Neural Networks, Radial Basis Functions, and Complexity

Mark A. Kon[1]
Boston University and University of Warsaw

Leszek Plaskota
University of Warsaw

1. Introduction

This paper is an introduction for the non-expert to the theory of artificial neural networks as embodied in current versions of feedforward neural networks. There is a lot of neural network theory which is not mentioned here, including the large body of work on natural neural nets (i.e., the theory of neural networks in animals). For some excellent foundational work on this topic, see [G1, G2]. We also include very recent work [KP] regarding what we consider to be the next important step in the theory of neural networks and their complexity, namely informational complexity theory for neural nets.

In some sense this paper, being presented at a conference on condensed matter physics, follows a tradition which David Hestenes began at a conference on the maximum entropy method in honor of Edward Jaynes, the father of this technique, which has been so successful in the study and applications of condensed matter physics. In his contribution to the proceedings of this conference [He], which we highly recommend, Hestenes presented a powerful argument for the viewpoint of Grossberg (see references above) in his explanations of numerous properties of natural neural networks in a set of equations known as the *Grossberg equations,* which form a dynamical system with interesting properties in its own right, studied extensively by Hirsch and others. This however is as much as we will say about work in the area of natural neural nets. For a general presentation of the theory of artificial neural networks, including methods and approaches beyond the scope of this paper, see [Ha].

The current emphasis on neural networks as a venue toward the eventual ideal of artificial intelligent systems had its newest genesis in the mid-1980's, when it was realized that the classical von Neumann architecture (i.e., standard model of computation) was not coming as close as some had hoped to true artificial intelligence, at the pace hoped for. The vision of the 1960's, when the current field of artificial intelligence became popular, had been that certainly by the year 2000, von Neumann architectures would be capable of coming close to simulating tasks which are called "intelligent." At this point, around 1985-87, workers in many fields, including computer science, mathematics, physics, biology, and psychology, began an interdisciplinary effort which continues to the present. A large conference in San Diego with some 1500 participants in the "summer of networks" of 1987 topped off a very rapid increase in interest in this area, which had previously been marginalized due to a variety of circumstances dating to the 1960's and earlier.

[1]Research partially supported by the National Science Foundation, Air Force Office of Scientific Research, and U.S. Fulbright Commission

The impetus for this "switch" of the hopes of many workers to neural nets came largely from the "existence proof" which neural nets provide. Specifically, working neural networks performing intelligent tasks certainly exist in the world today as natural neural systems.

Applications of neural nets have been studied in physics, biology, psychology, engineering, and mathematics, and working artificial networks have been built to make decisions on mortgage loan applications, identify properties of satellite photographs, balance a broom on its end, and a large number of other varied tasks.

In this paper, we first introduce notions related to feedforward networks, including those related to their most modern incarnations, so-called radial basis function (RBF) networks. We include a more detailed mathematical analysis of the approximation properties of such RBF networks. Complexity issues for networks are covered in detail in the last several sections, including very recent results of the authors in these areas. The reader will hopefully be convinced of the utility of the work in complexity (as well as see examples of how neural networks would work in a practical setting) in several examples given at the end.

2. Feedforward networks

A basic component of many neural nets, both natural and artificial, is the *feedforward network*. A basic such network has the structure depicted in the following diagram:

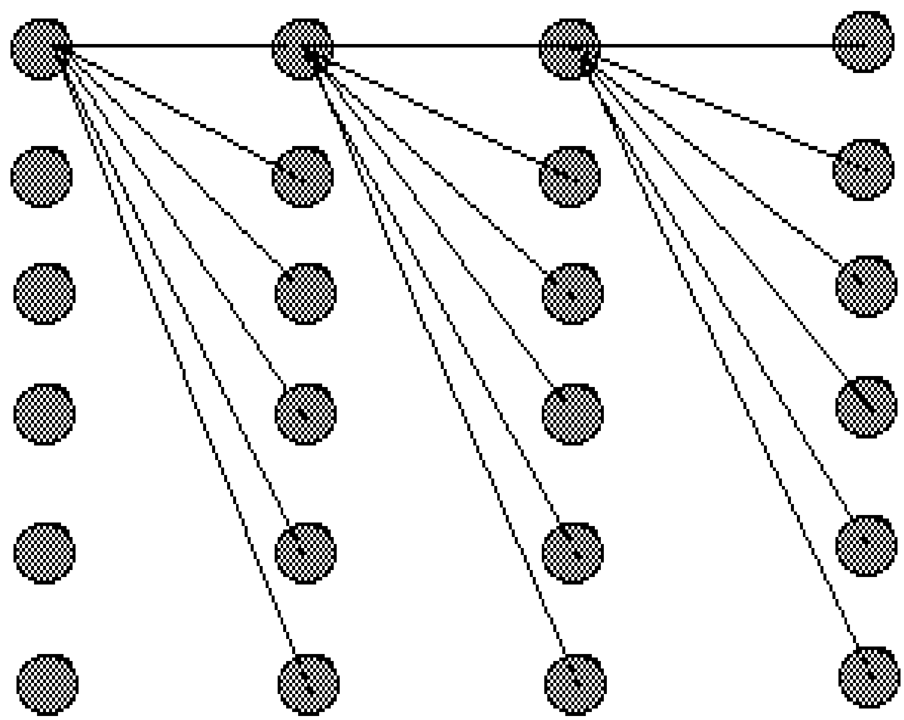


fig. 1

Here a layer is the usual term for a vertical row of neurons. There is full connectedness between the n^{th} and $n+1^{th}$ layer, i.e., every neuron in the n^{th} layer has a connection feeding *forward* into every neuron in the $n+1^{th}$ layer. These are not all shown in fig. for reasons of clarity. Thus neurons in each layer influence neurons in the successive layer. The first layer contains the "input", i.e., we control activations of its neurons. For example, the first layer might represent the "retina" of a visual system, which obtains information which will be fed forward into and processed in further layers. The last layer contains "output", i.e., its activations provide a desired output that the neural network provides in response to input in first layer.

Funahashi [Fu] and Hecht-Nielsen, among others, have shown that if we desire a network which is able to take an arbitrary input pattern in the first layer, and provide an arbitrary desired output pattern in the last layer, all that is necessary is 3 layers:

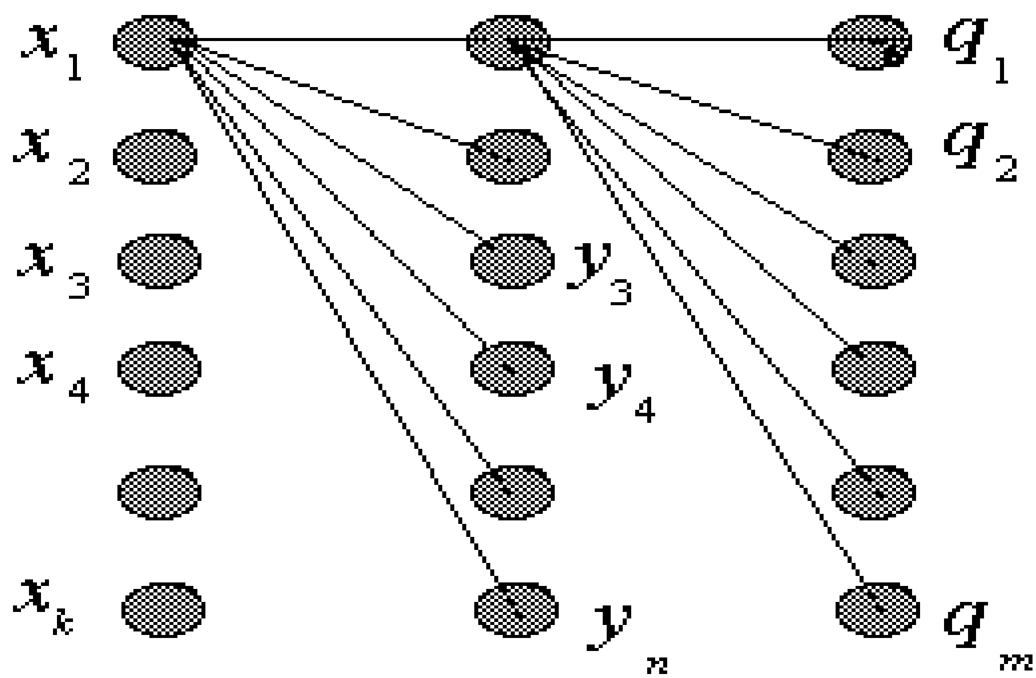


fig. 2

More specifically, general i-o functions can be approximated in a general class of error norms using networks of this type.

