ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

On the complexity of shallow and deep neural network classifiers

Monica Bianchini and Franco Scarselli

Department of Information Engineering and Mathematics
University of Siena
Via Roma 56, I-53100, Siena, ITALY

Abstract. Recently, deep networks were proved to be more effective than shallow architectures to face complex real–world applications. However, theoretical results supporting this claim are still few and incomplete. In this paper, we propose a new topological measure to study how the depth of feedforward networks impacts on their ability of implementing high complexity functions. Upper and lower bounds on network complexity are established, based on the number of hidden units and on their activation functions, showing that deep architectures are able, with the same number of resources, to address more difficult classification problems.

1 Introduction

Recently, there has been a substantial growth of interest in feedforward neural networks with many layers [1, 2, 3]. The main idea underlying this research is that, even if shallow networks are more commonly used, it is advisable to use deep architectures to solve complex AI problems (as, e.g., in image analysis or language understanding [4]).

Nevertheless, from a theoretical point of view, the advantages of deep architectures are not yet completely understood. The existing results are limited to neural networks with boolean inputs, implementing logical gates or threshold activations, and to sum–product networks [5, 6]. Unfortunately, such results cannot be extended to common networks, with sigmoidal activation functions.

Intuitively, an important limitation is that no measure is currently available to evaluate the complexity of the functions implemented by neural networks. In fact, the claim that deep networks can solve more complex problems can be reformulated as "deep networks can implement functions with higher complexity than shallow ones, when using the same number of resources."

In this paper, we propose to exploit some concepts coming from topology theory in order to design a measure of function complexity that can be applied to neural networks, used in the classification framework. Moreover, we present some upper and lower bounds derived using such a measure. These results allow us to compare deep and shallow networks, showing that the former have some theoretical advantages over the latter. An intuitive explanation of the possible advantages of deep architectures is also provided.

The paper is organized as follows. Section 2 presents the main theoretical results, whose proofs, omitted here for space limitations, can be found in [7]. In Section 3, such results are discussed and compared with the state–of–the–art achievements. Finally, conclusions are drawn in Section 4.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence
and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

2 Theoretical results

Let $f_{\mathcal{N}} : I\!R^n \to I\!R$ be the function implemented by a feedforward neural network $\mathcal{N}$, with n inputs and a single output. We will measure the complexity of the function $f_{\mathcal{N}}$ by the topological complexity of the set $S_{\mathcal{N}} = \{x \in I\!R^n | \, f_{\mathcal{N}}(x) \geq 0\}$. Such an approach is natural when the network $\mathcal{N}$ is used for classification since, in this case, $S_{\mathcal{N}}$ is just the set of inputs scored with a non–negative response, i.e., the set of patterns belonging to the positive class.

To this aim, we will exploit the concept of Betti numbers [8]. In algebraic topology, Betti numbers are used to distinguish spaces with different topological properties. More precisely, for any subset $S \subset I\!R^n$, there exist n Betti numbers[1], denoted by $b_i(S)$, $0 \leq i \leq n-1$. Intuitively, the first Betti number $b_0(S)$ is the number of connected components of the set S, while the i–th Betti number $b_i(S)$ counts the number of $(i+1)$–dimensional holes in S. For example, let us consider the four subsets S_{BI}, S_B, S_D, S_I in $I\!R^2$, representing the character strings "BI", "B", "D" and "I", respectively. All the subsets are constituted by a single connected component except for S_{BI}, which has two components. Thus, $b_0(S_B) = b_0(S_D) = b_0(S_I) = 1$ and $b_0(S_{BI}) = 2$. Moreover, $b_1(S_{BI}) = b_1(S_B) = 2$, $b_1(S_D) = 1$ and $b_1(S_I) = 0$, because S_{BI} and S_B contain two (2–dimensional) holes, S_D contains one hole and S_I has no hole, respectively. For an example in $I\!R^3$, let us compare a sphere, S_π, and a torus, S_τ (see Fig. 1). We can observe that both of them have a single connected component and contain a 3–dimensional hole (i.e., $b_0(S_\pi) = b_2(S_\pi) = b_0(S_\tau) = b_2(S_\tau) = 1$). On the other hand, the sphere contains a single 2–dimensional hole, defined by any circle on its surface, whereas the torus has two 2–dimensional holes, the one in the center and the one in the middle of the "tube" (i.e., $b_1(S_\pi) = 1$, $b_1(S_\tau) = 2$). Thus, Betti numbers capture a topological notion of complexity that can be used to compare subspaces of $I\!R^n$. In the examples, we can assert that S_{BI} is more complex than S_B, that S_B is more complex than S_D, and that the torus is more complex than the sphere.

