[1,2]

Complexity of Predictive Neural Networks

Mark A. Kon*
Boston University

Leszek Plaskota†
Warsaw University

October 14, 2003

Abstract

We announce some general bounds on the interactions of neural and information complexities of feedforward neural networks using general classes of activation functions. We show that, up to constant factors, neural and information complexities combine in a well-defined way in the determination of the complexity of a network.

1. Introduction

Neural networks have as one of their main functions the prediction of outputs from inputs in large, multi-component, complex systems, through extrapolation from example inputs and outputs of such systems. They are currently used for such tasks as prediction of delinquency rates of credit card holders based on past credit parameters, prediction of outputs of chemical admixtures as functions of their physical components and parameters (e.g., temperature), robotic motion (prediction of outputs of motion parameters from inputs of desired final positions), and stock market forecasting. The effective task of such a predictive neural network is to forecast values of an input-output (i-o) function f at unknown input values from known values of f. The inputs are current information, and the output is a prediction of what will occur at a future time.

The complexity of building such a network is a crucial issue, since it has been shown that neural networks are complete, i.e., if given enough information and hardware, they can predict any i-o function f with arbitrary accuracy (see, e.g., [11], [23]). The complexity of building a network consists largely of two parts, that of learning the function f to be predicted (information complexity), and that of constructing the network which will perform the approximation to f (neural complexity); see below.

The purpose of this paper is to present new results on learning and performance complexities of neural systems for implementing input-output (i-o) relations. We consider general versions of three-layer feedforward neural networks of radial basis function (RBF) type, with activation functions which may be different for each neuron. The results are new for all classes of activation functions, including RBF functions typical of artificial network architectures and ridge functions common in biological models.

We now briefly present the context of these results. The mathematical study of feedforward neural networks starts with completeness questions, which ask whether any function in a class F can be computed by some network in a given category $\mathcal{N}$. Here $\mathcal{N}$ might be a class of networks whose middle neurons y_i compute radial basis functions (RBF's) (fig. 2 below) or ridge functions (described below) of the input vectors $x \in \mathbb{R}^n$. Completeness issues were studied from around 1985 to 1990 with great interest and a number of comprehensive positive results (see, e.g., [7], [11], [23]).

Neural complexity theory addresses the next natural question once completeness is established, What is the size (complexity) of neural networks required to ϵ-approximate functions f in F? Equivalently,

*Research partially supported by the National Science Foundation and the U.S. Fulbright Commission

†Research partially supported by the Polish-American Fulbright Foundation and the National Committee for Scientific Research of Poland

how difficult is it to build a network which performs a given task? Active study of complexity for networks began in the early nineties after completeness issues had been settled through around 1990. A number of excellent foundational works, including those of Poggio and Girosi ([12], [13], [25]-[27]), Micchelli and Mhaskar ([16]-[21]), Chui and Li ([4]-[6]), and Barron ([1],[2]) have studied the performance or neural complexity (numbers of neurons needed) of these networks, if basic elements compute functions of reasonable "smoothness" (small Sobolev norm). In classical computer science completeness and complexity theory take on the forms of *theory of computation* and *computational complexity theory*, respectively.

Complexity theory of neural networks can be separated into learning complexity (how much work needs to be done to learn f) and performance or neural complexity (how many neurons will be needed to implement a good approximation $q(x)$ to $f(x)$. Neural performance questions, as indicated above, relate to the existence of (sufficiently simple) networks capable of computing a class F of desired i-o functions. Neural learning theory deals with whether (and to what accuracy) networks can learn the i-o function f which they are to predict from series of examples $Nf = (f(z_1), f(z_2), \ldots, f(z_n))$, with $\{x = z_i\}_i$ an array of example input vectors. (Most real-world i-o functions are known only through examples). The complexity of learning has up to now dealt largely with questions of complexity of algorithms such as backpropagation and the Boltzmann machine.

A complete complexity theory of neural networks needs to address both learning and performance as network-related complexity issues. Learning complexity is measured here as *information complexity*, which is the number of examples of f required to approximate f within a given tolerance ϵ. Performance complexity is measured as *neural complexity*, which is the number of neurons necessary to approximate f within ϵ.