Henceforth we consider only 3 layer networks. We define x_i to be the activation level (either chemical or electrical potential) of the i^{th} neuron in first layer, y_i to be the activation level of the corresponding neuron in second layer, and q_i to be the corresponding activation level in the third layer. In addition, we define v_{ij} to be the strength of the connection (weight) from the j^{th} neuron in layer 1 to i^{th} neuron in layer 2, w_{ij} to be the weight from the j^{th} neuron in layer 2 to i^{th} in layer 3.

As an example, the first layer might be the retina and x_i might be proportional to the illumination level at the neuron labeled x_i. This is the input layer - in this case light shines on retina and activates it. The last layer might represent what we would call a speech center (neurons ultimately connected to a vocal device), and its pattern q_i of neuron activations corresponds to verbal description about to be delivered of what is seen in first layer.

3. Neuron interaction rule

Neurons in one layer will be assumed to influence those in next layer in almost a linear way:

$$y_i = H\left(\sum_{j=1}^{k} v_{ij}\, x_j - \theta_i\right),$$

i.e., activation y_i is a linear function of activations x_j in previous layer, aside from the modulating function H; here θ_i is a constant for each i.

The function H has on biological grounds traditionally been assumed a sigmoid:

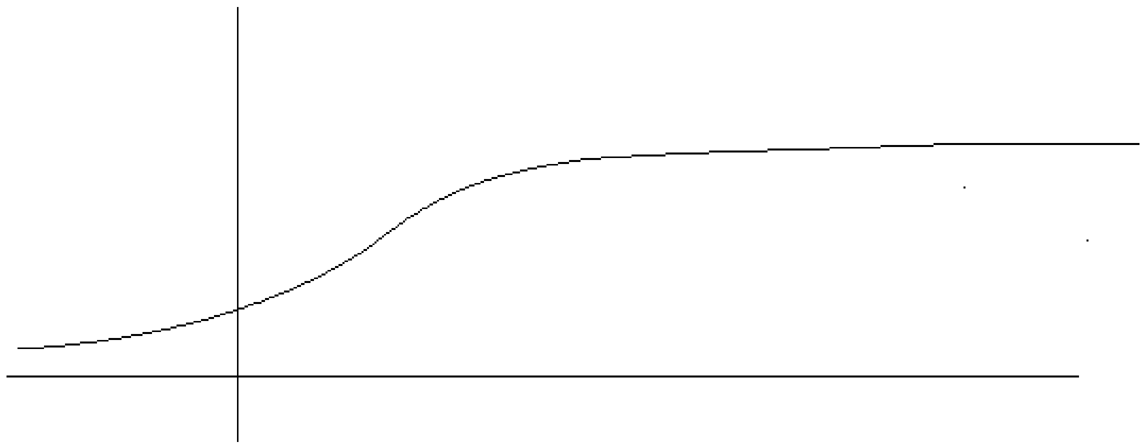


fig 3

Note that H has a finite upper bound, so that response cannot exceed some fixed constant. The activation in third layer has the form $q_i = \sum_{j=1}^{n} w_{ij}\, y_j$, which is a linear function of the y_j's.

Our initial goal is to show here that we can get an arbitrary desired output pattern q_i of activations on last layer as a function of inputs x_i in the first layer. We can impose some vector notation here -

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix}$$

will denote the vector of neuron activations in layer, while

$$V^i = \begin{bmatrix} v_{i1} \\ v_{i2} \\ \vdots \\ v_{ik} \end{bmatrix}$$

denotes vector of connection weights from the neurons in first layer to the i^{th} neuron in the second layer.

Now the activation y_i of the second layer is:

$$y_i = H\left(\sum_{j=1}^{k} v_{ij}\, x_j - \theta_i\right) = H(V^i \cdot x - \theta_i).$$

The activation q_i in the third layer is $q_i = \sum_{j=1}^{n} w_{ij}\, y_j = W^i \cdot y.$

Thus we wish to show the activation pattern

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_k \end{bmatrix}$$

on the last layer (output) can be made an arbitrary function of the input activation pattern

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix}.$$

Note the activation the of i^{th} neuron in layer 3 is:

$$q_i = \sum_{j=1}^{n} w_{ij}\, y_j = \sum_{j=1}^{n} w_{ij}\, H(V^j \cdot x - \theta_j). \tag{3.1}$$

Thus our question is: if $q = f(x)$ is defined by (3.1) (i.e., input determines output through a neural network equation), is it possible to approximate any function in this form?

Speaking in the context of the above example, if the first layer represents the retina, then if any input-output (i-o) function can be approximately encoded in the form (3.1) we can require that if x represents the visual image of a chair (vector of pixel intensities corresponding to chair), then q represent the neural pattern of intensities corresponding to articulation of the words “this is a chair.”

Mathematically, we are asking, given any function $f(x) : \mathbb{R}^k \to \mathbb{R}^k$, can we approximate $f(x)$ with arbitrary precision by a function $\overline{f}(x)$ of the form (3.1) using various measures of error, or norms (here $\mathbb{R}^k$ represents k-tuples of real numbers). What norms might we be interested in? Some are:

$$\|f-\overline{f}\|_C = \sup_{x\in\mathbb{R}^k} |f(x)-\overline{f}(x)|$$
$$\|f-\overline{f}\|_2 = \sqrt{\int dx |f(x)-\overline{f}(x)|^2}$$
$$\|f-\overline{f}\|_1 = \int dx |f(x)-\overline{f}(x)|$$

or more generally, for any fixed $p \geq 1$, $\|f-\overline{f}\|_p = \left(\int dx |f(x)-\overline{f}(x)|^p\right)^{1/p}$, where above sup denotes supremum. These norms are denoted as $C(\mathbb{R}^k)$, L^2, L^1, and finally the L^p norms, respectively. In general L^p denotes the class of functions f such that $\|f\|_p < \infty$. We say that a function $\overline{f}$ approximates another function f well in L^p if $\|\overline{f}-f\|$ is small. A sequence $\{f_n\}_{n=1}^{\infty}$ converges to a function f in L^p if $\|f_n - f\|_p \xrightarrow[n\to\infty]{} 0$.

It can be shown that the components of the above questions can be decoupled to the extent that they are equivalent to the case where there is only one q. Indeed, if any desired i-o function can be approximated in systems with one output neuron, such single-output systems can be easily concatenated into larger ones (with more outputs) which have essentially arbitrary approximable input-output properties. In any case, the configuration we assume is therefore:

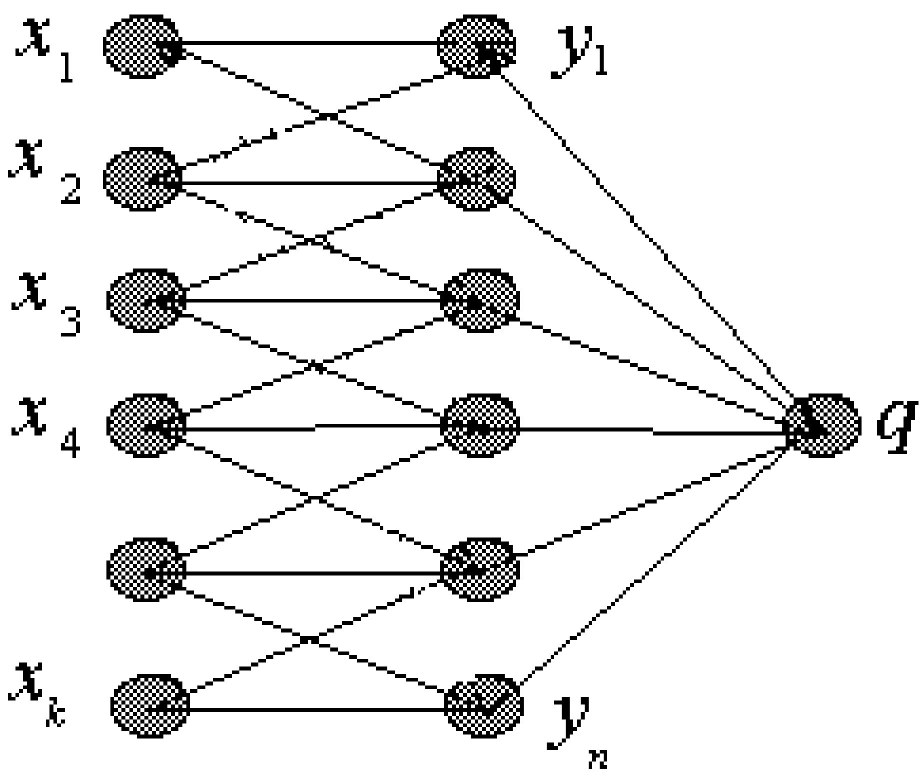


fig 4

Thus from (3.1):

$$q = \sum_{j=1}^{n} w_j\, y_j \;=\; \sum_{j=1}^{n} w_j\, H(V^j \cdot x - \theta_j). \tag{3.2}$$

The fundamental question then is: Can any function $f(x)$: $\mathbb{R}^k \to \mathbb{R}$ be approximately represented exactly in this form?