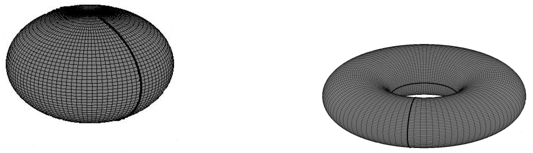

Fig. 1: A three dimensional sphere and a torus.

More precisely, the complexity of a space S is often measured by the sum $B(S)$ of the Betti numbers, i.e., by $B(S) = \sum_i b_i(S)$. Following this idea, the sum of the Betti numbers $B(S_{\mathcal{N}})$ of the region $S_{\mathcal{N}}$, which contains the patterns positively classified by a feedforward neural network $\mathcal{N}$, can be studied in order

[1] Formally, $b_i(S)$ is defined for any $i \geq 0$, but $b_i(S) = 0$ for $i > n$.

Inputs	**Hidden layers**	**Activation function**	**Bound**
Upper bounds			
n	1	threshold	$O(h^n)$
n	1	arctan	$O((n+h)^{n+2})$
n	1	polynomial, degree r	$\frac{1}{2}(2+r)(1+r)^{n-1}$
1	1	arctan	h
n	many	arctan	$2^{h(2h-1)}O((nl+n)^{n+2h})$
n	many	tanh	$2^{(h(h-1))/2}O((nl+n)^{n+h})$
n	many	polynomial, degree r	$\frac{1}{2}(2+r^l)(1+r^l)^{n-1}$
Lower bounds			
n	1	any sigmoid	$(\frac{h-1}{n})^n$
n	many	any sigmoid	2^{l-1}
n	many	polynomial, deg. $r \geq 2$	2^{l-1}

Table 1: Upper and lower bounds on the growth of $B(S_{\mathcal{N}})$, for networks with h hidden neurons, n inputs, and l hidden layers. Architecture with many layers will be called deep, architectures with one hidden layer will be called shallow.

to understand how it is affected by the architecture of $\mathcal{N}$, i.e. based on being $\mathcal{N}$ deep or not. In this paper, we provide some upper and lower bounds on $B(S_{\mathcal{N}})$ for multilayer networks, varying the network architecture and the activation function of the hidden neurons. Formally, the existence of a lower bound L for a class of networks implies that, for at least one network $\mathcal{N}$, belonging to the considered class, $B(S_{\mathcal{N}}) \geq L$ holds, while an upper bound U is such if $B(S_{\mathcal{N}}) \leq U$ for all the networks in the class.

More precisely, networks are supposed to have a single output unit with a linear activation. Different activation functions for the hidden neurons are considered: the inverse tangent, $\arctan(a) = \tan^{-1}(a)$, the hyperbolic tangent, $\tanh(a) = (e^a - e^{-a})/(e^a + e^{-a})$ and polynomial functions.

The obtained upper and lower bounds on the growth of $B(S_{\mathcal{N}})$ are reported in Table 1 (see [7], for more details and proofs). Each row of the table shows a bound for a given architecture, where l is the number of hidden layers, h the total number of hidden neurons in all the layers, and n the number of inputs. Moreover, the common big O notation is used to describe the limit behaviour of $B(S_{\mathcal{N}})$. Interestingly, a general result, collected in the following two propositions, seems to emerge from the above bounds. Intuitively, it suggests that deep networks have the capability of implementing more complex sets than shallow ones.

Proposition 1 *For network architectures with a single hidden layer, the sum of the Betti numbers, $B(S_{\mathcal{N}})$, grows at most polynomially with respect to the number of the hidden units h, i.e., $B(S_{\mathcal{N}}) \in O(h^n)$, where n is the input dimension.*

Proposition 2 *For deep networks, $B(S_{\mathcal{N}})$ can grow exponentially in the number of the hidden units, i.e. $B(S_{\mathcal{N}}) \in \Omega(2^h)$.*

Actually, the above propositions have been proved only for networks with some specific activation functions. In fact, by simplifying the bounds in Table 1, both

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

the claims are proved for polynomial activation functions: $B(S_{\mathcal{N}})$ is $O(r^n)$, for shallow networks, and it ranges between $\Omega(2^{2h})$ and $O(2^{hn})$ for deep networks. Moreover, both claims are also demonstrated for networks using $\arctan(\cdot)$: $B(S_{\mathcal{N}})$ ranges between $\Omega((h/n)^n)$ and $O((n+h)^{n+2})$ in shallow networks, and between $\Omega(2^{2h})$ and $O(4^{h^2}(nh)^{n+2h})$ in deep networks. Finally, Proposition 2 holds for generic sigmoid activation functions[2]: in this case, the lower bound is $O(2^{2h})$.

