning

Artificial Neural Networks (ANN) are only loosely inspired by the brain. The presence of chaos at various spatiotemporal scales in the human brain is well established by several experimental studies. This "neuro-chaos" is exhibited by networks of neurons in the brain [97,98] as well as individual activity seen at the level of a single neuron. Contrast this with ANNs where individual neurons are intrinsically linear and the only nonlinearity is at the activation function. This is sufficient to induce chaos at the level of networks, for e.g. Recurrent Neural Networks are known to exhibit chaos

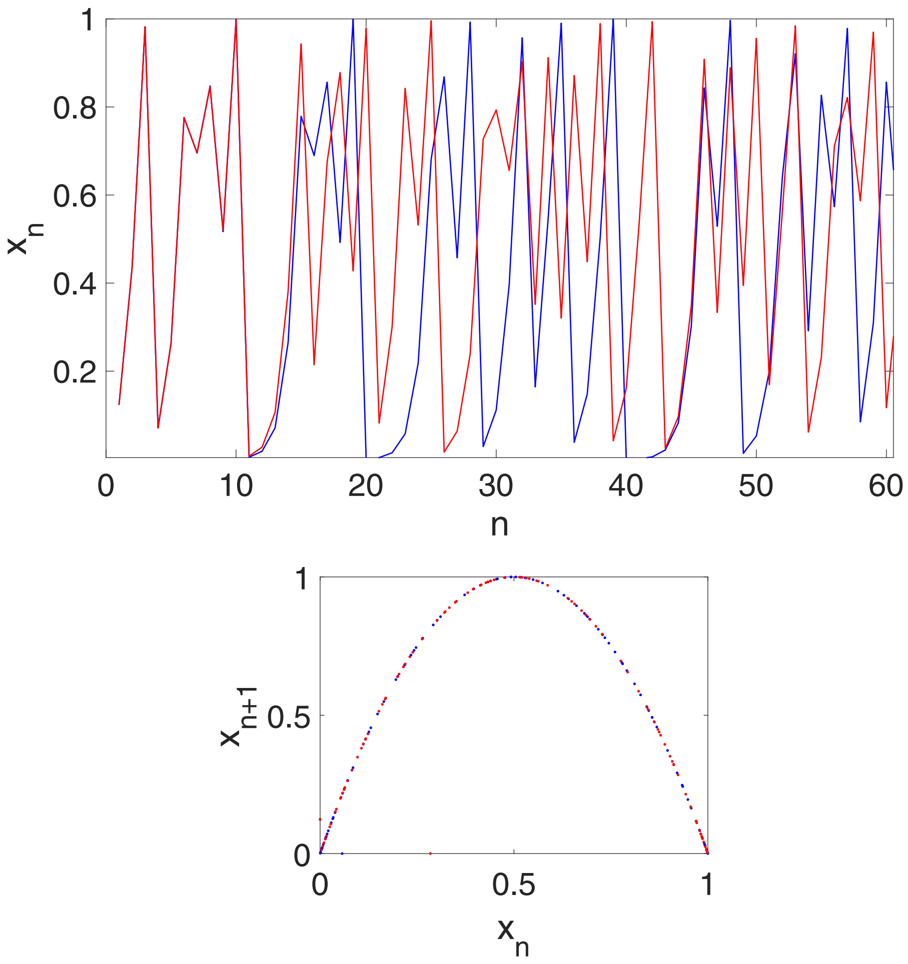


Fig. E.2. Logistic map: $x_n = 4x_{n-1}(1 - x_{n-1})$. Top: Sensitive dependence on initial values for two trajectories starting from $x_0 = 0.12345$ (Blue) and $x_0 = 0.12346$ (Red). As it can be seen, the two trajectories diverge after a few iterations. Bottom: The first return map for the two trajectories indicating that they lie on a thin manifold (a parabola).

[99]. There has been some early studies of the efficacy of chaos for learning. Sprott hypothesises that weak chaos allows for exploring a wide range of conditions while retaining some memory and predictability, which could be beneficial for learning [100]. Too much noise or chaos on the other hand is not good for learning. He specifically demonstrates that an ANN trained on logistic map time series at the onset of chaos is more effective when it exhibits weak chaos. He goes to further suggest that it is quite possible that human volunteer subjects might exhibit higher Lyapunov exponent in their EEG measurements during their performance of creative activities.

A novel neural network architecture for classification tasks employing 1D chaotic maps as individual neurons has been recently proposed [101]. Dubbed as `ChaosNet`, the authors of this work make use of the topological transitivity property of 1D chaotic maps known as Generalized Lurӧth Series (GLS) – the Binary map (described in E.1) is a specific example of GLS. These GLS neurons form the input layer of `ChaosNet` which when stimulated by the input data, produce a chaotic trajectory as its neural activity. The algorithm then proceeds to learn statistical features of this *neural code* to produce internal representation vectors. Their paper demonstrates the performance of `ChaosNet` on classification tasks on publicly available datasets. With as low as 0.05% of the total data used for training, their method reports accuracies ranging 73.89% to 98.33% while also claiming to be robust to external noise added to the weight vectors. The main reason for this success is the property of topological transitivity and the ability of chaos to abstract learning representations from limited samples while also being robust to noise. Probably, this is why chaos is found in abundance in the human brain – that it helps the brain to abstract learning representations with very few samples (unlike DNN) and is also very robust to neural noise and interference.

Activation function is the only nonlinear component of ANNs. They play a very important role in that they enable approximations to a large class of continuous functions. Cybenko [21] was one of the first to prove the Universal Approximation Theorem which states the conditions required for activation functions to achieve this feat. Linear functions simply do not have the requisite power to achieve this and hence the need for nonlinear activation function. Chaotic maps are necessarily nonlinear and ideally suited for approximations. In their `ChaosNet` paper, the authors demonstrate that the nonlinear chaotic neuron (1D GLS map) satisfies a version of the Universal Approximation Theorem. Unlike neurons in ANNs, the GLS neuron does not need a separate activation function and this is much more closer to how a neuron functions in the brain. We have explored the chaotic dynamics of SBAF. Additionally, future research could explore the rich properties of chaotic nonlinear functions for designing novel activation functions.