A partial answer has come in the form of the solution to Hilbert's 13th problem, which was constructed by Kolmogorov in 1957. He proved that continuous function $f: \mathbb{R}^k \to \mathbb{R}$ can be represented in the form

$$f(x) = \sum_{j=1}^{2k+1} \chi_j \left(\sum_{i=1}^{k} \psi_{ij}(x_i) \right).$$

where χ_j, ψ_{ij} are continuous functions, and ψ_{ij} are monotone and independent of f. That is, f can be represented as sum of functions each of which depends just on a sum of single variable functions.

4. Some results on approximation

In 1987, Hecht-Nielsen showed that if we have 4 layers,

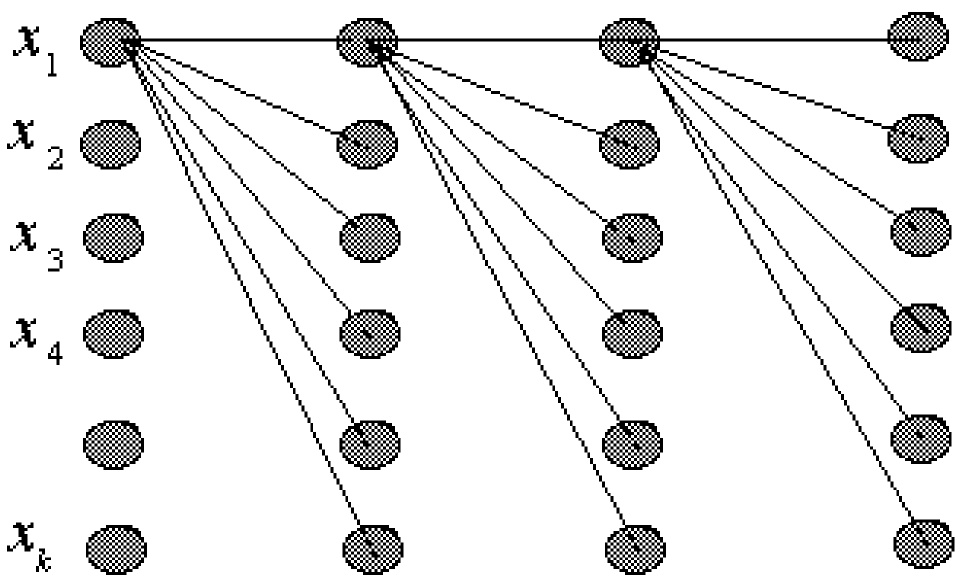


fig 5

then any continuous function $f(x)$ can be approximated within ϵ in C norm by such a network. The caveat here is that we do not yet have the techniques which will allow us to know how many neurons it will take in the middle (so-called hidden) layers to accomplish the job.

In 1989 Funahashi [Fu] proved:

Theorem: Let $H(x)$ be a non-constant, bounded, and monotone increasing function. Let K be a compact (closed and bounded) subset of $\mathbb{R}^k$, and $f(x)$ be a real-valued continuous function on K:

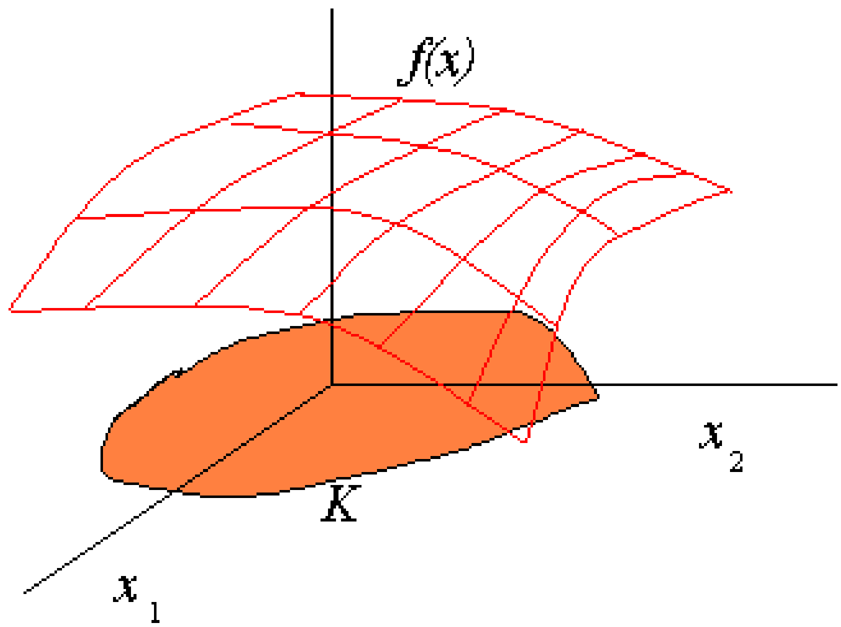


fig 6

Then for arbitrary $\epsilon > 0$, there exist real constants w_j, θ_j, and vectors V^j such that

$$\overline{f}(x) = \sum_{j=1}^{n} w_j \, H(V^j \cdot x - \theta_j) \tag{4.1}$$

satisfies

$$\| \overline{f}(x) - f(x) \|_C \ \leq \ \epsilon. \tag{4.2}$$

This is the statement that functions of the form (4.1) are *dense* in the Banach space $C(K)$ of continuous functions on K, defined in the $\| \cdot \|_C$ norm.

Corollary: Functions of the form (4.1) are dense in $L^p(K)$ for all p, $1 \leq p < \infty$. That is, given any such p, and an input-output function $f(x)$ in $L^p(K)$, i.e. such that $\int |f|^p dx < \infty$, and $\epsilon > 0$, there exists an $\overline{f}$ of the form (4.1) such that $\| f - \overline{f} \|_p \leq \epsilon$, i.e.,

$$\left(\int_K |f - \overline{f}|^p \, dx \right)^{1/p} < \epsilon$$

The caveat of this Corollary, as indicated earlier is that we may need a very large hidden layer to accomplish this approximation within ϵ. An important practical question naturally is, how large will the hidden layer need to be to get such an approximation ϵ (e.g., how complex need a neural network be so that it is able to recognize a picture of a chair)? Some of these issues are touched in the brief discussion of complexity issues [KP] at the end of this paper.