We show here that in general contexts, the relationship between information and neural complexity can be simply parsed, so the two questions can be studied separately before their interaction is analyzed. The two parts of the complexity question for neural networks pose difficult mathematical and phenomenological problems, and the fact that they can be largely separated can be very helpful.

We thus initially separate complexity theory for networks into two scenarios. The first occurs when we know exactly the i-o function f which is to be approximated, i.e., have full information. Here the question is, how complex a network do we need to express f to tolerance ϵ? The second occurs when it is assumed we have unlimited computational resources (as many neurons as we need), and ask how many examples of f we need for its ϵ-approximation.

Information complexity [14] can be considered the "second half" of neural complexity theory dealing with information issues, the basic element of learning. While study of neural complexity has had a good start (see above references), information complexity is still in its initial stages. The interaction of the two is our concern here.

2. Preliminaries and definitions

We restrict ourselves to networks of the form in Fig. 1 below. Each node is a neuron in this artificial network, with each vertical column a layer. The activation level of neuron j in layer i is x_{ij} (we will also let x_{ij} denote the neuron itself). We define $x_i = (x_{i1}, \ldots, x_{in})$ to be the vector of activations in layer i, and assume feedforward vector functions $f_i : \mathbb{R}^n \to \mathbb{R}^n$ map the activation vector x_{i-1} into x_i, i.e., $f_i(x_{i-1}) = x_i$. We initially assume general forms for the f_i, and define $f_{ij}(x_i) = x_{(i+1)j}$ to be the j^{th} output component of the vector function f_i.

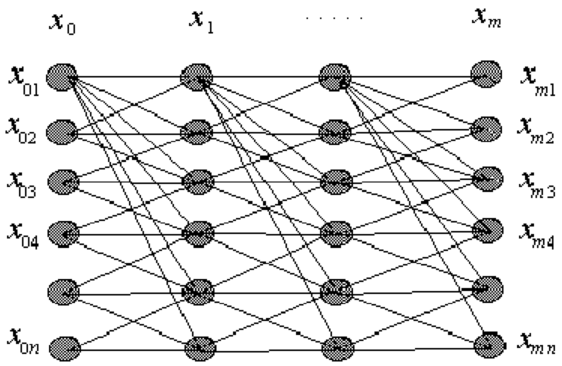

Fig. 1: A multilayer neural network with $m = 4$ layers; feedforward connections completely connect each layer to the next. Input is x_0 and output is x_m.

A major goal of studying artificial feedforward networks is to find what input-output functions $f(x)$ can be approximated by compositions $\tilde{f}(x) = f_m \circ \ldots \circ f_0(x)$, given the f_i are drawn from various parametric families. These include so-called *ridge functions* $f_{ij}(x) = r(x \cdot a - \theta)$ with r: $\mathbb{R} \to \mathbb{R}$ a monotone sigmoid function which increases from 0 to 1. They also include *radial basis functions* (RBF's) $f_{ij}(x) = G(x - z_i)$, generally radial functions localized at points z_i. Since all computational and other tasks can be formalized as input-output tasks, theoretically any problem in intelligent action can be formulated as that of approximating a desired input-output function $f(x)$ by $\tilde{f}(x)$, with input x and output $f(x)$ appropriately coded in $\mathbb{R}^n$.

Because of theorems proving universal approximation properties of even simple networks, the stakes have been raised to the extent that many approximation questions (both in terms of approximability and complexity of approximation) are now formulated for simple three layer networks of the form in Fig 2 below. Many types of desired outputs can be encoded in a single number, and multi-output networks can be reduced to concatenations of single-output networks. Thus there is an assumption of only one output neuron q, computing strictly additive output

$$q = \sum_i w_i y_i = \sum_i w_i K(x - z_i), \tag{1}$$

with $y_i = K(x - z_i)$ a fixed localized function centered at z_i. The units y_i are denoted as hidden units.