Appendix F: Interplay between activation and loss functions

This is a note on Sigmoid activation and cross-entropy loss. Suppose that $y*$ is a latent, continuous random variable. In the case of logistic regression, (leading to binary cross entropy), we assume that $y*\ \ f(x) + e$ where $e \sim \text{logit}(\text{mean} = 0)$ pdf. We can see that sigmoid is the CDF of logit distribution (or conversely, sigmoid is a valid CDF, and its derivate is the logit distribution), where $f(x)$ is the mean. Define $y = I(y* > 0)$. Then,

$$\begin{aligned} p(y = 1) &= p(y* > 0) \\ &= 1 - p(f(x) + e < 0) \\ &= 1 - \text{cdf}_{\text{logit}}(-f(x)) \\ &= 1 - \sigma(-f(x)). \end{aligned}$$

But $1 - \sigma(-x) = \sigma(x)$ due to symmetry. Therefore,

$$p(y = 1) = \sigma(f(x)).$$

The negative log-likelihood under the above generative model results in the binary cross entropy. We have followed the same constructive approach to come up with a loss function for binary quantile regression. There, the error distribution is asymmetric Laplace distribution.

Appendix G: A mind map of key ideas presented in the paper (Fig. G.1)

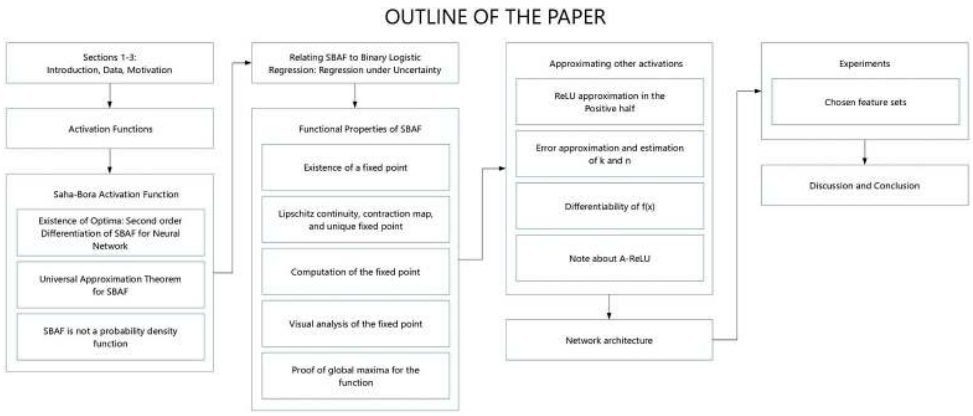

OUTLINE OF THE PAPER
Sections 1-3: Introduction, Data, Motivation
Activation Functions
Saha-Bora Activation Function
Existence of Optima: Second order Differentiation of SBAF for Neural Network
Universal Approximation Theorem for SBAF
SBAF is not a probability density function
Relating SBAF to Binary Logistic Regression: Regression under Uncertainty
Functional Properties of SBAF
Existence of a fixed point
Lipschitz continuity, contraction map, and unique fixed point
Computation of the fixed point
Visual analysis of the fixed point
Proof of global maxima for the function
Approximating other activations
ReLU approximation in the Positive half
Error approximation and estimation of k and n
Differentiability of f(x)
Note about A-ReLU
Network architecture
Experiments
Chosen feature sets
Discussion and Conclusion

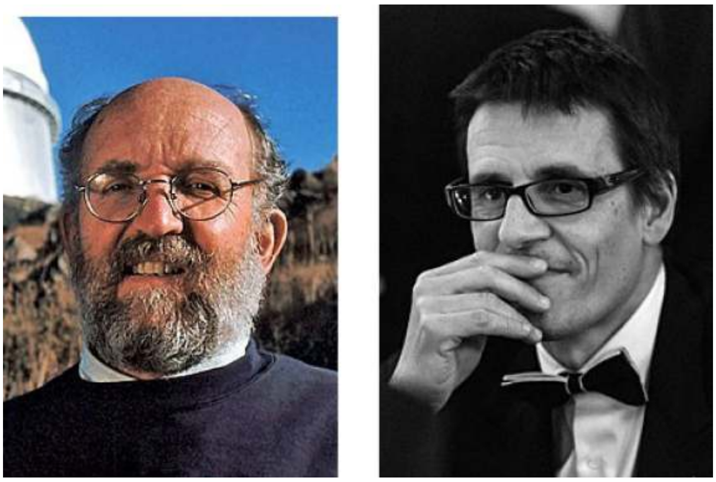
Fig. G.1. The flowchart is a visual description of integration of several ideas leading to novel activation functions and subsequent use in habitability classification of exoplanets. (For a better readability please refer to the online version).

Appendix H: Epilogue

H.1 Nobel prize

Nobel prize is a set of international awards presented by Swedish and Norwegian institutions to several laureates in different categories every year. The award is bestowed in various disciplines vis-a-vis Physics, Chemistry, Literature, Peace and Medicine in order to give recognition to scientific, academic and cultural advances in respective fields. Each year laureates receive a gold medal, a diploma and amount of money based on the donation received by the Nobel Foundation. By large, the prize is considered as the most reputed award in the field of Science, Literature and Physiology. Interestingly, Nobel prize is Physics for the year 2019 was awarded to two Swiss astronomers, Michel Mayor and Didier Queloz, for their outstanding discovery of an exoplanet (named 51-Pegasi-b) orbiting around Solar-like star. Nobel prize was awarded to them with prize-motivation cited as: "For the discovery of an exoplanet orbiting a solar-type star" The award was shared by Canadian-American Astrophysicist, James Peebles for laying theoretical foundations in Cosmology, and transforming the cosmic world into a precision science. The next section outlines the accomplishment of two Nobel Laureates that has granted benefits in the field of Astronomy and Astrophysics.

Michel Mayor Michel is a retired Professor from University of Geneva in the department of Astronomy. He received MS in Physics and PhD in Astronomy and had worked at European Southern Observatory and Observatory at Geneva to work closely on extra-solar planets and understand their statistical properties. He had spent a good amount of time in building photoelectric-based spectrometers for measuring radial velocity of stellar objects. Apart from the Nobel Prize award which he received for the discovery of exoplanet 52 Pegasi b, Michel Mayor is a recipient of the Swiss Marcel Benoist Prize, the Balzan Prize, the Albert Einstein Medal, BBVA Foundation Frontiers of Knowledge Award of Basic Sciences, the Gold Medal of the Royal Astronomical Society and the Wolf Prize in Physics (Fig. H.1).