5. Newer activation functions

Recall $H(x)$ is assumed to be a sigmoid function having the form of fig. 3. The reason for this choice is biological plausibility in natural networks. There are newer ideas which have been studied. For example, what is possible with a choice of a localized $H(x)$:

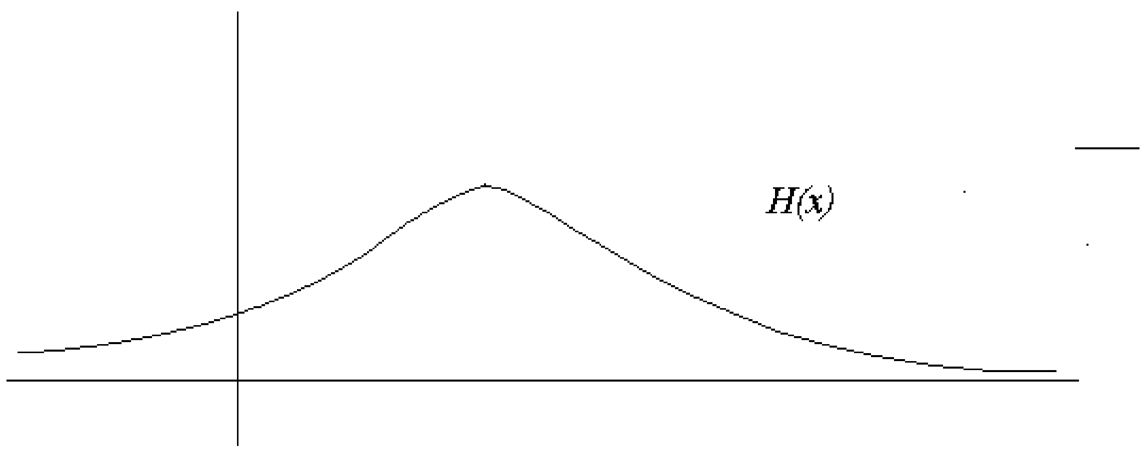


fig 7

Such a choice, though not as biologically plausible, may work better. For example, H could be a wavelet. Poggio, Girosi [GP] and others have pointed out that if $H(x) = \cos x$ on the interval $[-\pi,\pi]$, we get

$$\bar{f}(x) = \sum_{j=1}^{n} w_j \cos(V^j \cdot x - \theta_j). \tag{5.1}$$

Now choose $V^j = m = (m_1,m_2,m_3,\dots)$ where m_i are nonnegative integers, and $\theta_j = 0$. Then

$$\bar{f}(x) = \sum_{m} w_m \, cos(m \cdot x).$$

Now if

$$K = \{(x_1,x_2,\dots,x_k) : \ -\pi \le x_i \le \pi \text{ for } i = 2,3,\dots \text{ and } 0 \le x_1 \le \pi\},$$

then this is just a multivariate Fourier cosine series in x. Continuous functions can be approximated by multivariate Fourier series, and, as is well-known, we know how to find the w_j very easily:

$$w_j = \frac{4}{(2\pi)^r} \int f(x) \cos m \cdot x \, dx,$$

where r again denotes dimension. We can build the corresponding network immediately, since we know what the weights need to be if we know the i-o function. This is a very powerful concept as well as technique.

Notice that $H(x)$ here has the form

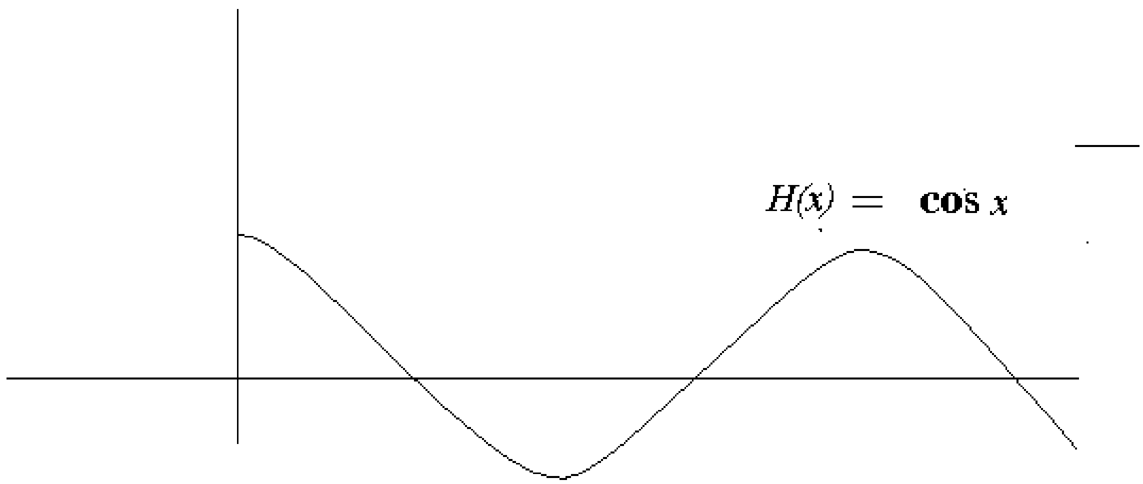


fig. 8

which is nothing like a sigmoid. Note there are questions of stability, however - if we make a small mistake in x, then $\cos m \cdot x$ may vary wildly. Nevertheless, in machine tasks this may not be as critical as in biological systems.

6. Radial basis functions

Recall that we have for the single neuron output system:

$$q = \sum_{j=1}^{n} w_j \, y_j = \sum_{j=1}^{n} w_j \, H(V^j \cdot x - \theta_j) = \bar{f}(x).$$

We will now consider newer families of activation functions and neural network protocols. Instead of each neuron in hidden layer summing its inputs, in artificial systems there is no reason why it cannot take more complicated functions of inputs, for example a function which is a bump defined on the variable $x = (x_1, \ldots, x_k)$.

Assume now that H is a fixed function in the form of a "bump" which is the multidimensional analog of figure 7:

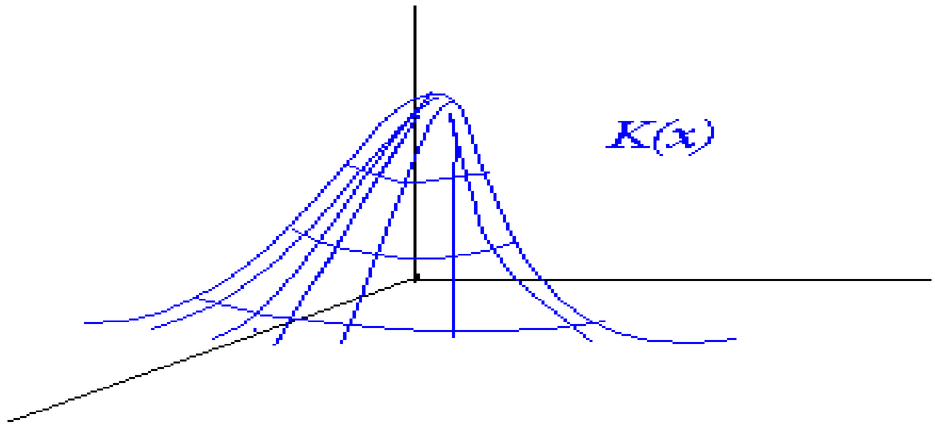


fig 9

and assume (for some fixed choice of z_i, σ) :

$$y_1 = y_1(x) = H\left(\frac{x - z_1}{\sigma}\right)$$
$$y_2 = y_2(x) = H\left(\frac{x - z_2}{\sigma}\right)$$

and in general

$$y_i = y_i(x) = H\left(\frac{x - z_i}{\sigma}\right)$$

Now (again) we give the dependence of the output neuron q which we had earlier:

$$q = \sum_i w_i\, y_i(x) = \sum_i w_i H\left(\frac{x - z_i}{\sigma}\right).$$

The goal now is to represent the i-o function f as a sum of bump functions. There are mathematical and phenomenological rationales for the choice of such a representation. The mathematical ones are discussed below, and rooted in approximation theory. It turns out that there are strong optimality properties demonstrable when the choice of H is a so-called *radial basis function* (*RBF*). Such functions, depending on the goal and the context, can simply be "bumps" such as the generic function in figure 9, or more specifically certain Green's functions, for example of the Laplace operator plus a constant. The details of the optimality of such choices are given in [PG1, PG2, MM, MB].

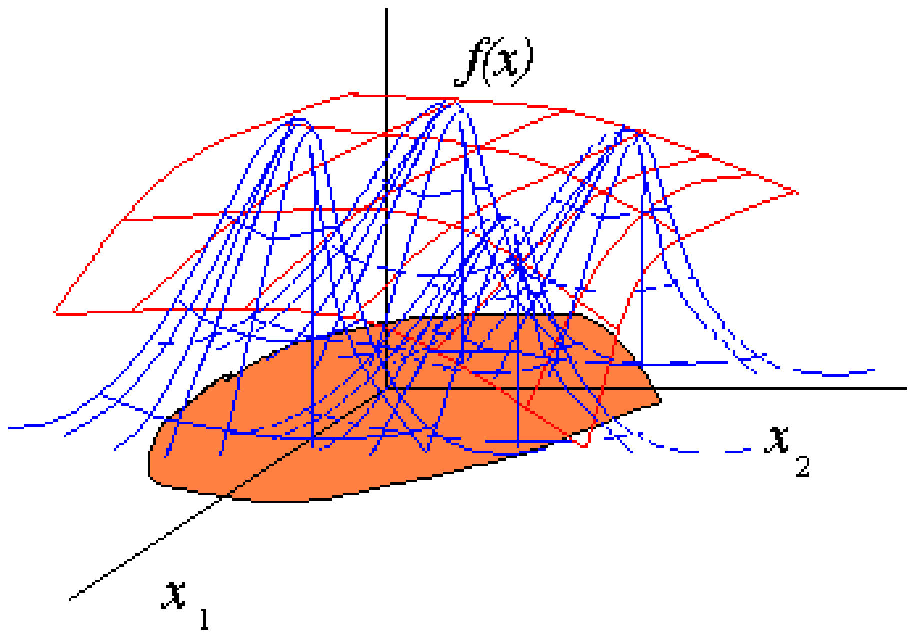


fig 10

There has been justifiably great interest in the representation of functions in this form in the mathematical and neural network communities (see references above).