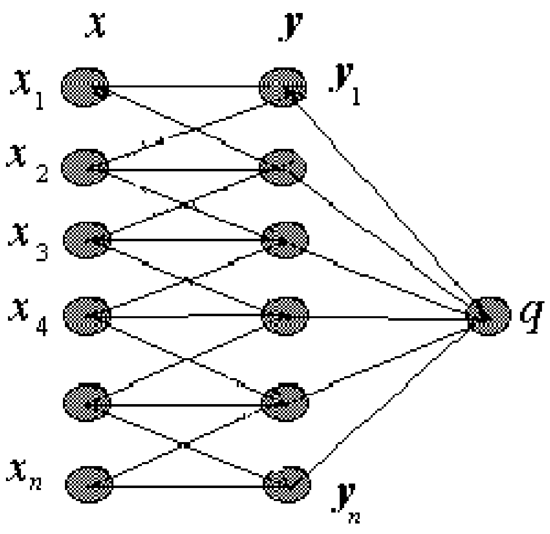

Fig. 2: A three layer model network; there is complete connectivity from each layer to the next.

There are important reasons why such networks are learning complexity-optimal in certain types of i-o function approximations; this is explained to an extent in [14],[15].

Alternatively, three layer networks are also studied in the more biologically oriented form

$$q = \sum_i w_i y_i = \sum_i w_i r(x \cdot a - \theta) \tag{2}$$

with r the above ridge functions. Our results are appropriate to a general three layer network

$$q = \sum_i w_i y_i = \sum_i w_i d_i(x) \tag{3}$$

with $\mathcal{D} = \{d_i(x)\}_i$ a dictionary of activation functions. Such networks can implement nonlinear approximations of desired i-o functions f; in approximation theory and matching pursuit such approximations have been of recent interest ([3],[8]-[10]).

3. Neural complexity and information complexity

It should be noted that the approximation (3) epitomizes the parallels between feedforward neural network theory and approximation theory which began to be noticed in the late eighties. Indeed, it shows effectively that feedforward neural nets are analysis-equivalent to approximation theoretic paradigms. The two fields have taken parallel paths even when they have not interacted, and the issues here involve a neural network formulation of some of the more interesting problems currently in complexity-theoretic approximation theory, those involving matching pursuit and nonlinear approximation.

The artificial neural network paradigm, freed from requirements that activation functions $f_{ij}(x)$ must necessarily have biologically motivated forms, now stands to extend the standard RBF architecture, which may be necessary in a number of contexts. Specifically we assume that the network in Fig. 2 is endowed so each hidden neuron y_i can compute a function $d_i(x)$ of whatever form necessary to "get the job done". In this case neural and approximation theoretic complexity combine in the following question:

Question 1: Given a dictionary $\mathcal{D} = \{d_i(x)\}_{i \in I}$ of functions $d_i : \mathbb{R}^n \to \mathbb{R}^n$, what is the neural complexity (number of hidden neurons y_i) of the smallest neural network which can approximate a given i-o function $f(x)$ within error $\epsilon > 0$ in a given error measure?

This question has been studied in approximation theory ([9],[10]) and wavelet and statistical theory (matching pursuit; see [3],[8]). Results from these areas translate into interesting ones for neural networks with general computable activation functions $d_i(x)$. The notion of information complexity used here (involving the number of function evaluations $d_i(x)$ needed to approximate the i-o function f) carries over from computer science and continuous computational complexity theory ([28],[29]), and does not involve counting the Turing bit operations of more classical computational complexity. See [9],[10] for mathematical discussion of this question in the context of nonlinear approximation.

The second half of this question is

Question 2: What is the information complexity of function approximation in the above context? That is, given partial information about the i-o function f, what is the smallest number of examples $Nf = (f(x_1), \ldots, f(x_n))$ from which we can approximate f within ϵ (for the moment assuming unlimited neural resources)?

Question 2 is central to neural network theory, since examples are generally the only source of information about f.

The above questions have been dealt with reasonably successfully in nonlinear approximation theory and continuous computational complexity theory, respectively. The interaction of the two complexities determines how hard it is to build a network which implements an i-o function:

Question 3: How do information complexity and neural complexity interact in determining the difficulty of building a neural network implementing a given i-o function $f(x)$ to a given tolerance ϵ?

This is the question dealt with here. It is shown next that question 3 can be reduced to questions 1 and 2 through a decoupling of information and neural complexities.

4. Interaction of the two complexities

Neural and information complexities interact simply, as the following theorems (whose proofs are omitted for brevity) demonstrate. To more easily quantify the interaction, we work with the error of approximation $\epsilon(k)$, which is the inverse function of the ϵ-information complexity $k(\epsilon)$.