(a) (b)

Fig. H.1. (a) Michel Mayor: Nobel Prize in 2019. (b) Didier Patrick Queloz: Nobel Prize in 2019.

Didier Patrick Queloz Professor Didier Queloz is a Professor at University at Cambridge, also a fellow of Trinity College, Cambridge and Professor at University of Geneva. He completed his M.Sc. degree in Physics from University of Geneva and obtained his PhD under the supervision of Dr Michel Mayor. Michel had been working on improving the measurements of radial velocity of stellar objects and had developed spectrometer, COREAL, to measure radial velocity with large accuracy. Michel and Didier later developed ELODIE, a refined spectrometer with a capacity to measure radial velocity of astronomical objects to a very large degree of accuracy.

51 Pegasi b is the first exoplanet discovered in the 1995 by Mayor and Didier using ELODIE spectrometer. Both the Swiss Astronomers confirmed the presence of extra-solar planet that orbits sun-like star at about 50 light years away from Earth. 51 Pegasi b has orbital period of 4 days with surface temperature of 1000 °C and is 50% larger in size than Jupiter.

H.2 Exoplanet detection methods and the most important exoplanet search missions

Time and again, there were missions launched across different parts of the world for discovering exoplanets. Space-based missions are large-scaled projects initiated by different agencies and universities led by a group of research scientists for placing large, complex, powerful telescope in space to know more about mysteries of stellar objects. These telescopes are launched with the help of a special launch vehicle or rocket. The space based missions include COROT, Kepler, K2 and TESS that have captured unobstructed image of the entire sky providing a large ground for studying stellar objects. Alternatively, there are many ground-based observatories equipped with telescope to make observations in different sections of sky at night. Few ground-based missions were MEarth Project, SuperWASP, KELT, and HATNet. Scientists keep a watchful eye on the objects and events of the universe with their telescope. Though there had been decent number of techniques available for detecting exoplanets, Pulsar timing, Transit method and Radial velocity method are predominantly used in these missions. Other detection methods contributing to other critical discoveries are Astrometry, Microlensing and Direct Imaging. Some of these methods are described below.

Pulsar Timing Wolszczan in 1994, devised a method of exoplanet detection that uses periodic variation in pulse signals received from a pulsar to detect planet orbiting around it. A pulsar is a "dead" star, that gets formed after its parent star explodes as supernova. Even the planets that are orbiting pulsars are formed due the same powerful stellar explosion indicative of the fact that formation of planet is not a rare phenomenon. The limitation here is, only those planets that orbit pulsars can be detected by this method; and pulsars are rarely seen as against the main sequence stars. Moreover, habitability of such planets is unlikely since pulsars emit high intensity radiations and chances of survival of life form on planets that orbits radiations-emitting stars are negligible. As a result the research in this direction becomes feeble.

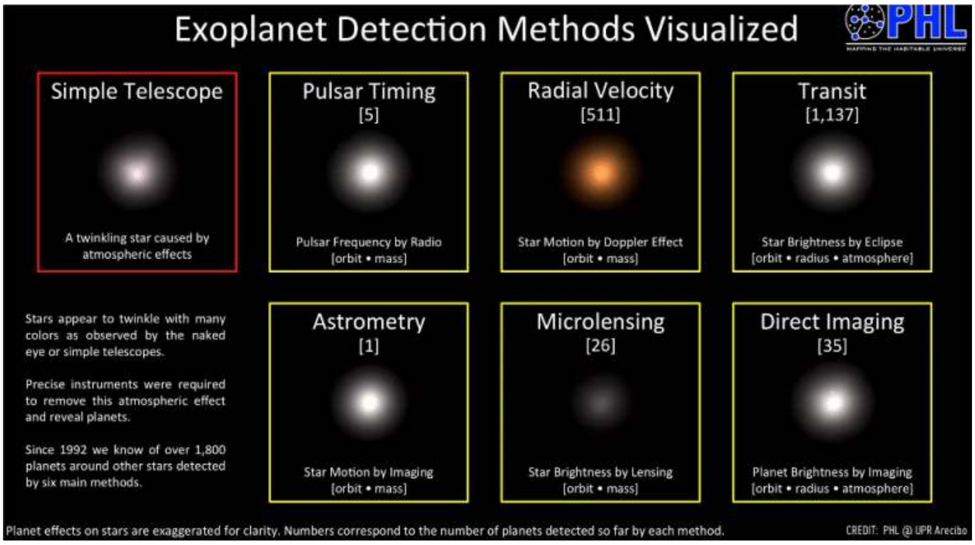
Transit Photometric Method Transit method utilizes a phenomenon when an exoplanet blocks or obstruct a part of the light being received from its host star. When an exoplanet is passing across its star, the event in called transit. The transiting object blocks some part of light reaching an observer on Earth. If dimming of light is observed at regular time period, then it is believed to be due to an exoplanet orbiting around its star. The transit photometric method is useful in finding planets that revolves in close orbit of its star. The method also enables the measurement of exoplanet diameter, though computing mass is not possible with this technique. Furthermore, exoplanet mass is computed using Radial Velocity method described next. Hence the two methods complement each other in characterizing exoplanets and enabling astronomers in deriving more features from already-computed ones.

Radial Velocity Method Radial velocity method (Doppler Spectroscopy) is a successfully utilized method for detecting exoplanet that relies on the fact that star is not completely stationary but moves in a small circle along with its planet when the planet is orbiting around it. The effect is known as Doppler effect. These movememts, even though small in degree, can be detected by highly powerful spectrographs used in telescopes for detecting exoplanets. Spectrographs monitors a star's spectrum and detects a spectral shift toward redder wavelength when star is moving away from the observer and toward bluer wavelength when it is moving in the direction of the observer. If these shifts appear at regular interval, one can certainly feel the presence of a planet tugging its star while orbiting around. This method of detecting planets became so revolutionary that scientists began developing highly powerful spectrographs which are capable of detecting light movements of star with higher accuracies (Fig. H.2).

H.2.1 Space based missions

The section brings-up few most popular and successful space missions conducted exclusively for exoplanet search and their characterization presented in reverse chronological order.