For the sake of clarity, it is worth pointing out that we do not know whether the two propositions hold for all the commonly used activation functions: the presented results are non conclusive and leave open interesting avenues of investigation for future work. For instance, Proposition 1 has not been proved for networks with the hyperbolic tangent function. Interestingly, as discussed in [7], the extension of such results actually may require significant advancements in algebraic topology, related to some important and still unsolved problems.

3 Related literature and discussion

The multilayer networks' ability of approximating any continuous map, up to any degree of precision, was established during the early 90's. After that, this property was extended in several ways [9], considering networks with very generic neurons (e.g., with analytic activation functions), expanding the class of approximable maps (e.g., to integrable functions), considering non–static networks (e.g., recurrent [10], recursive [11] and graph neural networks [12]), and providing upper bounds on the rate of approximation w.r.t. the number of neurons (e.g., see [13]). Unfortunately, such results do not allow to distinguish between deep and shallow architectures, since they are based on properties which are inherited from simple to more complicated architectures, instead of evidencing distinguishing features.

On the other hand, researchers in the field of logic networks have actually pursued the goal of defining the effect of the network depth on the amount of resources required to implement a given function. In this ambit, it has been shown that there exist boolean functions, whose realization requires a polynomial number of logic gates (AND, OR, NOT) using a network with l layers, whereas an exponential number of gates is needed for a smaller number of layers [14]. A well–known function of this class is the parity function, which requires an exponential number of gates (in the number of inputs), if the network has two layers, whereas it can be implemented by a linear number of logic gates, if a logarithmic number of layers (in the input dimension) are employed.

In [5], deep and shallow sum–product networks[3] were compared, using two classes of functions. Their implementation by deep sum–product networks requires a linear number of neurons with respect to the input dimension n and the network depth l, whereas a two–layer network needs at least $O(2^{\sqrt{n}})$ and $O((n-1)^l)$ neurons, respectively. Finally, similar results were obtained for weighed threshold circuits[4]. For example, it has been proved that there are

[2] A sigmoid is a monotone increasing function having left and right limits.

[3] A sum–product network consists of neurons that either compute the product or a weighted sum of their inputs.

[4] Thresholds circuits are multilayer networks with threshold activation functions, i.e., $\sigma(a) = 1$, if $a \geq 0$, and $\sigma(a) = 0$, if $a < 0$.

monotone functions f_k that can be computed with depth k and a linear number of logic (AND, OR) gates, but they require an exponential number of gates to be computed by a weighed threshold circuit with depth $k-1$ [15].

The above mentioned results, however, cannot be applied directly to common BackPropagation networks. In this sense, as far as we know, the results reported here are the first attempt to compare deep and shallow architectures, that exploit the common $\tanh(\cdot)$ and $\arctan(\cdot)$ activation functions.

Interestingly, the presented results are also related to Vapnik–Chervonenkis dimension (VC–dim). In fact, the VC–dim has been proved to be $O(p^2)$, for networks with $\arctan(\cdot)$ and $\tanh(\cdot)$ activation, where p is the number of parameters [16]. Thus, the VC–dim is polynomial with respect to the number of parameters and, as a consequence, it is polynomial also in the number of hidden units. These bounds do not depend on the number of layers in the network, suggesting that, in practice, the depth of a network has a larger impact on the complexity of the implemented function than on its generalization capability. In other words, using the same amount of resources, deep networks are able to face more complex applications without loosing in generalization.