The essential phenomenological rationale for the use of radial basis functions and more generally "bumps" of the form in figure 9 rests in the realm of the purpose of feedforward neural networks. Specifically, such a network is typically a "feature extractor", i.e., a machine which, given input x, has an output q which depends on whether x contains a given "feature" or not. Intuitively, very often such a feature, as represented by the corresponding i-o function f, is representable as a region of inputs x in which f has large values (i.e., the location of the "bump", or equivalently, the location in the input space of inputs which contain the object of recognition, be it a chair in a photograph or some other feature in stock market data), while it would be assumed that outside the region containing this feature f would have small values. Hence the notion of "bump". More generally, we might expect that the region of inputs we wish to recognize may be a union of regions of this form, meaning that in that case f would be a sum of such bumps. More fundamentally however, it can be proved in the context of mathematical approximation theory that under very general assumptions, *any* i-o function f can be represented as a sum of such bumps.

In terms of the neural network itself, each neuron y_i in the hidden layer has a given activation function $y_i(x)$ which depends on activations $x = (x_1, \ldots, x_k)$ in the first layer. Weights w_i connect the middle layer to the output layer (a single neuron); see fig. 4. The output is

$$q = \sum_{i=1}^{n} w_i\, y_i(x)$$

this should be a good approximation to the desired i-o function $q = f(x)$.

The best choice of weights w_i is attained by choosing w_i large if there is large "overlap" between the desired i-o function $f(x)$ and the given function $y_i(x) = H\left(\frac{x-z_i}{\sigma}\right)$ (i.e., $y_i(x)$ is large where $f(x)$ is large):

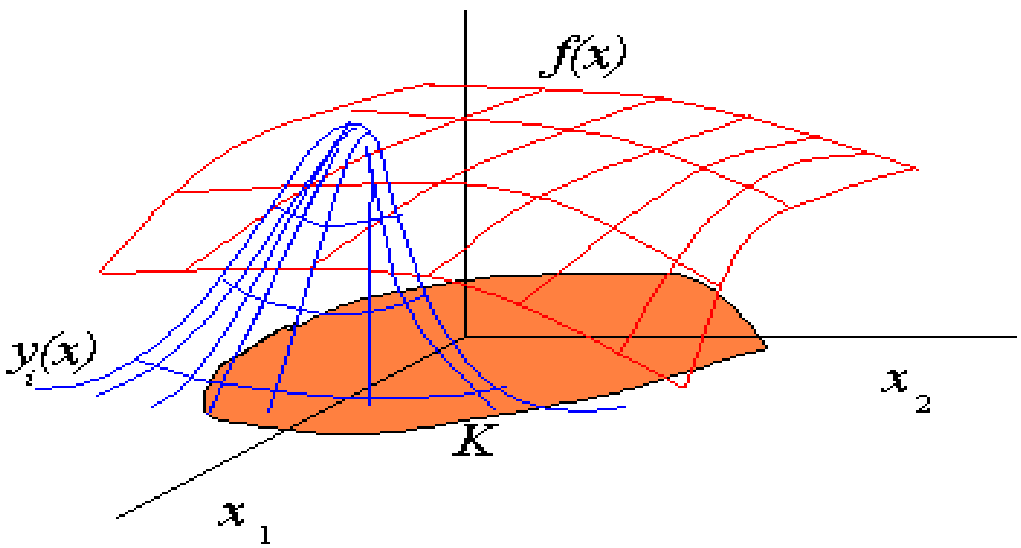


fig 11

Thus w_i measures the "overlap" between $f(x)$ and the activation function $y_i(x)$.

Usually there is one neuron y_i which has the highest overlap w_i; in adaptive resonance theory, this neuron is the "winner" and all other neurons are suppressed to have weight 0. Here however, each neuron provides a weight w_i according to its "degree of matching" with the desired $f(x)$.

Tomaso Poggio [Po], in his paper "A theory of how the brain might work" (1990) gives plausible arguments that something like this "matching" of desired i-o function against bumps like $y_i(x)$ may be at work in the brain (he discusses examples in facial recognition and motor tasks in cerebellum).

7. Mathematical analysis of RBF networks:

Mathematically, the class of functions we obtain from the above-described network has the form:

$$q(x) = \sum_{i=1}^{k} w_i \, H\Big(\frac{x - z_i}{\sigma}\Big), \tag{7.2}$$

where K is a fixed function and $\{z_i, \sigma\}$ are constants which may vary. The class of functions $q(x)$ of this form will be called $S_0(K)$. The question we will address, this time in a mathematically more precise way, is, What functions $f(x)$ can be approximated by $\overline{f}(x) = q(x)$?

Park and Sandberg [PS] answered this question (other versions of related questions have appeared previously as well):

Theorem (Park and Sandberg (1993)): Assuming H is integrable, S_0 is dense in $L^1(\mathbb{R}^r)$ if and only if $\int H \neq 0$

Recall the definition (below (4.2)) of a dense collection of functions. In particular the collection S_0 is dense in $L^1(\mathbb{R}^r)$ if for any function f in $L^1(\mathbb{R}^r)$ and number $\epsilon > 0$ there exists a $\overline{f}(x)$ in the class S_0 such that

$$\|f - \overline{f}\|_1 = \int dx |f(x) - \overline{f}(x)| < \epsilon.$$

Thus any i-o function $f(x)$ in L^1 (i.e., any integrable function) can be approximated to an arbitrary degree of accuracy by a function of the form (7.2) in L^1 norm. We now provide a sketch of the proof of the result of Park and Sandberg.

Sketch of Proof: Assume that $\int H \neq 0$. Let C_c denote continuous compactly supported (i.e., non-zero on a bounded set) functions on $\mathbb{R}^d$. Then any function in L^1 can be approximated arbitrarily well in L^1 norm by functions in C_c, i.e., C_c is dense in L^1; this is a standard fact of functional analysis.

Thus to show that L^1 functions can be arbitrarily well approximated in L^1 norm by functions in S_0, it is sufficient to show that C_c functions can be well approximated in L^1 by functions in S_0, i.e.` that for any function f in C_c there exists $\overline{f}$ in L^1 such that $\|f - \overline{f}\|_1 < \epsilon$. Choose $\epsilon_1 > 0$ and a function H_c in C_c such that

$$\| H - H_c \|_1 < \epsilon_1.$$

Let the constant $a = \frac{1}{\int H_c(x)\, dx}$. Define $\phi(x) = aH_c(x)$, so that

$$\int \phi(x)\, dx = a \int H_c(x)\, dx = 1.$$

Define $\phi_\sigma(x) = \frac{1}{\sigma^r} \cdot \phi(x/\sigma)$. A basic lemma we use here but do not prove is the fact that for f_c in C_c,

$$\| f_c - \phi_\sigma * f_c \|_1 \to 0,$$

where $*$ denotes convolution.

