Annals of Mathematics and Artificial Intelligence (2019) 87:361–372
https://doi.org/10.1007/s10472-019-09657-3

A linear relation between input and first layer in neural networks

Sebastián A. Grillo[1]

Published online: 8 August 2019

Abstract

Artificial neural networks grow on the number of applications and complexity, which require a minimization on the number of units for some practical implementations. A particular problem is the minimum number of units that a feed forward neural network needs on its first layer. In order to study this problem, it is defined a family of classification problems following a continuity hypothesis, where inputs that are close to some set of points may share the same category. Given a set S of $k-$dimensional inputs and let $\mathscr{N}$ be a feed forward neural network that classifies any input in S within a fixed error, there is proved that $\mathscr{N}$ requires $\Theta(k)$ units in the first layer, if $\mathscr{N}$ can solve any instance from the given family of classification problems. Furthermore, this asymptotic result is optimal.

Keywords Neural networks · Computational learning theory · Machine learning

Mathematics Subject Classification (2010) 00A69

1 Introduction

Despite the enormous number of applications, artificial neural networks (ANN) are not fully understood from a theoretical point of view and the development of new algorithms has a big empirical dependence [17]. Principally, Deep Learning has several applications and that reignites the importance of understanding the success and limitations of ANN [5, 16]. As an example of theoretical challenge, there is controversy about the role of depth in ANN [18].

Another problem is determining the resources that an algorithm requires for achieving a classification task. This problematic intersects Computational Complexity with Machine Learning and requires relating and restricting our computational measures [3]. For Machine Learning, we may be interested in the size of samples [2] for some classification problems. In this sense, Computational Learning Theory offers frameworks like Vapnik—Chervonenkis Theory [22] and Probably Approximately Correct Learning [21]. But, before we look for an algorithm that is efficient in time for learning some parameters, we must

✉ Sebastián A. Grillo
sgrillo@uaa.edu.py

[1] Faculty of Science and Technology, Universidad Autónoma de Asunción, Asunción, Paraguay

answer if such parameters exist for the classification problem. For ANN, this is equivalent to asking if a given set of network instances is able to represent some function, where such set depends on some complexity measure. Some important measures for a network architecture are the number of units [11] or the size of weights [9]. Minimizing both measures are important problems, as ANN are progressively applied to more complex tasks that require an enormous computational effort [8, 15].

In this work, there is a different complexity measure for ANN. That is the number of units in the first layer. A practical reason for selecting such measure is that it gives us information about the structure of the architecture. Therefore, this measure is complementary to others, which is helpful for designing an architecture for a given problem. The first layer determines which inputs have the potential of receiving different outputs. Because, if the first layer does not have enough units, then some inputs may be indistinguishable for the following layers. In this sense, the first layer can act as a bottleneck, that controls the amount of information processed by the ANN. Thus, the input dimension may be reduced in relation to the output layer, which implies information compression and relates this problem with auto-encoders [10] and manifold learning [4]. We also have that other layers can be analyzed as first layers, where input is the output of lower layers. That implies that understanding the first layer gives us a view of the rest of the ANN. Therefore, a better understanding of individual layers implies a step towards understanding the action of many layers within a deep architecture.

In order to study first layer complexity, we need a previous model for classification problems. For example, the VC-dimension considers the capacity of shattering or separating some sets of points in $\mathbb{R}^k$ [22]. As computers have a finite capacity for representing numbers, a realistic model may consider a discrete learning problem. In this sense, we assume that our input is a vector of size k and the variable for such k terms is a finite geometric sequence A. However, this model requires the classification of all possible inputs, that is not always necessary, but it is a task that we expect from strong artificial intelligence [13]. That is a general intelligence able to classify an input as noisy or non-sense, if no previous pattern is found. Thus, we have the domain of the learning problem, but another important consideration is the smoothness of the problem. That is, that two similar inputs may share the same category in the classification. In this sense, the model is completed by assuming that our classification problem is a function f with domain S, that depends on a continuity parameter β and a set of inputs C in S. The function f satisfies that any input x have the same category as $c \in C$ $(f(c) = f(x))$ if $d_1(x, c) < \beta$, for *taxicab distance* d_1. This work presents the following main result. Let k be the size of our input. The first layer in a feed forward neural network requires $\Theta(k)$ units (see asymptotic notation [7]), for solving any possible instance of a learning problem that follows the given hypothesis. The proof assumes a bound constant error in the classification and a big family of potential activation functions for the first layer. Notice that this is an analysis of the first layer size as input dimension grows to infinity. This result suggests that first layers need to be proportional to input, when the learning problem satisfies all the following properties.