Finally, let us provide an intuitive explanation of the possible advantages of deep architectures. First, notice that the k–th hidden layer of a feedforward network realizes a function that correlates the outputs of the neurons in layer $k-1$ with the inputs of the neurons in layer $k+1$. Such a correlation can be represented as a map, so that the global function implemented by a deep network results in a composition of several maps, whose number depends on the number of its hidden layers. On the other hand, the function composition mechanism intuitively allows "to replicate the same behaviour on different regions of the input space." Without providing a formal definition of such a statement, let us illustrate this concept with an example. Consider the composition $f = g \circ t$, with $g : D \rightarrow I\!R$, $t : D \rightarrow D$, defined on the domain $D \subseteq I\!R^n$. Moreover, let us assume that there exist m sets, $A_1, \ldots, A_m \subseteq D$, such that $t(A_i) = D$, $1 \leq i \leq m$. We can observe that the number of connected regions $b_0(S_f)$ of the set $S_f = \{x | \, f(x) \geq 0\}$ is at least m times $b_0(S_g)$, where $S_g = \{x | \, g(x) \geq 0\}$. In fact, f behaves on each A_i as g behaves on the whole domain D. Moreover, this argument can be extended to the case when g is composed with t several times, i.e., $f = g \circ t_k$, where $t_k = t \circ t \circ \ldots \circ t$ for k occurrences of t. In this case, the ratio $b_0(S_f)/b_0(S_g)$ is at least m^k.

Therefore, we can intuitively conclude that deep networks are able to realize, with few resources, functions that replicate a certain behaviour in different regions of the input space. Obviously, here the concept of "replicating a behaviour" must be interpreted in a very broad sense, since the hidden layers of a deep network can approximate any function.

4 Conclusions

In this paper, we have proposed a new measure, based on Betti numbers, to evaluate the complexity of functions implemented by neural networks. The measure has been used for comparing deep and shallow feedforward neural networks, with arctangent and polynomial activation functions. It has been shown that, with the same number of hidden units, deep architectures can realize maps with a higher complexity with respect to shallow ones. An informal discussion on the practical differences between deep and shallow networks is also included.

Interestingly, the proposed measure provides a tool that allows us to study connectionist models from a new perspective. It is a future matter of research the application of that measure to more complex architectures, such as convolutional neural networks, recurrent neural networks, and neural networks for graphs. Moreover, it is interesting to design algorithms that can estimate the actual complexity of the function implemented by a given network, f.i., by computing the Betti numbers of a dataset classified by the network (see e.g., [17]).

References

[1] Y. Bengio, Y. LeCun, R. Salakhutdinov, and H. Larochelle, editors. *Proceedings of the Deep Learning Workshop: Foundations and Future Directions (NIPS 2007)*. 2007.

[2] H. Lee, M. Ranzato, Y. Bengio, G.E. Hinton, Y. LeCun, and A.Y. Ng, editors. *Proceedings of the Deep Learning and Unsupervised Feature Learning Workshop (NIPS 2010)*. 2010.

[3] K. Yu, R. Salakhutdinov, Y. LeCun, G.E. Hinton, and Y. Bengio, editors. *Proceedings of the Workshop on Learning Feature Hierarchies (ICML 2009)*. 2009.

[4] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[5] Y. Bengio and O. Delalleau. Shallow vs. deep sum–products networks. In *Advances in Neural Information Processing Systems*, volume 24, pages 666–674, 2011.

[6] O. Delalleau and Y. Bengio. On the expressive power of deep architectures. In *Proceedings of the 22nd International Conference on Algorithmic Learning Theory*, pages 18–36, 2011.

[7] M. Bianchini and F. Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks*, 2014. To be published.

[8] G.E. Bredon. *Topology and Geometry, Graduate Texts in Mathematics*. Springer, 1993.

[9] F. Scarselli and A.C. Tsoi. Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results. *Neural Networks*, 11:15–37, 1998.

[10] B. Hammer. On the approximation capability of recurrent neural networks. *Neurocomputing*, 31(1–4):107–123, 2000.

[11] M. Bianchini, M. Maggini, L. Sarti, and F. Scarselli. Recursive neural networks for processing graphs with labelled edges: Theory and applications. *Neural Networks*, 18(8):1040–1050, 2005.

[12] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, and G. Monfardini. Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks*, 20(1):81–102, 2009.

[13] V. Kurkova, P. Savicky, and K. Hlavackova. Representations and rates of approximation of real–valued boolean functions by neural networks. *Neural Networks*, 11(4):651–659, 1998.

[14] J. Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 6–20. ACM, 1986.

[15] J. Håstad and M. Goldmann. On the power of small–depth threshold circuits. *Computational Complexity*, 1(2):113–129, 1991.

[16] P.L. Bartlett and W. Maass. Vapnik–Chervonenkis dimension of neural nets. In M.A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 1188–1192. Cambridge, MA: MIT Press, 2003. Second Edition.

[17] M. Maillot, M. Aupetit, and G. Govaert. A generative model that learns Betti numbers from a data set. In *ESANN2012, 15th European Symposium on Artificial Neural Networks*, pages 537–542, Bruges, Belgium, 2012.