CHEOPS – Characterizing Exoplanets Satellite (December, 2019–ongoing) CHEOPS is a space-based mission that is first of its kind, launched with an intention to characterize and identify features of already discovered exoplanets. The project is an initiative of European Space Agency. The agenda is to obtain finer details of the exoplanets like diameter, mass and use these values to derive new parameter like density. Moreover, CHOEPS closely examine the surface of the planets to know whether they are rocky or in gaseous state and explore similar characteristics in order to facilitate astronomers in evaluating life-harbouring potency of planets. Though, images captured by the mission telescope are slightly blurred and


Exoplanet Detection Methods Visualized
PHL
Simple Telescope
A twinkling star caused by atmospheric effects
Pulsar Timing [5]
Pulsar Frequency by Radio [orbit • mass]
Radial Velocity [511]
Star Motion by Doppler Effect [orbit • mass]
Transit [1,137]
Star Brightness by Eclipse [orbit • radius • atmosphere]
Astrometry [1]
Star Motion by Imaging [orbit • mass]
Microlensing [26]
Star Brightness by Lensing [orbit • mass]
Direct Imaging [35]
Planet Brightness by Imaging [orbit • radius • atmosphere]
CREDIT: PHL @ UPR Arecibo

Stars appear to twinkle with many colors as observed by the naked eye or simple telescopes.

Precise instruments were required to remove this atmospheric effect and reveal planets.

Since 1992 we know of over 1,800 planets around other stars detected by six main methods.

Planet effects on stars are exaggerated for clarity. Numbers correspond to the number of planets detected so far by each method.

Fig. H.2. Exoplanet Detection Methods Visualized, Video is available at [102]. Credits: PHL@UPR Arecibo.

Fig. H.3. TESS Spacecraft: Loopable animation of TESS spacecraft. Credit: NASA's Goddard Space Flight Center/CI Lab – Animation video is available at TESS Spacecraft Animations, by Walt Feimer, Released on January 8, 2018, https://svs.gsfc.nasa.gov/20272 Credits: PHL@UPR Arecibo.

not very clear, scientists believe that this gives room to capture finer photometric precision in the images when telescope spots a transiting planet.

Transiting Exoplanet Survey Satellite (TESS) (April, 2018–ongoing) is launched by NASA in April 2018 to open possibilities for studying mass, size, radius and orbit of planets though it began its science operations from July 2018 in space. TESS is led by the Massachusetts Institute of Technology and was originally planned to complete its mission in 2 years. TESS is furnished with 4 wide-angle telescope with CCD cameras that uses transit photometry for detecting exoplanet. The cost of TESS project is US$200 million with an additional cost of US$87 million for launching the spacecraft. The image data collected by telescope are transmitted to Earth in every 2 weeks along with the transient images with exposure time of nearly 2 h for exploring unexpected phenomenon. HD 21749c is the first Earth-like exoplanet captured by TESS telescope which has 89% Earth diameter. Another tiny exoplanet, L 98-59b, orbiting a close star is found to have mass between Mars and Earth. TESS, being an all-sky survey, is believed to discover more than 20 000 exoplanets and few of them will be Earth-sized orbiting in the habitable zone. Till date, TESS has successfully discovered 1785 extra-solar planets, 45 are confirmed to be exoplanets.

Kepler (March, 2009–May, 2013) Nasa's Kepler mission was launched in March 2009 by sending Kepler spacecraft in search of Earth-like exoplanets orbiting around Sun-like star in the Universe. Kepler telescope has spent almost 9 years in space finding more and more exoplanets that are similar to Earth. The mission used transit photometry technique by searching for dips in brightness when the planet passes across its star. Kepler made its first 5 discoveries as Kepler-4b, Kepler-5b, Kepler-6b, Kepler-7b and Kepler-8b which were identified as hot Jupiters mostly in gaseous state. Soon after, Kepler-10b was discovered as first rocky planet, orbiting too close to its star and was named "Lava world" due to presence of molten rock on its surface. The first planet which was found in the habitable zone of its star was Kepler-22b. The planet is twice the size of Earth and orbits at perfect distance from star making its way to become suitable candidate for sustaining life on its surface. Subsequently, discovery of Kepler-62e and Kepler-62f brought new planets lying in the habitable zone and were recognized as "Super Earths" because their masses were higher than Earth's. Super-Earths are planets that are larger than Earth and smaller than Uranus and Neptune in size and mass. Another interesting discovery, Kepler-186f, brings to light the first planet which is similar in size with Earth, orbiting a red dwarf star and lies in its habitable zone. The star is 580 light years away from Earth.

K2 (May, 2016–November, 2018) It was NASA's subsequent mission that initiated after Kepler spacecraft faced technical malfunction in space in 2013. Kepler paused its operation in May 2013 when second of its 4 reaction wheels stopped working. A decision was made then to resume the mission after making a small fix in the functioning of Kepler's telescope. And within few months the mission began its exploration with a new name "K2". NASA's K2 mission used Kepler's telescope and discovered as many as 300 new exoplanets in its extended working. In December 2016, K2 discovered Trappist-1, a planetary system that has many Earth-sized planets. The most interesting outcome of the exploration revealed that the Universe has more number of Earth-like planets than Jupiter-like gas giants which gives out a ray of hope for finding life-form on planets outside solar system. The mission not just explored the presence of exoplanets but also brought into light many bright stars, galaxies, supernovae and asterseismology for researchers to conduct research in different dimensions. The mission got over in October 2018, after engineers found the spacecraft was running out of fuel, and later NASA officially announced the decommissioning of their space observatory from service.