1. The number of categories is unbounded.
2. All input has a category, even a 'nonsense category'.
3. As input dimension tends to infinity, any set of inputs representing a category has a bound cardinality.

A secondary contribution of this paper is the model for classification problems with parameterized continuity. This framework can be applied for studying other complexity measures on neural networks or machine learning problems, as well.

The structure of the paper is as follows. In Section 2 there is a formulation of the classification problem and a technical lemma. In Section 3, we introduce our formulation for feed-forward neural networks and more technical lemmas. In Section 4, there is the main result, proofs and an example. Finally, in Section 5 we have an analysis about the applicability and possible extensions of these results.

2 A model for learning tasks

We state a formulation for classification problems. Let $A = \{a_i\}$ be a finite sequence, where $a_{i+1} - a_i = \Delta$ for all i and $|A| = n$. We denote a set $S = A^k$ for $0 < k \in \mathbb{N}$. The set S represents all $k-$dimensional inputs, whose terms takes values from A. Notice that A is an ordered set, which allows the representation of intensity. Thus, each x_i term of $x \in S$ can represent a pixel from an image or a quantity from a discrete-time signal. Suppose that each possible input x must be classified within a category, we denote T as a set of possible categories such that $\emptyset \in T$ which denotes the *null* output. An *objection function* f is a mapping $S \to T$, that must be emulated by a learning algorithm L. However, assuming no error of L is unrealistic. Thus, we need to measure the approximation of some learning algorithm L to f.

Definition 1 Denoting $L(x)$ as the output of algorithm L for input x. An algorithm L approximates a function $f : S \to T$ within error ε, if $L(x) = f(x)$ for all $x \in R \subset S$, where $|R| \geq (1 - \varepsilon)|S|$.

Notice that this notion of approximation only considers the proportion of inputs with correct output. As T is not an ordered set, an output only can be correct or incorrect. Another consideration is the kind of functions $f : S \to T$ that we can find as an objective in real learning problems. For example, if we change just few pixels in some image representing an object, it is likely that such object is still recognizable. Thus, we would expect that two points x, y that are 'close' in S probably satisfy $f(x) = f(y)$. This implies that S presents disjoint 'compact' subsets with constant values in f. A model for such learning problem needs a definition of distance between inputs x, y, because such model must decide which input is close from another. As each term x_i from input x is a value from the ordered set A, suppose that we obtain a new input y from x by changing the 'intensity' of some terms. Then, an additional hypothesis is that we measure the difference between two inputs x and y by the times that we change the intensity on the terms x_i, in order to transform x to y. Figure 1 shows an example with a 4 × 4 image, where we have the right original element

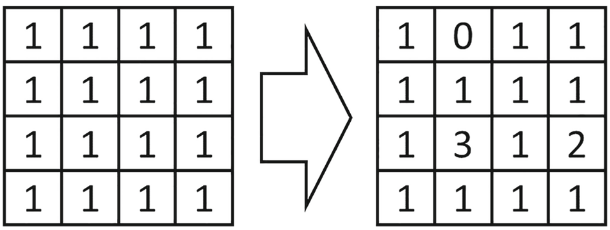

Fig. 1 The left image can be obtained from the right image; by reducing once on some pixel, increasing twice on another pixel and increasing once on a last pixel. Therefore, we have a distance of value 4 between both inputs

from S with all pixels equal to 1 and a left image by changing some pixels. That is formally the *taxicab distance*, which is denoted as

$$d_1(x, y) = \sum_{i=1}^{n} |x_i - y_i| .$$

A final hypothesis for this model is that some elements in S, are highly recognizable in their category. Thus, if some input is not too distant from some highly recognizable input, then they share the same category. The following definition formalizes this idea, by introducing a notion of 'neighborhood' covering a highly recognizable input that determines a category. Figure 2 presents the intuition behind.

Definition 2 Let $x \in A$, $R \subseteq S$, $c \in R$, f an objection function and $\beta \in \mathbb{R}$. If

- $x \in R$ implies $f(x) = f(c)$, and
- $d_1(c, x) \leq \beta$ if and only if $x \in R$

then R is *β-regular* in relation to f and c is the center of R.