Thus functions of the form f_c can be arbitrarily well approximated by $\phi_\sigma * f_c$; therefore it is sufficient to show $\phi_\sigma * f_c$ can be approximated by functions in S_0 arbitrarily well. We now write (for T below sufficiently large):

$$(\phi_\sigma * f_c)(\alpha) = \int_{[-T,T]^r} \phi_\sigma(\alpha - x) f_c(x)\, dx \approx v_n(\alpha) \equiv \sum_{i=1}^{n^r} \phi_\sigma(\alpha - \alpha_i) f_c(\alpha_i) \left(\frac{2T}{n}\right)^r,$$

where α_i are all points of the form

$$[-T + \frac{2i_1 T}{n}, -T + \frac{2i_2 T}{n}, \ldots, -T + \frac{2i_r T}{n}],$$

so that we have one point in each sub-cube of size $2T/n$. The convergence of Riemann sums to their corresponding integrals implies that $v_n \underset{n\to\infty}{\to} \phi_\sigma * f_c$ pointwise; then we can use the dominated convergence theorem of real analysis to show that convergence is also in L^1. Thus we can approximate $\phi_\sigma * f_c$ by v_n. Therefore we need to show that v_n can be approximated by S_0.

To this end, we have

$$v_n(\alpha) = \frac{1}{n^r} \sum_0^{n^r} \frac{f_c(\alpha_i)(2T)^r}{\int H_c(\alpha)\, d\alpha \cdot \sigma^r} H_c\left(\frac{\alpha - \alpha_i}{\sigma}\right);$$

we then replace H_c by H (which we have shown can be made arbitrarily close), and we then have something in S_0 completing the proof of the forward implication in the theorem.

The converse of the theorem (only if part) is easy to show and is omitted here.

There is a second theorem for density in the L^2 norm. Define now $S_1 = S_1(H)$ to consist of all functions of the form $\sum_i a_i H((x - z_i)/\sigma_i)$ (the variable scale σ_i can depend on i here).

We then have the following theorems, also in [PS]:

Theorem: Assuming that H is square integrable, then $S_1(H)$ is dense in $L^2(\mathbb{R}^r)$ if and only if H is non-zero on some set of positive measure.

Theorem: Assume that H is integrable and continuous and $H^{-1}(0)$ (i.e., the set of x which H maps to 0) does not contain any set of the form $\{tw\colon t \geq 0\}$ for any vector w. Then S_1 is dense in $C(W)$ with respect to the sup norm for any compact set W.

This is an indication of what types of approximation results are possible in neural network theory. For a detailed analysis of approximation by neural nets, we refer the reader to [MM1-3; Mh1,2].

8. More general RBF networks

We now consider more general networks which we believe will have impact in future neural network architectures. Such networks are still of an RBF nature, but are also less constrained. In such networks neurons in first layer influence those in second (hidden) layer by:

$$y_i = \sum_{j=1}^{k} v_{ij} G_j(x).$$

The function $G_j(x) = G(x, z_j)$ is generally a radial basis function centered at z_j:

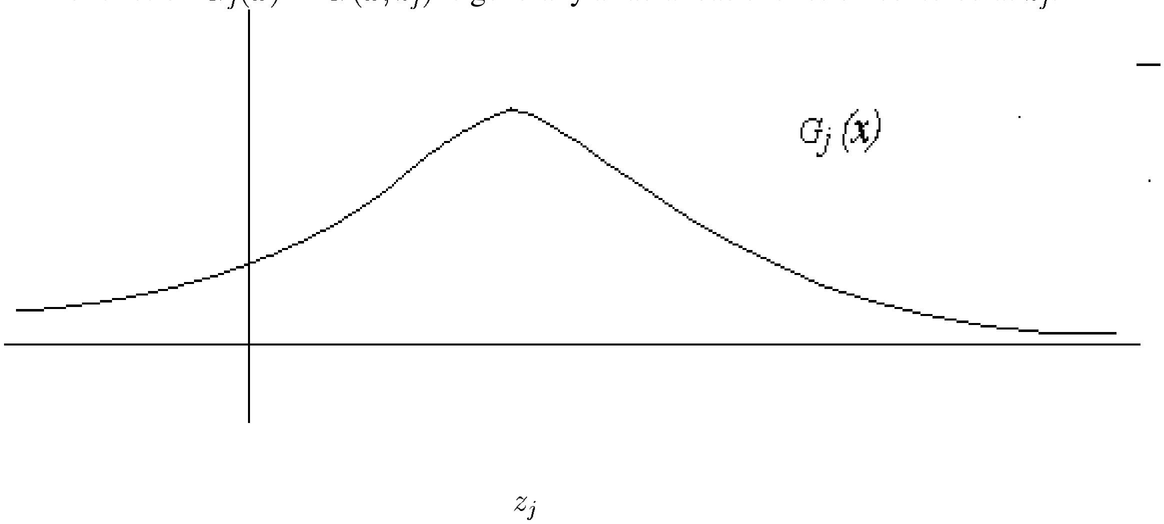
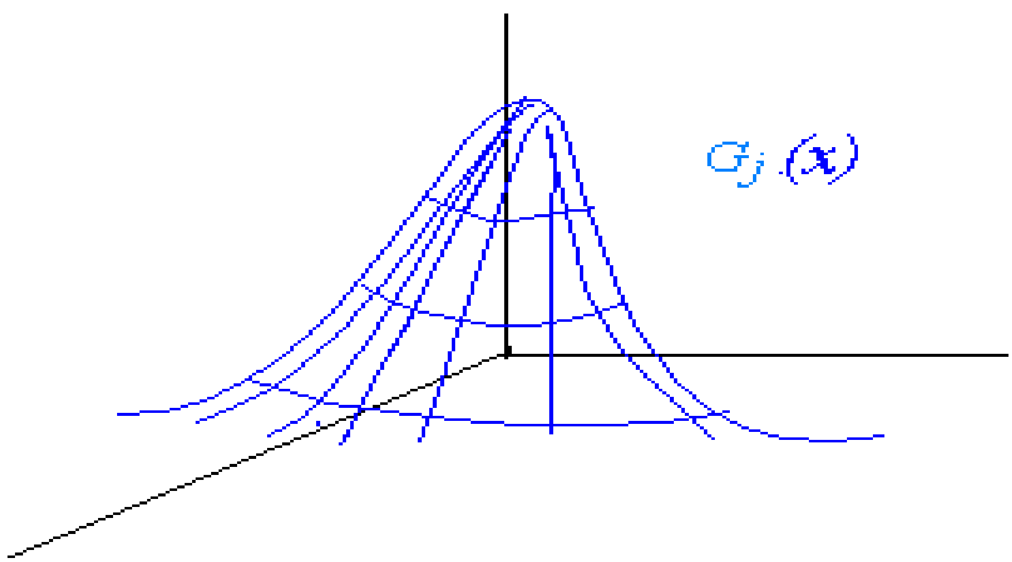


fig 12

or



:

fig 13

in higher dimension.

We want a network which will approximate a desired i-o function $f(x)$, i.e., so that

$$f(x) \approx q(x) = \sum_{j=1}^{n} w_j y_j = \sum_{j=1}^{n} w_j G_j(x).$$

Results regarding approximation have been obtained by a number of workers, including Funahashi, Hecht-Nielsen, as well as Park and Sandberg [PS], whose result is quoted above.

9. Neural complexity of networks

An important issue touched on above is: How large a hidden layer do we need to get approximation within ϵ (e.g., how complex is it to build a neural network which recognizes a chair)? The questions in this area can be formulated as follows.

Question 1. Given a fully known i-o function f and an error ϵ, what is the smallest size n of a hidden layer in a neural net which can approximate f within a given error?

Work has been done on this question by Chui, Li, Mhaskar, Micchelli, as well as Barron [Ba; CLM1, 2; MM1-3; Mh1,2]. The issues here are closely related to approximation theory.

10. Informational complexity of networks

Question 2: Given an unknown function f with a set of examples (information) $\{(\vec{x}_i, f(\vec{x}_i))\}_{i=1}^{k}$, what is the smallest number of examples k for which it is theoretically possible to estimate f within given error (regardless of the number of hidden units)?

This is the entre into the area of learning theory, a very large one in the neural network community. It is what we consider to be the second half of complexity theory for neural networks, and has been given far less treatment than the issue of neural complexity mentioned above [KP]. Learning problems are partial information issues: the network is given a number of examples of how it must compute, and it must generalize from this partial information. Thus we want to reconstruct the i-o function f from examples $f(\vec{x}_i)$.

The issues here are closely related to continuous complexity theory [TWW; TW1,2]. The essential question is, What do we do with limited information - this is at the center of learning theory for neural nets, and has close connections with the information issues arising from computer science mentioned in the above references.