We first consider neural complexity. For a network with n neurons, a network using the dictionary $\mathcal{D} = \{d_i(x)\}_{i \in I}$ computes functions of the form

$$q(x) = \sum_{i \in J} w_i d_i(x)$$

for subsets $J \subseteq I$ of cardinality n (here $x \in \mathbb{R}^k$). Let $\mathcal{N}_n$ be the set of functions obtainable in this form, so $\mathcal{N}_n$ is the set of neural networks of neural complexity n (here n measures the number of hidden units).

We formally define neural and information complexities of approximating a given i-o function f by $q(x)$ as follows. Let F be a class of functions which we know f belongs to a priori. For example if f is assumed smooth, $F = \{f(x) : \|f\|_{H^s} \leq a\}$ might be a class of functions of restricted Sobolev (smoothness) norm $\|\cdot\|_{H^s}$ ([28],[29]). Alternatively F might be one of an increasing sequence of spaces $\{F_i\}_i$, with F chosen because it contains a sufficiently rich class of functions which nevertheless is not too large (i.e., does not overfit sample values Nf; see [30],[31]). Such classes are measured by their so-called VC-dimension.

We assume our error is measured by a norm $\|\cdot\|$ which defines a normed linear space F_1 containing F. We assume the information we have about f consists of examples $Nf = (f(x_1), \ldots, f(x_k))$. More generally we can allow other types of linear information about f, of the form

$$Nf = (L_1 f, \ldots, L_k f),$$

with L_i bounded linear functionals from a given class $\mathcal{L}$ (which may consist of all functionals or a subset of them). In the present context we are interested in information from examples, and so we would normally restrict $\mathcal{L}$ so that $L_i f = f(x_i)$ always consist of function evaluations (note that examples of f are normally evaluations of f at specific values of the input data x_i; see [28],[29]).

For given $f \in F$ and information Nf, let $e(f, \phi, N)$ be error of guessing f only from the information $Nf = (L_1 f, \ldots, L_k f)$, using a given reconstruction algorithm $\phi(Nf)$ to best approximate f from Nf, so

$$e(f, \phi, N) = \|f - \phi(Nf)\|.$$

If we consider the most difficult $f \in F$ to approximate, we define

$$e(\phi, N) = \sup_{f \in F} e(f, \phi, N) \tag{4}$$

as the worst case error of using the fixed algorithm ϕ with fixed information N. We then choose the best N and ϕ by defining

$$e(n, k) = \inf_{N : F \to \mathbb{R}^k;\, \phi : \mathbb{R}^k \to \mathcal{N}_n} e(\phi, N)$$

to be the minimal error in approximating functions $f \in F$ that can be made by using information $Nf = (f(x_1), \ldots, f(x_k))$ of cardinality k, and a neural network with n neurons (in class $\mathcal{N}_n$). This is the smallest error using k pieces of information and n neurons.

We can now define the neural and information complexities as

$$e_{\text{neur}}(n) = \inf_k e(n,k)$$

which is the minimal error assuming n neurons and unlimited information, and

$$e_{\text{info}}(k) = \inf_n e(n,k),$$

which is the minimal error assuming k pieces of information $f(x_1), \ldots, f(x_n)$ and unlimited neural resources.

The neural and information complexities, as usual, are the inverses of the above functions:

$$n(\epsilon) = \inf\{n : e_{\text{neur}}(n) \leq \epsilon\}$$
$$k(\epsilon) = \inf\{k : e_{\text{info}}(k) \leq \epsilon\}.$$

They represent the minimal neural and informational complexities of approximating a function $f \in F$ with error $\epsilon > 0$.

We are interested in the relationship of these two complexities, and in particular their interaction. Since inverting functions is not hard, we will study the relationships of the errors, $e_{\text{neur}}(n)$, $e_{\text{info}}(k)$, with joint error $e(n,k)$.