Using the following definition, we complete our model for classification problems. It basically defines a function whose domain contains $\beta-$regular sets. If some input is outside of such $\beta-$regular sets, then it has a special null category. This category models an input with no sense, like a noisy input.

Definition 3 Let $C \subseteq S$, f an objection function, $\beta \in \mathbb{R}$ such that, for each $c \in C$, there is $\beta_c \in \mathbb{R}$ such that $\beta_c \geq \beta$ and c is the center of some $R_c \subseteq S$ that is β_c-regular in relation to f. If $f(x) = \emptyset$ for any $x \notin (\bigcup_{c \in C} R_c)$ then f is β-regular.

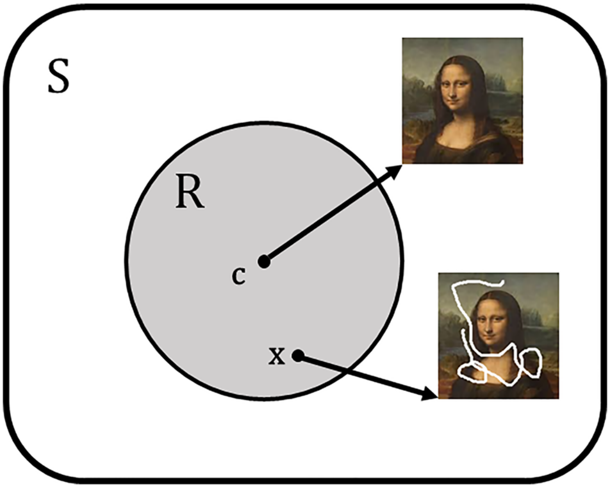

Fig. 2 A $\beta-$regular set $R \subset S$ models a set of inputs that are associated to some category from T. The input c represents the most easily recognizable element in R. Other inputs x in R may represent noisy variations of c, that are still recognizable in the category. The union of several $\beta-$regular sets can form a super-set under the same category, in such case the centers in each $\beta-$regular set may represent different states of the same object, like a shape in different positions

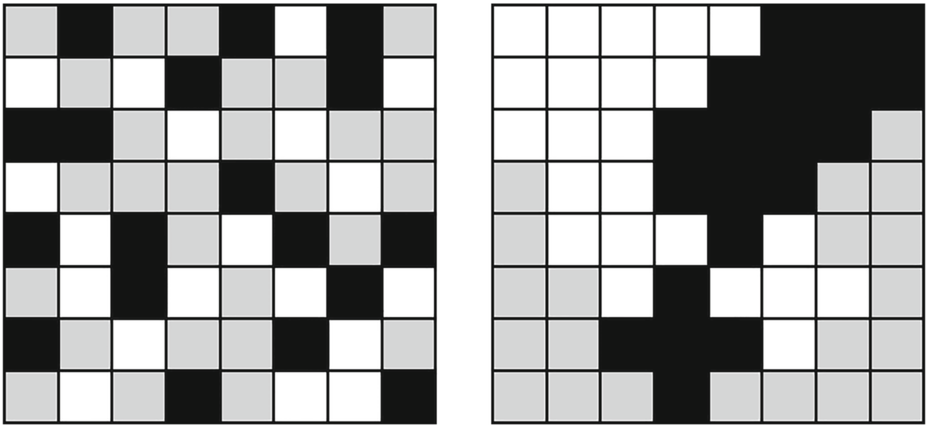

Fig. 3 The grids represent the discrete functions f at left and g at right, where the black boxes, the white boxes and gray boxes represent inputs with output 1,0 and $\varnothing$, respectively. Notice that both f and g are $0-$regular, however only g is $1-$regular. This produces that g cannot have patterns as thin as f

For example, consider a discrete-time signal with just two terms, where $A = \{1, 2, ..., 8\}$ and $T = \{0, 1, \varnothing\}$. Figure 3 shows a $0-$regular function $f : S \to T$ and a $1-$regular function $g : S \to T$. Notice that two $\beta-$regular sets with different centers are no necessarily disjoint, but they must share the same category if they are not disjoint. Thus, a set $\{x : f(x) = t\}$ can present different shapes in S depending on $t \in T$. Also notice that Definition 3 does not depend on a particular C, if some C exists then f is β-regular. Non-gray boxes of the left grid in Fig. 3 represent C for g, however Fig. 4 shows a set C for the same g of Fig. 3.