CoRoT Convention, Rotation and Planetary Transit (CoRoT) December, 2006–June, 2014 was the first mission led by CNES in association with French laboratories and several International partners in 2006 for exploring stars and extra-solar planets. The internal structure of stars was also studied in the mission. This was made possible by closely examining the change in brightness of star, a technique named stellar seismology. Additionally, CoRoT used transit photometry for detecting exoplanets and 34 exoplanets were found in this mission, many of these were "Super-Earths". An interesting discovery of CoRoT-7b, a rocky planet which is 1.68 times Earth's size and 4.8 times Earth's mass orbiting its star at about 489 light years from Earth gained a lot of attention from scientist community

NN-EXPLORE, NASA-NSF Exoplanet Observational Research There had been plenty of space-based missions though, that have unveiled thousands of

Fig. H.4. Hubble finds more evidence of water vapor plumes on Europa. Credits: NASA's Goddard Space Flight Center, Animation video is available at Europa Water Vapor Plumes – More Hubble Evidence, Video Released on April 13, 2017, https://svs.gsfc.nasa.gov/12585.

exoplanet and stars but certainly, there would be many more undiscovered exoplanets in the universe to be captured in up-coming missions. Scientists and astronomers have realized a need to investigate the already available data on exoplanets. NN-EXPLORE has taken an initiative to affirm and assess the confirmed exoplanets and to perform high-level analysis of data on their scientific parameters. A partnership was established in 2017 between NASA and National Science Foundation for conducting ground-based radial velocity survey to pursue on observations made by Kepler, K2 and TESS missions. The work was conducted in two stages. In the initial stage, high precision Doppler Spectrometer was developed at the National Optical Astronomy Observatory (NOAO), and deployed at the 3.5-m Wisconsin, Indiana, Yale, and NOAO (WIYN) telescope. Alongside, a Guest Observer program, exclusively for exoplanet-targeted research scientists began to operate, in which the instrument was made available to Guest observers in time-sharing mode. In the second ongoing-stage, a database management system is build and the data is being stored for the scientists and research community for further explorations.

NASA's Hubble Telescope Hubble is a space telescope launched in the Earth's low orbit in 1990 that uncovered interesting aspects of plenty of exoplanets. During its mission, it explored characteristics of HD209458 b, and observed marks of sodium gas in its atmosphere. Likewise, Hubble also detected methane in the atmosphere of HD 189755 b, an exoplanet discovered sometime back. In a different perspective, Hubble created a history in detecting first organic molecule in the atmosphere of a planet that orbits outside our solar system. Discoveries led by Hubble indicated chemical impressions of life in its spectroscopic survey of exoplanet's atmosphere when it found the hydrogen-rich atmosphere, and the presence of other gases like CO_2, methane and oxygen (Fig. H.4).

H.2.2 Ground based observatories and their telescopes

Specific research teams have been investing time and money in constructing astronomical observatories for observing sky in the night. These observatories are build at very high altitudes. Here are details of location and functioning of a few popular

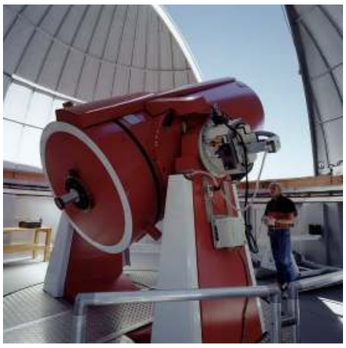


Fig. H.5. The Leonhard Euler Telescope in its dome at La Silla Observatory. Credit: ESO/H. Zodet.

observatories built by different research groups for exclusive study of exoplanets with regards to their potential to become habitable.

In recent past, countries like China, Japan and Korea took initiative to dive into the exploration of extra-solar planets and their characteristics. Scientists of these countries teamed-up and established *East-Asian Planet Search Network, (EAPSNET)*, to gather more accurate detection and measurements of exoplanets (Fig. H.3) using skilled instrument and technologies. They have been successful in detecting HD 119445 and scientists are optimistic about bringing more exoplanet characteristics into light. Similar ground-based missions at different Observatories in various parts of the world have gained momentum. Back in 1998, famous and reputed *Geneva Observatory* had built and deployed a Swiss EULER Telescope that internally use *CORALIE spectrograph* for detecting exoplanets. Currently, the Observatory hosted *Geneva Extrasolar Planet Search Program* in collaboration with many more organizations across the world to expedite exoplanet-science. The program was led by Michel Mayor and his team who utilized sensitive instruments like *CORALIE* and *ELODIE* for exoplanet detection. This resulted in discovery of plenty of exciting exoplanets including 51 Pegasi-b.

Similar initiatives at *La Silla Observatory in Chile* had discovered more than 130 exoplanet by using a powerful and sensitive planet-detecting spectrograph known as *High Accuracy Radial Velocity Planet Searcher (HARPS).* The spectrograph was established in 2004 and used radial velocity method to detect exoplanets. HARPS was used by Michel Mayor and Didier for characterisation of the Gliese 581 planetary system. *La Silla Observatory* also hosts a high-precision reflecting telescope, *Leonhard Euler Telescope*, which was built by Geneva Observatory in 1998. It uses CORALIE spectrograph to captures Dopplers effect of star for detecting planet. Euler telescope was used in Southern Sky extrasolar Planet search Programme for detecting multiple exoplanets (Fig. H.5).

Magellan Planet Search Program at *Las Campanas Observatory, Chile* is a ground-based mission started in 2002 for finding exoplanets that uses *Magellan II "Clay" telescope.* In a span of few years, the program had announced the discoveries of HD 48265 b, HD 86226 b, HD 129445 b, HD 143361 b and many more extrasolar planets that are orbiting their stars.

Wide Angle Search for Planets, WASP is an Ultra-wide angle search program for exoplanets started by international consortium of 6 universities across the globe. The search program uses two mighty robotic observatories that covers millions of stars and planets in northern and southern hemisphere. The mission named SuperWASP-North is at *Roque de los Muchachos Observatory* that covers northern hemisphere and SuperWASP-South covers southern hemisphere from the *South African Astronomical Observatory*, Sutherland. The Observatories have been using 8 wide-angle cameras to observe the entire sky. The researchers of SuperWASP have discovered one of the hottest Jupiter-sized planet, WASP-178b in August 2019.

H.2.3 Future missions and their special telescopes in development stage

The popular space and ground based missions like Kepler, K2, TESS and many more, have confirmed as many as 4152 exoplanets beyond our solar system. Many of these are orbiting in the habitable zone and could sustain liquid water on their surface hence, the challenge for astronomers swiftly shifts from finding exoplanets to seeking signs of life on planets.

James Webb Space Telescope (JWST or "Webb") European Space Agency and Canadian Space Agency supported and contributed in making of a special telescope, JWST that is planned to be launched in Earth's orbit as successor of the popular telescope, Hubble. As already mentioned, Hubble has been orbiting in space since 1990 and has contributed in various important exoplanets discoveries. The scheduled launch of JWST is March 2021 but there could be delay because of the impact of coronavirus pandemic. It is believed that JWST is highly complex, large and accurate telescope that will expand and explore discoveries made by Hubble with greater accuracy and improved sensitivity.