Definition: Information of the form $Nf \equiv (f(\vec{x}_1), \ldots, f(\vec{x}_n))$ is called *standard information* about f. We define

$$\text{card}(N) = n$$

to be the *cardinality of information.*

One assumption standardly made is that prior knowledge of f places it into a set F_1 of functions which is convex and balanced (i.e., if f is in the set then $-f$ is as well). More precisely, the complexity-theoretic formulation is: Given an unknown function $f(\vec{x})$ in F_1, and information $y = Nf = \{(\vec{x}_i, f(\vec{x}_i))\}_{i=1}^{n}$, what is the minimum cardinality of the operator N (i.e., smallest amount n of information) for which error ϵ will occur, if we use the best possible algorithm for reconstructing f?

Formally we denote the reconstruction algorithm by

$$\phi: \mathbb{R}^d \to F_1.$$

Remarks:

1. The best possible algorithm is a mapping ϕ which maps information y into the center of the set $N^{-1}(y) \cap F_1$ consisting of choices of f which are consistent with the information y.

2. Then the radius $r(N^{-1}(y) \cap F_1)$ of this set becomes the smallest possible error of an algorithmic approximation $\phi(N(f))$, denoted $e(N, y)$, and the supremum

$$e(N) = \sup_y e(N, y).$$

is denoted as worst case error, sometimes also called the *radius of information*

Theorem: We have the error

$$e(N, y) = r(N^{-1}(y) \cap F_1)$$

Taking suprema over f:

Corollary: The maximum error $e(N)$ for information N is given by

$$e(N) \equiv \sup_y e(N, y) = \sup_y r(N^{-1}(y) \cap F_1) \equiv r(N)$$

Some conclusions are:

1. If information is only limitation and arbitrary functions can be reconstructed by our RBF network, then issues of function reconstruction reduce to the geometric ones involving radii given by the above theorem.

2. In particular the maximum possible error of the best possible algorithm for reconstructing f is the radius of a set in a normed linear space.

11. Informational and neural complexity

Definition: Define minimal error with information of cardinality n to be $e(n) \equiv \inf_{\text{card}(N)=n} e(N)$. Define the *informational ϵ-complexity of function approximation in the set F_1* to be

$$m(\epsilon) = \inf\{n : e(n) \leq \epsilon\}$$

Bounds on these quantities give informational complexities, i.e., how much information is needed to bound f within given error. These are geometric quantities related to notions involving Gelfand radii of sets in a normed linear space.

We now come to the general question involving the interaction of the two types of complexities mentioned above.

Question 3. In **Question 1** we assume network size n is the only limitation. In **Question 2** we assume example number k is only limitation. In practice both parameters are limited. How do values of n and k interact in determining network error ϵ?

Here assume F_1 is a bounded set in a *reproducing kernel Hilbert space* (RKHS), i.e., F_1 has an inner product $\langle f, g \rangle$ defined for $f, g \in F_1$, along with a function $G(x, y)$ such that for any $f \in F_1$,

$$f(x) = \langle G(x, y), f(y) \rangle$$

for all $f \in F_1$; the inner product above is in y. This condition may sound formidable, but in fact it is a minor restriction: if function values $f(x)$ are defined and continuous operations, then F_1 is an RKHS.

We consider the interaction of the two complexity theories (neural and informational). Specifically, what is n = number of neurons and k = number of examples necessary to approximate f within given error?

We wish to characterize interaction of the two complexities n and k, i.e., understand error as a function of both amount of information and available number of neurons. This is as yet not a well developed theory; see Poggio and Girosi [PG1,2].

The aims of this theory are:

- To develop algorithms which optimize information (i.e., number of examples) and number of neurons necessary to compute i-o functions f in given classes.

- To show they apply to examples of interest in constructing RBF neural networks.

- To show radial basis function (RBF) algorithms are best-possible in very strong sense, and do define and describe theoretical and practical applications.

The conclusions of the theory in [KP] are:

- Relationships between informational and neural complexity can be simply bounded.

• Thus the two can be studied independently before details of interaction.

12. Theoretical results

We will show algorithms using radial basis functions are optimal from the standpoint of informational and neural complexity. Our results can be used in practical situations for upper and lower bounds on numbers of examples and neurons for systems with given i-o functions.

The main theoretical results require a choice of F_1 as a convex, balanced set of possible i-o functions, and $f \in F_1$. For discussion of approximation of the function f, we define:

• $e(k, n) =$ minimal error given n neurons and k examples

• $e_{\text{neur}}(n) =$ minimal error given n neurons with unlimited examples (neural error)

• $e_{\text{info}}(n) =$ minimal error given k examples with unlimited neurons (informational error)

Theorem 1: For an RBF network,

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

Corollary: For an RBF network,

$$e(n, k) = \Theta(e_{\text{neur}}(n) + e_{\text{info}}(k)), \tag{12.1}$$

i.e., the $e(n, k)$ and $e_{\text{neur}}(n) + e_{\text{info}}(k)$ have the same order.

Specifically, (12.1) states that the ratio $\frac{e(n,k)}{e_{\text{neur}}(n)+e_{\text{info}}(k)}$ is bounded above and below by fixed positive constants.

Corollary: A necessary condition for error e$(n,k) \leq \epsilon$ is that $e_{\text{neur}}(n)$, $e_{\text{info}}(k) < \epsilon$, while a sufficient condition is $e_{\text{neur}}(n)$, $e_{\text{info}}(k) \leq 1/2\,\epsilon$.

Theorem 2 : Informational complexity dominates neural complexity. Specifically for all n,

$$e_{\text{neur}}(n) \leq e_{\text{info}}(n).$$

Conjecture: $e_{\text{neur}}(n)$ is comparable with $e_{\text{info}}(k)$ for standard RBF networks, i.e., also

$$e_{\text{info}}(n) \leq C\, e_{\text{neur}}(n)$$

If true this would reduce the work currently done on neuronal complexity to the large body of work in continuous complexity theory and bounds there.

Theorem 3: If the number n of neurons is larger than the number k of examples, then

(a) A strongly optimal algorithm for constructing the weights w_j of the network approximating $f(x)$ is given by a linear combination $\mathbf{s}_{\vec{y}} = \sum_j w_j G(t_j, \cdot\,)$, where $G(t, \cdot\,)$ is the family of radial basis functions, and $\vec{y}$ is the information vector, $y_i = f(x_i)$.
(b) For information with bounded noise, $\mathbf{s}_{\vec{y}}$ is an optimal algorithm for the choice of c_j for which $\mathbf{s}_{\vec{y}} = g$ optimizes a regularization functional of the form $\mathcal{F}(g) = \lambda \|g\|_F^2 + \sum_i |y_i - g(x_i)|^2$

Such algorithms have been studied by Poggio and Girosi [PG1, 2].

Definition: An *almost optimal* algorithm is one which is optimal up to a constant factor, i.e., whose error is at most a constant factor times the optimal error.

Let Al be any almost optimal algorithm for finding neural network approximations with *full* information.

Theorem 4:
(a) The composite algorithm $Al \circ \mathbf{s}_{\vec{y}}$ yields an approximation which is almost optimal in the number k of neurons and the number n of examples.
(b) If the above conjecture is true, then $\mathbf{s}_{\vec{y}}$ itself yields an almost optimal approximation.

Classical (and more difficult) algorithms for programming neural networks include backpropagation and the Boltzmann machine for neural nets.

13. Examples

A. *Building a control system controlling homeostatic parameters of an industrial mixture:*

Imagine an industrial system which can control the following input variables:

• Temperature
• Humidity
• Specific chemical contents
• Other related parameters

Assume that the output variable in this case is the ratio of elasticity and strength of a plastic produced from the above mixture. Combinations of input variables may have unpredictable effects on output variable. Such effects include:

• binary correlations
• tertiary correlations
• more complicated interactions of inputs

The goal is to build a neural network for which

• the input $\overrightarrow{x} = (x_1, \ldots, x_d)$ is a vector of input parameters
• the output is an approximation $q \approx f(\vec{x})$, which is the ratio of elasticity and strength

(f here is an unknown function)

We nevertheless have experimental data of the form

$$\{(\vec{x}_i, f(\vec{x}_i))\}_{i=1}^k$$

from previous runs of the equipment. We want to build network which

• runs on a moderate size computer
• requires a moderate number of experiments to learn
• will correctly predict the elasticity/strength ratio q from the homeostatic parameters $\vec{x} = (x_1, \ldots, x_d)$.