We denote the cardinality of the image of a function $f : S \to T$ as $\mathscr{I}_f = |\{f(x) : x \in S\}|$. This notation will be useful, starting from the following lemma.

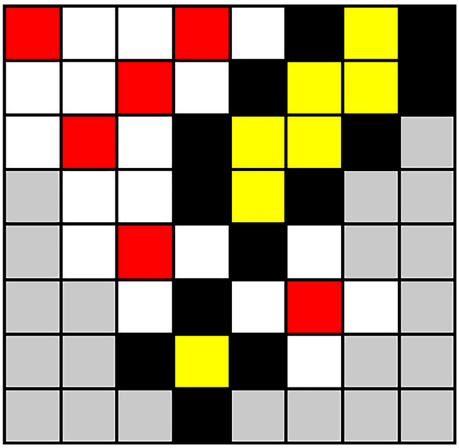

Fig. 4 The grid represents the function g with its set C. The set C is the union of red and yellow boxes. Black and yellow boxes represent inputs with output 1, white and red boxes represent inputs with output 0 and gray boxes represent inputs with output $\varnothing$

Lemma 1 *If $\beta > 0$ satisfies $\frac{k\Delta}{2\beta+\Delta} \geq 2$, then there exists a $\beta-$regular function $f : S \to T$, such that*

$$\mathscr{I}_f \geq |A|^{\left(\frac{k\Delta}{2\beta+\Delta}-1\right)}. \tag{1}$$

Proof We need to prove that we can have at least $|A|^{\frac{k\Delta}{2\beta+\Delta}}$ disjoint subsets of S, that are $\beta-$regular in relation to some $\beta-$regular function f. Since, assigning a different output to each $\beta-$regular subset we will satisfy (1).

The existence of such $\beta-$regular sets can be obtained from a set $C \subset S$, satisfying the following properties.

- First,

$$|C| \geq |A|^{\left(\frac{k\Delta}{2\beta+\Delta}-1\right)}. \tag{2}$$

- Second, $c, d \in C$ and $c \neq d$ implies that

$$d_1(c, d) \geq \gamma, \tag{3}$$

  for $\gamma = 2\left\lfloor \frac{\beta}{\Delta} \right\rfloor + 1$.

Those properties guarantee the existence of the $\beta-$regular sets, since each $c \in C$ can be taken as the center of each $\beta-$regular set.

We construct a set C satisfying (2) and (3), by the following two properties on its elements.

1. If $c \in C$ and c_i is the $i-$th component of c, then

$$c_{(1+i\gamma)} = c_{(2+i\gamma)} = ... = c_{(i+1)\gamma},$$

   for $i \geq 0$. Notice that we are dividing vectors c from C in constant blocks of size γ, that may change jointly in relation to another vector. This property guarantees (3); because if $c \neq d$, then they must be different at least in one block of size γ.
2. If $c \in C$ and

$$i > \gamma \left\lfloor \frac{k}{\gamma} \right\rfloor$$

   then $c_i = a_1$, where $a_1 \in A$. This property comes from the fact k is no always divisible by γ, therefore we may have a remaining block of size smaller than γ. For simplicity, we keep constant such remaining blocks.

We count the number of vectors in C satisfying last properties, this construction implies that

$$|C| \geq |A|^{\left\lfloor \frac{k}{\left(2\left\lfloor \frac{\beta}{\Delta} \right\rfloor+1\right)} \right\rfloor},$$

and (2) is satisfied. □

This technical lemma is applied in Section 4. Lemma 1 gives a lower bound to the maximum number of categories in β-regular functions.

3 Considerations about neural networks

This section specifies the learning algorithm that is analyzed within the model for classification problems. We consider classes of feed-forward neural networks (FFNN), whose definitions depend on the activation function applied by the first layer units. We denote G the set of activation functions containing all $\mathbf{a} : \mathbb{R} \to [-1, 1]$ non-constant monotonic functions, such that $\mathbf{a}(x) = \pm 1$ when $x \to \pm\infty$.

Definition 4 If the first layer of some FFNN has a function $f \in \mathbf{G}$ as activation function of its units, then we say that such FFNN is an *initial* $\mathbf{G}-$*class FFNN*.

We denote a threshold function $th(x)$ by $th(x) = 1$ for $x \geq 0$ and $th(x) = 0$ otherwise. A unit applying an activation function $th(x)$ is denoted as a threshold unit.

Definition 5 If the first layer of some FFNN is entirely composed by threshold units, then we say that it is an *initial threshold FFNN*.