We then have:

Theorem 1. *The neural and information complexities of function approximation by a neural network interact in a simple way to determine the joint complexity of approximation. Specifically,*

$$\max(e_{\text{info}}(k), e_{\text{neur}}(n)) \leq e(n,k) \leq 2e_{\text{info}}(k) + e_{\text{neur}}(n)$$

As a consequence, it is possible to characterize the joint information and neural complexity of a general network for performance of a given task in terms of the two complexities individually:

Corollary 1. *In order to construct a network in the class $\mathcal{N} = \cup_n \mathcal{N}_n$ with error at most ϵ, it is necessary to use at least $n(\epsilon)$ neurons and $k(\epsilon)$ examples, and it is sufficient to use $n(\alpha\epsilon)$ neurons and $k(\beta\epsilon)$ examples, where α and β are any numbers satisfying $\alpha + 2\beta \leq 1$.*

If we choose $\alpha = \beta = 1/3$ we have

Corollary 2. *In order to approximate any function f from the class F with error $\epsilon > 0$, it is necessary to use at least $n(\epsilon)$ neurons and $k(\epsilon)$ examples, and sufficient to use at most $n(\epsilon/3)$ neurons and $k(\epsilon/3)$ examples.*

To this extent (up to constant factors in arguments of complexity functions) the study of the joint complexity effectively decouples into the separate study of information complexity and of neural complexity.

5. Example

We will briefly mention an example of an application of such results. There is currently great interest in using parametric neural methods to predict consumer purchasing patterns. Indeed, the amount of information on consumers today via Internet interaction protocols is daunting (the appropriateness of such information collection is not discussed here). Suppose we wish to predict for a given consumer the expectation $E(X)$ of the general random variable X representing the purchases he or she will make from our web site over the next year. We may assume that this depends parametrically on a number of known quantities, including values $p_1, \ldots, p_n$ of purchases for this consumer over the last year at other web sites $s_1, \ldots, s_n$. In order to model the i-o function $f(x) = E(X \mid p_1, \ldots, p_n)$ (with $x = (p_1, \ldots, p_n)$), we might wish to know the number of examples of this function and the size of the network required to estimate it within error ϵ, measured, say, in mean square norm. We should mention that since the function $f(x)$

is not measured deterministically, use of examples $Nf = (f(x_1), \ldots, f(x_n))$ (here $x_i = (p_{i1}, \ldots, p_{ij})$) in determining f, using a library $\mathcal{D}$ of RBF's ([25]-[27]), must be done in a model including error terms, i.e., one in which observation $f^*(x_i)$ is related to its expected value $f(x_i)$ by

$$f^*(x_i) = E(X \,|\, p_{i1}, \ldots, p_{in}) + \delta_i,$$

with δ_i a random error term. Such modifications to standard RBF techniques are not difficult in RBF models including error terms [24], if we assume Gaussian independent errors δ_i. Once the function (and its smoothness parameters) are modeled (determining the RBF class we will use), it is possible to find lower and upper bounds for the information complexity function $k(\epsilon)$ using methods of continuous complexity theory (see [28],[29],[14]). Similarly, it is possible to find the neural complexity function $n(\epsilon)$ for the presumed class F of f (see [9],[10],[16]-[21]). For brevity we avoid the details of how this can be done in such a parametric model, but remark that there are reasonable ways of choosing the class F, say, as a ball in a Sobolev space in cases where a given degree of smoothness is reasonable for the class.

Using the information and neural complexities obtained above, we can use Corollary 2 to determine upper and lower bounds on numbers of neurons and examples which will be necessary to find an ϵ-estimate of the desired function f. We remark that there exist constructive algorithms for using information of cardinality k and a network of size n to approximate f (see [25]-[27] for the RBF cases).

References

[1] Barron, A.R. Universal approximation bounds for superposition of a sigmoidal function, *IEEE Trans. Information Theo.* **39** (1993), 930-945.

[2] Barron, A.R. Approximation and estimation bounds for artificial neural networks, *Machine Learning* **14** (1984), 115-133.

[3] Bergeaud, F. and Stephane Mallat, Matching pursuit: Adaptative representations of images and sounds, *Computational and Applied Mathematics* **15** (1996), October 1996.

[4] Chui, Charles and Xin Li, Approximation by ridge functions and neural networks with one hidden layer, *J. Approximation Theory* **70** (1992), 131-141.

[5] Chui, Charles, Xin Li, and Hrushikesh Mhaskar, Neural networks for localized approximation, Center for Approximation Theory Report 289, 1993.