Thus k, the number of examples, n, the number of neurons in the simulation, are limited.

We specify an error tolerance ϵ for q, and wish to optimize our ϵ-network by optimizing some function $h(n, k)$. This function might depend on:

• only the neuron number n
• only the example number k
• a linear combination of the two (with weights determined by relative difficulty of increasing computational scale versus obtaining information)
• a more complicated function.

Given the above theoretical results we summarize some conclusions whose details are given in [KP]:

• Knowing the dependence of ϵ on n and k from the above results allows us to do this.
• The optimal algorithms mentioned in the theorems above allow the above best possible error tolerance ϵ to be implemented in a computable algorithm
• Such learning algorithms are practical, optimal, and much faster than classical neural learning algorithms, e.g., backpropagation; Boltzmann machine.

B. *Example neural network which studies purchasing patterns of people using mail order corporations.*

This example is a modification of one which has been presented to one of the authors as an issue being worked on by a marketing consulting firm in the U.S. The paradigm is as follows. Corporations currently share large databases on purchasing patterns of consumers. Correct "mining" of such data can produce large numbers of sales to clients very likely to purchase given product classes. A reasonable approach to predicting such patterns would be as follows:

• create a tree structure on the family of products under consideration
(e.g., one node would be appliances, a subnode would be kitchen appliances, and a sub-subnode ovens.),
• input variables in the form of a vector $\vec{x} = (x_1, \ldots, x_d)$ for a given individual include dollar quantities of purchases .
• The desired output is $f(x)$, the probability that the consumer will purchase a given target product, e.g., a blender, toaster, or oven.

The goal here is to find an algorithm yielding a network with the smallest practical error ϵ for our given information cardinality k and computationally limited neuron cardinality n.

Features of this particular scenario include:

• A large dimension d of the input data (many products can be purchased),
• An unchangeable size k of the learning set $\{(\vec{x}_k, f(\vec{x}_k))\}$
• The above results yield the minimal computational ability required to find a given purchase probability within given tolerance
• The above algorithms can be used to find the best utilization of information and computational resources to compute $f(\vec{x})$.

14. Remarks on our assumptions:

We have here assumed that the i-o function f belongs to a function class F. A simple example of such an F is $B_m(L_s^\infty)$, the set of functions with s derivatives bounded by a constant m. If a function f^* with small norm in the space F can "well" approximate the unknown i-o function f, it is unnecessary that f be exactly smooth, or exactly belong to the indicated class. There must be a global fit which is acceptable, and under such assumptions these theorems can be applied.

As an example, in the above homeostatic system, assume we know that the variation of the output quality $f(x)$ is such that f can be approximated by a 10 times differentiable function (e.g., a polynomial) whose first 10 derivatives are smaller than 15 (in some appropriate scale). In the context of polynomial approximation, for example, this places bounds on the coefficients of the polynomials. Such bounds in this case would be a reasonable way to "guess" the nature of the unknown function. We remark that such bounds would be a heuristic process, with techniques and guidelines.

Most importantly, the guesses made through such a process can be validated a posteriori with the data subsequently obtained. For example, that the gradient of the data is bounded by, e.g., 15 units, can be verified experimentally; higher derivatives can be bounded similarly.

The results of such an analysis might have the following features in a concrete situation:

• The I-O function f can be well-approximated by a function in the Sobolev space L_{10}^∞, in the ball of radius 15. This would provide our set F_1 and function space F.

• With this plus a given error tolerance, say $\epsilon = .1$, we can compute upper and lower bounds for the informational and neural complexity of our problem using the above results.
• We can then use the related RBF algorithm (mentioned in Theorem 3 above).
• For example, such a calculation might show we need 10,000 examples and a network with 10,000 neurons (typically optimality is achieved with equal numbers by the above results).

We mention a caveat here: we still will need to decide what examples to use; continuous complexity theory yields techniques only for choosing good data points as examples.

References

[Ba] A.R. Barron, Universal approximation bounds for superposition of a sigmoidal function, preprint, Yale University.

[CLM1] C.K. Chui, X. Li, and H.N. Mhaskar, Neural networks for localized approximation, Center for Approximation Theory Report 289, 1993.

[CLM2] C.K. Chui, Xin Li, and H. N. Mhaskar, Limitations of the approximation capabilities of neural networks with one hidden layer, Advances in Computational Mathematics **5** (1996), 233-243.

[CL] C. K. Chui and X. Li, Approximation by ridge functions and neural networks with one hidden layer, *J. Approximation Theory* **70** (1992), 131-141.

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

[G1] Stephen Grossberg, *Studies of Mind and Brain,* Reidel Publishing Co., Boston, 1982

[G2] Stephen Grossberg, *The Adaptive Brain,* North-Holland, Amsterdam, 1987

[Ha] Mohamad Hassoun, *Fundamentals of Artificial Neural Networks*, M.I.T. Press, Cambridge, MA., 1995

[He] D. Hestenes, How the Brain Works: the next great scientific revolution. In C.R. Smith and G.J. Erickson (eds.), *Maximum Entropy and Bayesian Spectral Analysis and Estimation Problems*, Reidel, Dordrecht/Boston (1987),173-205.

[KP] M. Kon and L. Plaskota, Informational Complexity of Neural Networks, preprint.

[MM1] H.N. Mhaskar and C.A. Micchelli, Approximation by superposition of sigmoidal and radial basis functions, *Advances in Applied Mathematics* **13** (1992), 350-373.

[MM2] H.N. Mhaskar and C.A. Micchelli, Dimension independent bounds on the degree of approximation by neural networks, *IBM J. Research and Development* **38** (1994), 277-284.

[MM3] H. Mhaskar and C. Micchelli. Degree of approximation by neural and translation networks with a single hidden layer *Advances in Applied Mathematics* **161** (1995)**,** 151-183.

[Mh1] H.N. Mhaskar., Neural networks for optimal approximation of smooth and analytic functions, *Neural Computation* **8** (1996), 164-177.

[Mh2] H.N. Mhaskar, Neural Networks and Approximation Theory, *Neural Networks* **9** (1996), 721-722.

[MB] Micchelli, C.A. and M. Buhmann, On radial basis approximation on periodic grids, *Math. Proc. Camb. Phil. Soc.* **112** (1992), 317-334.

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

[PG1] T. Poggio, and F. Girosi, Regularization algorithms for learning that are equivalent to multilayer networks, *Science* **247** (1990), 978-982.

[PG2] T. Poggio and F. Girosi, A theory of networks for approximation and learning, A.I. Memo No. 1140, M.I.T. A.I. Lab, 1989.

[Po] T. Poggio. A theory of how the brain might work, In *Proc. Cold Spring Harbor meeting on Quantitative Biology and the Brain*, 1990.

[TWW] Traub, J., G. Wasilkowski, and H. Woźniakowski, *Information-Based Complexity*, Academic Press, Boston, 1988.

[TW1] Traub, Joseph and Henryk Wózniakowski, *A General Theory of Optimal Algorithms*, Academic Press, New York, 1980.

[TW2] Joseph Traub and Henryk Woźniakowski, Breaking intractability, *Scientific American* **270** (Jan. 1994), 102-107.

[MB] Micchelli, C.A. and M. Buhmann, On radial basis approximation on periodic grids, *Math. Proc. Camb. Phil. Soc.* **112** (1992), 317-334.

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

[PG1] T. Poggio, and F. Girosi, Regularization algorithms for learning that are equivalent to multilayer networks, *Science* **247** (1990), 978-982.

[PG2] T. Poggio and F. Girosi, A theory of networks for approximation and learning, A.I. Memo No. 1140, M.I.T. A.I. Lab, 1989.

[Po] T. Poggio. A theory of how the brain might work, In *Proc. Cold Spring Harbor meeting on Quantitative Biology and the Brain*, 1990.

[TWW] Traub, J., G. Wasilkowski, and H. Woźniakowski, *Information-Based Complexity*, Academic Press, Boston, 1988.

[TW1] Traub, Joseph and Henryk Wózniakowski, *A General Theory of Optimal Algorithms*, Academic Press, New York, 1980.

[TW2] Joseph Traub and Henryk Woźniakowski, Breaking intractability, *Scientific American* **270** (Jan. 1994), 102-107.