This class of FFNN is still more general than the following.

Definition 6 If some feed-forward neural network is entirely composed by threshold units, then we say that it is a *threshold FFNN*.

Notice that all threshold FFNN is an initial threshold FFNN. However an initial threshold FFNN is not a threshold FFNN if there is some unit in upper layers that is not a threshold unit.

Another important concept is the following measure.

Definition 7 The base of an FFNN is the number of neurons that appear in its first layer.

We state the next optimization problem, determine the minimum base of an FFNN able to approximate any function $f : S \to T$. We are not considering if the FFNN is able to reach such function by a training method, for our purposes the FFNN can approximate f if there exist appropriate weights and biases.

Notice that the output of the first layer partitions S in subsets. In Section 4 we apply a combinatorial approach on such partitions. That is, we count the number of different outputs produced by the first layer of the FFNN. A unit with sign function as activation can divide the space in no more than 2 subsets, by giving a label in $\{1, -1\}$ for each subset. In return, if m units apply the sign function, then we label subsets using elements from $\{1, -1\}^m$. For units with activation function in $\mathbf{G}$, we can have an infinite number of outputs. However, the units depend on a finite physical device, and we assume that such device can represent a fixed number V of different outputs by any kind of unit. Therefore, a unit with activation in $\mathbf{G}$ cannot divide the space in more than V subsets. The following lemma shows a simulation of a given unit with activation in $\mathbf{G}$ by several threshold units.

Lemma 2 *Consider a unit* $\mathbf{u}$ *on the first layer, whose output for input* x *is*

$$\mathbf{a}\left(\sum_{i=1}^{n} w_i x_i + b\right),$$

where $b, w_i \in \mathbb{R}$ *and* $\mathbf{a} \in \mathbf{G}$. *If the computing device implementing* $\mathbf{u}$ *can represent just* V *elements from set* $[0, 1]$, *then* V *threshold units on the first layer divide* S *in the same partition.*

Proof The partitions in S come from the discretization of function $\mathbf{a}$ in $\mathbb{R}$, by considering just V elements from set [0, 1]. Therefore, the computer represents a discrete function $\tilde{\mathbf{a}}$, where $\tilde{\mathbf{a}}\left(y_i\right) = z_i$. Given a constant K, $\mathbf{a}\left(\sum_{i=1}^{n} w_i x_i + b\right) = K$ denotes a hyperplane and different constants K_1, K_2 denote a set of parallel hyperplanes. This implies that $\mathbf{u}$ cuts S by no more than V parallel hyperplanes, denoting such set of hyperplanes as H. Thus, for each k_j where $\tilde{\mathbf{a}}\left(y_{k_j}\right) \neq \tilde{\mathbf{a}}\left(y_{k_j-1}\right)$, we denote a unit $\mathbf{v}_j$ with output

$$th\left(\sum_{i=1}^{n} w_i x_i + b_j\right),$$

such that $b_j = b - z_j$. Considering that each hyperplane in H divides S equivalently to some unit $\mathbf{v}_j$, we obtain the same partition of S. □

Notice that this simulation produces the same partition, but it changes the outputs of the first layer. While $\mathbf{u}$ labels subsets in S using elements from [0, 1], the set $\left\{\mathbf{v}_j\right\}$ labels subsets in S using elements from $\{1, -1\}^{(V)}$.

Before the results, there is a last technical lemma.

Lemma 3 *Let $R_k(m)$ be the maximum number of subsets that m $(k-1)$-dimensional hyperplanes cut in $\mathbb{R}^k$, then*

$$R_k(m) < \sqrt{\frac{2}{\pi k}}\left(\frac{me}{k}\right)^k \tag{4}$$

or

$$R_k(m) \leq 2^m. \tag{5}$$

Proof We have the identity

$$R_k(m) = 2\sum_{i=0}^{k-1}\binom{m-1}{i}, \tag{6}$$

for any positive m [19]. However, if $m > k$, we have

$$R_k(m) < 2\frac{m^k}{k!}. \tag{7}$$

Taking (7) and applying the Stirling's approximation in its upper bound formulation [6], we obtain (4). If $m \leq k$, then (6) becomes (5). □

Lemma 3 gives an upper bound to the number of different outputs that can be produced by a threshold layer.

4 Results

The following results show asymptotic relations between base, dimension and the size of our input set. Thus, we analyze the base when Δ, β, n are constant and k grows to infinity.