Planetary Transits and Oscillations of stars (PLATO) Another initiative of European Space Agency is the development of spacecraft with an aim to search for unexplored exoplanets that have been orbiting around millions of stars, many of those could be red dwarfs, yellow dwarfs and sub-giants stars. Not only detection, their bulk properties like mass, radius, architectures, their evolutionary process and atmospheric conditions will be examined as a part of mission so that scientists can get insights in determining their habitability potential. The telescope is equipped with 26 special cameras, out of those, 24 are "normal" cameras and 2 are "fast" cameras for observation of bright stars, arranged in a way to enable continuous observation of a region of sky. ESA is planning to launch the telescope in 2026.

Wide Field Infrared Survey Telescope, WFIRST an infrared telescope is in its development stage, proposed to be launched in 2025. Telescope is equipped with extra capabilities to capture sharp images in comparison to those obtained from Hubble. WFIRST instrument has two major components. The primary component is Wide Field instrument, to capture region of sky 100 times larger than Hubble infrared instrument. The method of Gravitational microlensing will be used to survey inner Milky Way for exoplanet search. The second component is Coronagraph instrument for inducing sharpness and contrast in images of exoplanets to be used for spectroscopy (Fig. H.6).

A great deal of reading material and books about exoplanets and mysteries of the universe are available like "Exoplanets" by Sara Seager, "The History of Astronomy: A Very Short Introduction" by Michael Hoskin, "Five Billion Years of Solitude: The Search for Life Among the Stars" by Lee Billings, "Mirror Earth: The Search for Our Planet's Twin" by Michael D. Lemonick, "Just Right: Searching for the Goldilocks Planet" by Curtis Manley, "The Edge of Physics: A Journey to Earth's Extremes to Unlock the Secrets of the Universe" by Anil Ananthaswamy are few popularly read books on exoplanets.

Fig. H.6. WFIRST Coronagraph Animation-Animation illustrating how a planet can disappear in a star's bright light, and how a coronagraph can reveal it. Animation video is available at WFIRST Coronagraph Animation, by Walt Feimer, Released on September 20, 2016, https://svs.gsfc.nasa.gov/20243. Animation Credits – Walt Feimer (HTSI): Lead Animator, Scott Wiessinger (USRA): Producer, Neil Gehrels (NASA/GSFC): Scientist.

References

1. https://fi.pinterest.com/amp/pin/142004194471914978/
2. A.M. Mendez, E.G. Rivera-Valent'in, D. Schulze-Makuch, J. Filiberto, R.M. Ramirez, T.E. Wood, A.F. Davila, C. McKay, K.O. Ceballos, M. Jusino-Maldonado, G. Nery, R. Heller, P. Byrne, M.J. Malaska, E. Nathan, M.F. Simoes, A. Antunes, J. Martinez-Frias, L. Carone, N.R. Izenberg, D. Atri, H.I. Chitty, P.V. Nowajewski-Barra, F. Rivera-Hernandez, C.M. Brown, K. Lynch, D.C. Catling, J.I. Zuluaga, J.F. Salazar, H.T. Chen, G. Gonzalez, M.K. Jagadeesh, R. Barnes, C.S. Cockell, J. Haqq-Misra, arXiv:2007.05491 (2020)
3. M. Safonova, J. Murthy, Y.A. Shchekinov, Int. J. Astrobiol. **15**, 93 (2016)
4. J. Krissansen-Totton, S.L. Olson, D.C. Catling, Sci. Adv. **4**, eaao5747 (2018)
5. W.J. Borucki, D. Koch, G. Basri, N. Batalha, T. Brown, D. Caldwell, J. Caldwell, J. Christensen-Dalsgaard, W.D. Cochran, E. DeVore, E.W. Dunham, A.K. Dupree, T.N. Gautier, J.C. Geary, R. Gilliland, A. Gould, S.B. Howell, J.M. Jenkins, Y. Kondo, D.W.M. Latham, W. Geoffrey, S. Meibom, H. Kjeldsen, J.J. Lissauer, D.G. Monet, D. Morrison, D. Sasselov, J. Tarter, A. Boss, D. Brownlee, T. Owen, D. Buzasi, D. Charbonneau, L. Doyle, J. Fortney, E.B. Ford, M.J. Holman, S. Seager, J.H. Steffen, W.F. Welsh, J. Rowe, H. Anderson, L. Buchhave, D. Ciardi, L. Walkowicz, W. Sherry, E. Horch, H. Isaacson, M.E. Everett, D. Fischer, G. Torres, J.A. Johnson, M. Endl, P. MacQueen, S.T. Bryson, J. Dotson, M. Haas, J. Kolodziejczak, J. Van Cleve, H. Chandrasekaran, J.D. Twicken, E.V. Quintana, B.D. Clarke, C. Allen, J. Li, H. Wu, P. Tenenbaum, E. Verner, F. Bruhweiler, J. Barnes, A. Prsa, Science **327**, 977 (2010)
6. N.M. Batalha, J.F. Rowe, S.T. Bryson, T. Barclay, C.J. Burke, D.A. Caldwell, J.L. Christiansen, F. Mullally, S.E. Thompson, T.M. Brown, A.K. Dupree, D.C. Fabrycky, E.B. Ford, J.J. Fortney, R.L. Gilliland, H. Isaacson, D.W. Latham, G.W. Marcy, S.N. Quinn, D. Ragozzine, A. Shporer, W.J. Borucki, D.R. Ciardi, T.N. Gautier III, M.R. Haas, J.M. Jenkins, D.G. Koch, J.J. Lissauer, W. Rapin, G.S. Basri, A.P. Boss, L.A. Buchhave, J.A. Carter, D. Charbonneau, J. Christensen-Dalsgaard, B.D. Clarke, W.D. Cochran, B.-O. Demory, J.-M. Desert, E. Devore, L.R. Doyle, G.A. Esquerdo, M. Everett, F. Fressin, J.C. Geary, F.R. Girouard, A. Gould, J.R. Hall, M.J. Holman, A.W. Howard, S.B. Howell, K.A. Ibrahim, K. Kinemuchi, H. Kjeldsen, T.C. Klaus, J. Li, P.W. Lucas, S. Meibom, R.L. Morris, A. Pša, E. Quintana, D.T. Sanderfer, D. Sasselov, S.E. Seader, J.C. Smith, J.H. Steffen, M. Still, M.C. Stumpe, J.C. Tarter,