[6] Chui, Charles, Xin Li, and Hrushikesh Mhaskar, Limitations of the approximation capabilities of neural networks with one hidden layer, *Advances in Computational Mathematics* **5** (1996), 233-243.

[7] Cybenko, George, Approximation by superposition of sigmoidal functions, Math. Control, *Signals, and Systems* **2** (1989), 303-314.

[8] Davis, G., Stephane Mallat, and M. Avelaneda, Adaptive greedy approximations, *J. Constructive Approximation* **13** (1997), 57-98.

[9] DeVore, R., and V. Temlyakov, Nonlinear approximation by trigonometric sums, *J. Fourier Anal. Appl.* **2** (1995), 29-48.

[10] DeVore, R. and V. Temlyakov, Nonlinear approximation in finite dimensional spaces, *J. Complexity* **13** (1997), 489-508.

[11] Funahashi, K., On the approximate realization of continuous mappings by neural networks, *Neural Networks* **2** (1989), 183-192.

[12] Girosi, Federico, Regularization theory, radial basis functions and networks, in *From Statistics to Neural Networks, Theory and Pattern Recognition Applications*, V. Cherkassky, J.H. Friedman, and H. Wechsler, eds., Springer-Verlag, 1994.

[13] Girosi, Federico, M. Jones, and Tomaso Poggio, Regularization theory and neural network architectures, *Neural Computation* **7** (1995), 219-269.

[14] Kon, Mark and Leszek Plaskota, Information complexity of neural networks, to appear, *Neural Networks*

[15] Kon, Mark and Leszek Plaskota, Neural networks, radial basis functions, and complexity, in *Statistical Physics Proceedings*, Bialowieza, 1997, 322-335.

[16] Mhaskar, Hrushikesh, Approximation of smooth functions by neural networks; in Dealing with complexity: A neural network approach, K. Warwick et. al., eds, *Perspectives in Neural Computing*, Springer Verlag, London, 1998, 189-204.

[17] Mhaskar, Hrushikesh, Neural networks for optimal approximation of smooth and analytic functions, *Neural Computation* **8** (1996), 164-177.

[18] Mhaskar, Hrushikesh, Neural Networks and Approximation Theory, *Neural Networks* **9** (1996), 721-722.

[19] Mhaskar, Hrushikesh N. and Charles A. Micchelli, Approximation by superposition of sigmoidal and radial basis functions, *Advances in Applied Mathematics* **13** (1992), 350-373.

[20] Mhaskar, Hrushikesh N. and Charles A. Micchelli, Dimension independent bounds on the degree of approximation by neural networks, *IBM J. Research and Development* **38** (1994), 277-284.

[21] Mhaskar, Hrushikesh and Charles Micchelli, Degree of approximation by neural and translation networks with a single hidden layer, *Advances in Applied Mathematics* **161** (1995), 151-183.

[22] Packel, Edward and Henryk Woźniakowski, Recent developments in information-based complexity, *Bull. Amer. Math. Soc.* **17** (1987), 9-36.

[23] Park, J. and I. Sandberg, Approximation and radial-basis-function networks, *Neural Computation* **5** (1993), 305-316.

[24] Plaskota, Leszek, *Noisy Information and Complexity*, Cambridge University Press, 1996.

[25] Poggio, Tomaso and Federico Girosi, Regularization algorithms for learning that are equivalent to multilayer networks, *Science* **247** (1990), 978-982.

[26] Poggio, Tomaso and Federico Girosi, A theory of networks for approximation and learning, A.I. Memo No. 1140, M.I.T. A.I. Lab, 1989.

[27] Poggio, Tomaso and Federico Girosi, A sparse representation for function approximation, *Neural Computation* **10** (1998), No. 6.

[28] Traub, Joseph, Grzegorz Wasilkowski, and Henryk Woźniakowski, *Information-Based Complexity*, Academic Press, Boston, 1988.

[29] Traub, Joseph and Henryk Woźniakowski, *A General Theory of Optimal Algorithms*, Academic Press, New York, 1980.

[30] Vapnik, V.N., *The Nature of Statistical Learning Theory*, Springer, New York, 1995.

[31] Vapnik, V.N. and A. Ya. Chervonenkis, The necessary and sufficient conditions for consistency in the empirical risk minimization method, *Pattern Recognition and Image Analysis* **1**(3) (1991), 283-305.