Theorem 1 *Let m be the base of some initial threshold FFNN $\mathcal{N}$. Suppose that $\mathcal{N}$ can approximate any $\beta-$regular function $f : S \to T$ within error ε. If $k \to \infty$ then*

$$m \in \Omega(k), \tag{8}$$

(see asymptotic notation [7]).

Proof As we discussed in Section 3, the output given by the first layer represents a partition on S. Consider the subsets in S, that are delimited by hyperplanes defined by each threshold unit from the first layer. If two inputs x, y have different final outputs by $\mathscr{N}$, then they are contained in different subsets of our first layer partition, otherwise they are indistinguishable for the following layers. Therefore, the analysis involves considering the maximum number of subsets that a first layer must delimit.

An error ε implies that our first layer must delimit at least $\lceil \mathscr{I}_f(1-\varepsilon) \rceil$ subsets in S. Considering that our FFNN can approximate any $\beta-$regular function $f : S \to T$, we have

$$R_k(m) \geq \left\lceil \sup_{f \in \mathscr{F}_{S,T}^{\beta}} \left\{ \mathscr{I}_f \right\} (1-\varepsilon) \right\rceil, \tag{9}$$

where $\mathscr{F}_{S,T}^{\beta}$ is the set of $\beta-$regular functions $f : S \to T$ and

$$\sup_{a \in A} \{b\}$$

denotes the supremum value of b depending on a in domain A.

By Lemma 1, there is

$$\sup_{f \in \mathscr{F}_{S,T}^{\beta}} \left\{ \mathscr{I}_f \right\} \geq |A|^{\left(\frac{k\Delta}{2\beta+\Delta}-1\right)}. \tag{10}$$

Applying Lemma 3 and (10) to (9), we have two possible cases depending on (4) or (5).

1. If we consider (4), then

$$\sqrt{\frac{2}{\pi k}} \left(\frac{me}{k}\right)^k > |A|^{\left(\frac{k\Delta}{2\beta+\Delta}-1\right)} (1-\varepsilon). \tag{11}$$

   Solving the last inequality for m, we obtain

$$m > |A|^{\left(\frac{\Delta}{2\beta+\Delta}-\frac{1}{k}\right)} \frac{k\sqrt[k]{1-\varepsilon}}{e} \sqrt[2k]{\frac{\pi k}{2}}, \tag{12}$$

   which can be rewritten as

$$m > kg(k), \tag{13}$$

   where

$$g(k) = |A|^{\left(\frac{\Delta}{2\beta+\Delta}-\frac{1}{k}\right)} \frac{\sqrt[k]{1-\varepsilon}}{e} \sqrt[2k]{\frac{\pi k}{2}}.$$

   As

$$\lim_{k\to\infty} g(k) = \frac{|A|^{\left(\frac{\Delta}{2\beta+\Delta}\right)}}{e},$$

   we obtain the asymptotic relation from (13).
2. If we consider (5), then

$$2^m \geq |A|^{\left(\frac{k\Delta}{2\beta+\Delta}-1\right)} (1-\varepsilon), \tag{14}$$

   which implies the same lower bound.

□

The following theorem extends the linear lower bound to a big family of activation functions. However, we assume a constant arithmetic precision in our computer device as k grows.

Theorem 2 *Let m be the base of some initial* $\mathbf{G}-$*class FFNN* $\mathscr{N}$. *Suppose that (i)* $\mathscr{N}$ *can approximate any* $\beta-$*regular function* $f : S \to T$ *within error* ε *and (ii) the computer device* $\mathscr{D}$ *implementing* $\mathscr{N}$ *can represent just* V *elements from set* $[0, 1]$. *If* $k \to \infty$ *then*

$$m \in \Omega(k). \tag{15}$$

Proof The proof is similar to Theorem 1. Let $\mathbf{a} \in \mathbf{G}$ be the activation function of the first layer from $\mathscr{N}$. Let $\bar{R}_k(m)$ be the maximum number of subsets defined by m units, whose activation function is $\mathbf{a}$, then

$$\bar{R}_k(m) > \left\lceil \sup_{f \in \mathscr{F}^{\beta}_{S,T}} \left\{ \mathscr{I}_f \right\} (1-\varepsilon) \right\rceil. \tag{16}$$

Considering the hypothesis about the implementation of $\mathscr{N}$, each unit on the first layer defines no more than V different subsets on S. By lemma 2, a partition of S by some unit with activation function $\mathbf{a}$ can be simulated by V threshold units. Therefore

$$R_k(Vm) > \bar{R}_k(m),$$

that implies

$$R_k(Vm) > \left\lceil \sup_{f \in \mathscr{F}^{\beta}_{S,T}} \left\{ \mathscr{I}_f \right\} (1-\varepsilon) \right\rceil \tag{17}$$

by (16). Equation (17) implies expression (15), by following analogous steps in the proof of Theorem 1. □

An immediate question about theorems 1 and 2 is the tightness of such lower bounds. The following theorem tackles such question.