P. Tenenbaum, G. Torres, J.D. Twicken, K. Uddin, J. Van Cleve, L. Walkowicz, W.F. Welsh, Astrophys. J. Suppl. **204**, 24 (2013)
7. E.A. Petigura, A.W. Howard, G.W. Marcy, PNAS **110**, 19273 (2013)
8. E. Tasker, J. Tan, K. Heng, S. Kane, D. Spiegel, Nat. Astron. **1**, 0042 (2017)
9. C.J. Shallue, A. Vanderburg, Astron. J. **155**, 94 (2018)
10. A. Méndez, http://phl.upr.edu/hec (2018)
11. S. Agrawal, S. Basak, K. Bora, J. Murthy, in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (IEEE, 2018)
12. K. Bora, S. Saha, S. Agrawal, M. Safonova, S. Routh, A. Narasimhamurthy, Astron. Comput. **17**, 129 (2016)
13. F. Mullally, S.E. Thompson, J.L. Coughlin, C.J. Burke, J.F. Rowe, Astron. J. **155**, 210 (2018)
14. W. Bains, D. Schulze-Makuch, Life **6**, 25 (2016)
15. S. Agrawal, S. Basak, S. Saha, K. Bora, J. Murthy, arXiv:1804.11176 (2018)
16. S. Saha, P. Sarkar, A. Mathur, S. Basak, arXiv:1803.04644 (2018)
17. S. Basak, S. Agrawal, S. Saha, A.J. Theophilus, K. Bora, G. Deshpande, J. Murthy, arXiv:1805.08810 (2018)
18. S. Haykin, in *Neural Networks, A Comprehensive Foundation* (World Scientific Pub Co Pte Lt, 1994), pp. 363–364
19. L. Xiao, R. Lu, Neurocomputing **151**, 246 (2015)
20. A. Narayanan, E.C. Keedwell, J. Gamalielsson, S. Tatineni, Neurocomputing **61**, 217 (2004)
21. G. Cybenko, Math. Control Signals Syst. **2**, 303 (1989)
22. D. Volokin, L. ReLlez, SpringerPlus **723**, 20 (2016)
23. S. Snehanshu, M. Archana, B. Kakoli, B. Suryoday, A. Surbhi, in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (IEEE, 2018)
24. L.N. Irwin, A. Méndez, A.G. Fairén, D. Schulze-Makuch, Challenges **5**, 159 (2014)
25. S. Saha, S. Basak, K. Bora, M. Safonova, S. Agrawal, P. Sarkar, J. Murthy, Astron. Comput. **23**, 141 (2018)
26. J.R. Quinlan, Mach. Learn. **1**, 81 (1986)
27. L. Breiman, Mach. Learn. **24**, 41 (1996)
28. E. Strubell, A. Ganesh, A. McCallum, arXiv:abs/1906.02243 (2019)
29. A. Cassan, D. Kubas, J.-P. Beaulieu, et al., Nature **481**, 167 (2012)
30. L.E. Strigari, M. Barnabè, P.J. Marshall, R.D. Blandford, Mon. Not. R. Astron. Soc. **423**, 1856 (2012)
31. http://kepler.nasa.gov/
32. N.M. Batalha, Proc. Natl. Acad. Sci. **111**, 12647 (2014)
33. K.I. Öberg, V.V. Guzmán, K. Furuya, et al., Nature **520**, 198 (2015)
34. G. Gonzalez, D. Brownlee, P. Ward, Icarus **152**, 185 (2001)
35. P. Dayal, C. Cockell, K. Rice, A. Mazumdar, Astrophys. J. Lett. **810**, L12 (2015)
36. D. Schulze-Makuch, A. Méndez, A.G. Fairén, et al., Astrobiology **11**, 1041 (2011)
37. L.N. Irwin, A. Méndez, A.G. Fairén, D. Schulze-Makuch, Challenges **5**, 159 (2014)
38. Y.A. Shchekinov, M. Safonova, J. Murthy, Astrophys. Space Sci. **346**, 31 (2013)
39. S.-S. Huang, Publ. Astron. Soc. Pac. **71**, 421 (1959)
40. J.F. Kasting, Science **259**, 920 (1993)
41. L.N. Irwin, D. Schulze-Makuch, *Cosmic Biology* (Springer-Praxis, New York, 2011)
42. R. Heller, J. Armstrong, Astrobiology **14**, 50 (2014)
43. R.A. Wittenmyer, M. Tuomi, R.P. Butler, et al., Astrophys. J. **791**, 114 (2014)
44. A. Méndez, http://phl.upr.edu/library/notes/athermalplanetaryhabitability classificationforexoplanets (2011)
45. D. Schulze-Makuch, A. Méndez, A.G. Fairén, P. von Paris, C. Turse, G. Boyer, A.F. Davila, M.R. de Sousa António, D. Catling, L.N. Irwin, Astrobiology **11**, 1041 (2011)
46. http://phl.upr.edu/projects/habitable-exoplanets-catalog/data/database
47. J.S. Denker, Physica D **22**, 216 (1986)
48. S.-I. Amari, Neurocomputing **5**, 185 (1993)