Theorem 3 *There exists a threshold FFNN* $\mathscr{N}$ *with base* $m = k|A|$, *that is able to approximate any function* $f : S \to T$ *without error.*

Proof Taking $[i] = \{1, 2, ..., i\}$, we denote any first layer unit by u_{pq}, where $p \in [k]$ and $q \in [|A| - 1]$. We denote the output of unit u_{pq} on input $x \in S$, as $u_{pq}(x) \in \{0, 1\}$. If we choose the appropriate weights on $\mathscr{N}$, then we have that $u_{pi}(x) = 1$ for all $i < q$, if and only if $x_p = a_q$. Therefore, the first layer transforms the input x into a boolean vector $y \in \{0, 1\}^{k(|A|-1)}$. The idea is mapping the value of each term x_i by a unary code represented in $\hat{y}_i = y_{i1}y_{i2}...y_{i(|A|-1)}$, where $x_i = a_j$ if and only if the first $j - 1$ values y_{ij} are equal to one. Thus, this transformation defines a bijection between S and the set of matrices $y = \hat{y}_1\hat{y}_2...\hat{y}_k$, where each $\hat{y}_k$ is of the form 11...10...00.

We denote the output units of $\mathscr{N}$ as o_t for $t \in T$. The FFNN outputs $t \in T$ on input x, if all outputs units but o_t are equal to zero. Notice that within this configuration, we can compute $f : S \to T$ by $\mathscr{I}_f$ independent FFNNs $\mathscr{N}_t$, where each $\mathscr{N}_t$ computes a boolean function $f_t : S \to \{0, 1\}$ for $t \in T$, that represents the output of o_t on x. Thereby, each FFNN $\mathscr{N}_t$ takes y as input and activates o_t if and only if $f(x) = t$. That is possible because a threshold FFNN can compute any boolean function [1]. □

The construction described in the proof of last theorem is far from optimal, if we consider the number of threshold units applied. However, Theorem 3 proves that a linear base in relation to k is the minimum required for computing any $\beta-$regular function, for initial threshold FFNN. The following theorem gives an analogous result for the initial $\mathbf{G}-$class case.

Theorem 4 *For any* $\mathbf{a} \in \mathbf{G}$*, there exists an FFNN* $\mathcal{N}$ *that (i) applies the activation function* $\mathbf{a}$ *by its first layer, (ii) has a base* $m = k|A|$*, and (iii) is able to approximate any function* $f : S \to T$ *without error.*

Proof Denoting (i) $\{w_{ij}\}$ as the set of weights for connections between input and the first layer and (ii) $\left\{w'_{ij}\right\}$ as the set of weights for connections between second and first layer. Notice that giving appropriate values for $\{w_{ij}\}$ and $\left\{w'_{ij}\right\}$, we can simulate the first threshold layer from Theorem 3. The rest of the FFNN is identical to the construction from Theorem 3. □

5 Conclusions

We have a new problem in literature. That is the required number of units on the first layer for a classification task. Theorems 1, 2, 3 and 4 show that a constant error and continuity parameter allow the existence of problems where base complexity is proportional to input dimension.

The second contribution on this work appears in Section 2. This result offers a more complete framework for analyzing machine learning algorithms than boolean functions [1, 3], because $\beta-$regular functions can define continuity and boolean functions are a subset of this family of functions. The form that this work applies $\beta-$regular functions, implies the following assumptions.

1. The algorithm classifies all inputs, even with a 'no sense' label if this is the case.
2. The number of categories is unbounded, as input size tends to infinity. Thus, we may better identify this model with clustering or non-supervised classification [12]. This assumption has an alternative interpretation, 'more input implies more information'.

These considerations contrast with some learning problems, where potential inputs represent small subsets in relation to S or categories have a limited cardinality. However, this framework looks for understanding a general artificial intelligence able to classify any input and the number of labels given by language is unlimited. Another important discussion is the application of $\beta-$regular functions. This class of functions applies a restriction to learning problems, under a continuity hypothesis. However, this class of functions depends on some continuity parameter β and the asymptotic results are invariant for any β, if β is constant. We can see that bigger β parameters reduce the set of possible learning problems and if β is small enough, we are considering all functions $f : S \to T$.

Theorem 2 applies an extra hypothesis that other theorems in this paper ignore. That is the finite numerical capacity of some physical device. This hypothesis is necessary for applying the combinatorial approach in the proofs. Thus, avoiding such hypothesis may require a more complicated geometrical approach.

As we have asymptotic results and most ANN implementations consider fixed input sizes, there is no sense in comparing both in most cases. However, *convolutional neural networks* (CNN) apply filters of size $i \times i$ by their first layers, where such i has low values [14, 20]. If i is constant in relation to k, then a CNN architecture maintains the linear relation.

Some good continuations of this work are the following points:

- How appropriate is $\beta-$regularity for modeling real problems? Other distances than taxicab distance may be more realistic?
- The lower bounds ignore partial functions on S. An extended result can be a lower bound that considers the size of the domain in such function.
- Some learning problems involving images and sounds seem to be enhanced when intensity is increased, like more contrast or sound volume, respectively. If we combine such hypothesis with $\beta-$regularity, how can base complexity change?
- In relation to manifold learning, we found that $\beta-$regularity and unbounded categories may imply problems where no up-to-factor compression is possible. However, the linear relation can be broken if the continuity parameter β depends on input size k. Can we obtain asymptotically tight bounds for such hypothesis?

References

1. Anthony, M.: Boolean Functions and Artificial Neural Networks. Centre for Discrete and Applicable Mathematics, London School of Economics and Political Science (2003)
2. Baum, E.B., Haussler, D.: What Size Net Gives Valid Generalization? Computer Research Laboratory. University of California, Santa Cruz (1988)
3. Beiu, V.: A survey of perceptron circuit complexity results. In: Proceedings of the International Joint Conference on Neural Networks, 2003, vol. 2, pp. 989–994. IEEE (2003)
4. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1798–1828 (2013)
5. Bengio, Y. et al.: Learning deep architectures for ai. Foundations and trends® in Machine Learning **2**(1), 1–127 (2009)
6. Bollobás, B.: Modern Graph Theory, vol. 184. Springer Science & Business Media (2013)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT press (2009)
8. Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q.V., et al.: Large scale distributed deep networks. In: Advances in Neural Information Processing Systems, pp. 1223–1231 (2012)
9. Håstad, J.: On the size of weights for threshold gates. SIAM J. Discret. Math. **7**(3), 484–492 (1994)
10. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
11. Horne, B.G., Hush, D.R.: On the node complexity of neural networks. Neural Netw. **7**(9), 1413–1426 (1994)
12. Jain, A.K., Narasimha Murty, M., Flynn, P.J.: Data clustering: A review. ACM Comput. Surv. (CSUR) **31**(3), 264–323 (1999)
13. Kurzweil, R.: The Singularity is Near: When Humans Transcend Biology. Penguin (2005)
14. Lawrence, S., Lee Giles, C., Chung Tsoi, A., Back, A.D.: Face recognition: A convolutional neural-network approach. IEEE Trans. Neural Netw. **8**(1), 98–113 (1997)
15. Le, Q.V.: Building high-level features using large scale unsupervised learning. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8595–8598. IEEE (2013)
16. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
17. Lipton, Z.C., Berkowitz, J., Elkan, C.: A critical review of recurrent neural networks for sequence learning. arXiv:1506.00019 (2015)
18. Mhaskar, H.N., Poggio, T.: Deep vs. shallow networks: An approximation theory perspective. Anal. Appl. **14**(06), 829–848 (2016)
19. Rojas, R.: Neural Networks: A Systematic Introduction, pp. 143. Springer Science & Business Media (2013)
20. Simard, P.Y., Steinkraus, D., Platt, J.C., et al.: Best practices for convolutional neural networks applied to visual document analysis. In: ICDAR, vol. 3, pp. 958–962 (2003)
21. Valiant, L.G.: A theory of the learnable. Commun. ACM **27**(11), 1134–1142 (1984)
22. Vapnik, V.: The Nature of Statistical Learning Theory. Springer science & business media (2013)