49. N.B. Peng, Y.X. Zhang, Y.H. Zhao, Sci. Chin. Phys. Mech. Astron. **56**, 1227 (2013)
50. R.O. Duda, P.E. Hart, D.G. Stork, *Pattern Classification* (Wiley, New York, 2001)
51. N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, J. Artif. Intell. Res. **16**, 321 (2002)
52. J.T. Springenberg, arXiv:1511.06390 (2015)
53. T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, in *Proceedings of the 30th International Conference on Neural Information Processing Systems* (2016), pp. 2234–2242
54. T. Bergstrom, Economics 100B, www.econ.ucsb.edu/~tedb/Courses/Ec100BS06/PPSlides/Ch19.ppt (2007)
55. A. Mathur, S. Saha, GitHub Repository, https://github.com/mathurarchana77/A-RELUandSBAF
56. S. Makhija, S. Saha, S. Basak, M. Das, Astron. Comput. **29**, 300 (2019)
57. S. Sridhar, A. Sheikh, S. Saha, R. Yedida, S. Saha, in *Int. Joint Conference on Neural Networks* (2020)
58. E. Parzen, Ann. Math. Statist. **33**, 1065 (1962)
59. S. Saha, P. Sarkar, A. Mathur, S. Basak, J. Sci. Res. **7**, 48 (2018)
60. B.E. Rhoades, Trans. Am. Math. Soc. **226**, 257 (1977)
61. S. Saha, J. Sarkar, A. Dwivedi, N. Dwivedi, A.M. Narasimhamurthy, R. Roy, J. Cloud Comput. **5**, 1 (2016)
62. D. Hájková, J. Hurnik, Czech J. Econ. Finance (Finance a uver) **57**, 465 (2007)
63. D.-M. Wu, Econometrica **43**, 739 (1975)
64. M. Hossain, A. Majumder, T. Basak, Open J. Statist. **2**, 460 (2012)
65. A. Hassani, M.Sc. thesis, University of Nebraska, Lincoln, 2012
66. J. Felipe, F.G. Adams, Eastern Econ. J. Eastern Econ. Assoc. **31**, 427 (2005)
67. C.W. Cobb, P.H. Douglas, Am. Econ. Rev. **18**, 139 (2012)
68. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, J. Mach. Learn. Res. **12**, 2825 (2011)
69. A. Méndez, http://phl.upr.edu/library/notes/syntheticstars (2011)
70. P. Ramachandran, B. Zoph, Q.V. Le, *Neural and Evolutionary Computing* (2017)
71. F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation Forest, in *2008 Eighth IEEE International Conference on Data Mining* (December 2008), pp. 413–422
72. V. Chandola, A. Banerjee, V. Kumar, ACM Comput. Surv. **41**, 15 (2009)
73. F.T. Liu, K.M. Ting, Z.-H. Zhou, ACM Trans. Knowl. Discovery Data, **6**, 1 (2008)
74. M.C. Turnbull, W.A. Traub, K.W. Jucks, N.J. Woolf, M.R. Meyer, N. Gorlova, M.F. Skrutskie, J.C. Wilson, Astrophys. J. **644**, 551 (2006)
75. D.A. Zighed, G. Ritschard, S. Marcellin, in *Advances in Intelligent Information Systems* (Springer, Berlin, Heidelberg, 2010), pp. 27–42
76. S. Saha, K. Bora, S. Basak, G. Srinivasa, M. Safonova, J. Murthy, S. Agrawal, *Ebook-Astroinformatics Series Machine Learning in Astronomy: A Workman's Manual* (ResearchGate, 2018)
77. A.S. Nemirovski, M.J. Todd, Acta Numer. **17**, 191 (2008)
78. G. Ginde, S. Saha, A. Mathur, S. Venkatagiri, S. Vadakkepat, A. Narasimhamurthy, B.S. Daya Sagar, Scientometrics **108**, 1479 (2016)
79. G. Ginde, S. Saha, C. Balasubramaniam, R.S. Harsha, A. Mathur, B.S. Dayasagar, M.N. Anand, in *Proceedings of the fourth national conference of Institute of Scientometrics* (SIoT, 2015)
80. K. Mohanchandra, S. Saha, K. Srikanta Murthy, G.M. Lingaraju, Int. J. Intell. Eng. Inf. **3**, 313 (2015)
81. V.N. Vapnik, A.Y. Chervonenkis, Autom. Remote Control **1**, 103 (1964)
82. C. Corinna, V. Vladimir, Mach. Learn. **20**, 273 (1995)
83. L. Khaidem, S. Saha, S. Basak, S. Roy Dey, ResearchGate, https://www.researchgate.net/publication/301818771_Predicting_the_direction_of_stock_market_prices_using_random_forest (2016)

84. D. Schulze-Makuch, W. Bains, Nat. Astron. **2**, 432 (2018)
85. L. Irwin, A. Méndez, A. Fairén, D. Schulze-Makuch, Challenges **5**, 159 (2014)
86. J.J. Swift, J.A. Johnson, T.D. Morton, et al., Astrophys. J. **764**, 105 (2013)
87. R. Yedida, S. Saha, arXiv:1902.07399 (2019)
88. M. Rosenblatt, Ann. Math. Statist. **27**, 832 (1956)
89. L. Breiman, Random Forests Mach. Learn. **45**, 5 (2001)
90. A.S. Younger, S. Hochreiter, P.R. Conwell, *Meta-Learning With Backpropagation* (IEEE, 2001)
91. E.N. Lorenz, J. Atmos. Sci. **20**, 130 (1963)
92. K.T. Alligood, T.D. Sauer, J.A. Yorke, *Chaos* (Springer, Berlin, 1996)
93. R. Devaney, *An Introduction to Chaotic Dynamical Systems* (CRC Press, Boca Raton, 2018)
94. S.H. Strogatz, in *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*, 2nd edition (Westview Press, 2015), pp. 1–528
95. M. Barnsley, R. Devaney, K. Falconer, V. Kannan, V. Kumar, *Fractals, Wavelets, and their Applications* (Springer, 2014)
96. K. Dajani, C. Kraaikamp, in *Carus Mathematical Monographs* (Mathematical Association of America, 2002), pp. 1–190
97. H. Korn, P. Faure, C.R. Biol. (Elsevier) **326**, 787 (2003)
98. P. Faure, H. Korn, C.R. Acad. Sci.-Ser. III-Sci. Vie (Elsevier) **324**, 773 (2001)
99. A. Zerroug, L. Terrissa, A. Faure, Ann. Rev. Chaos Theory Bifurc. Dyn. Syst. **4**, 55 (2013)
100. J.C. Sprott, Nonlinear Dyn. Psychol. Life Sci. **17**, 223 (2013)
101. H.N. Balakrishnan, A. Kathpalia, S. Saha, N. Nagaraj, Chaos **29**, 113125 (2019)
102. A. Mendez, Exoplanet Detection Methods Visualized updated Aug 10, 2014 http://phl.upr.edu/library/media/exoplanetdetectionmethodsvisualized