Neural Networks Preliminaries:

Notation: Write $x = (x_1, x_2, \ldots x_d)$, and in integrals

$$dx = dx_1 dx_2 \ldots dx_d.$$

Recall:

Definition: L^p norm:

$$\|f(x)\|_p = \left(\int dx \, |f(x)|^p \right)^{1/p}.$$

$C(\mathbb{R}^d) =$ continuous functions on $\mathbb{R}^d$. Define norm on $C(\mathbb{R}^d)$ by

$$\|f(x)\|_\infty = \sup |f(x)|.$$

A set of functions S is *dense* in L^p norm if every $f \in L^p$ can be arbitrarily well approximated by functions in S

A set of functions is dense in $C(\mathbb{R}^d)$ norm if for every $f \in C(\mathbb{R}^d)$ there is a sequence of functions $f_i \in C(\mathbb{R}^d)$ such that $\|f_i - f\|_\infty \underset{i \to \infty}{\longrightarrow} 0$.

A function $f(x)$ defined on $\mathbb{R}^d$ is *compactly supported* if the set of $x \in \mathbb{R}^d$ for which $f(x) \neq 0$ is bounded (i.e., satisfies $\|x\|^2 = \sum_i x_i^2 < M$ for some fixed $M > 0$).

A function $f(x)$ defined for $x \in \mathbb{R}$ is *monotone increasing* if $f(x)$ does not decrease as x gets larger.

Neural Networks and Radial Basis Functions

1. Neural network theory

1. Since artificial intelligence (using Von Neumann processors) has failed to produce true intelligence, we wish work towards computational solutions to problems in intelligence

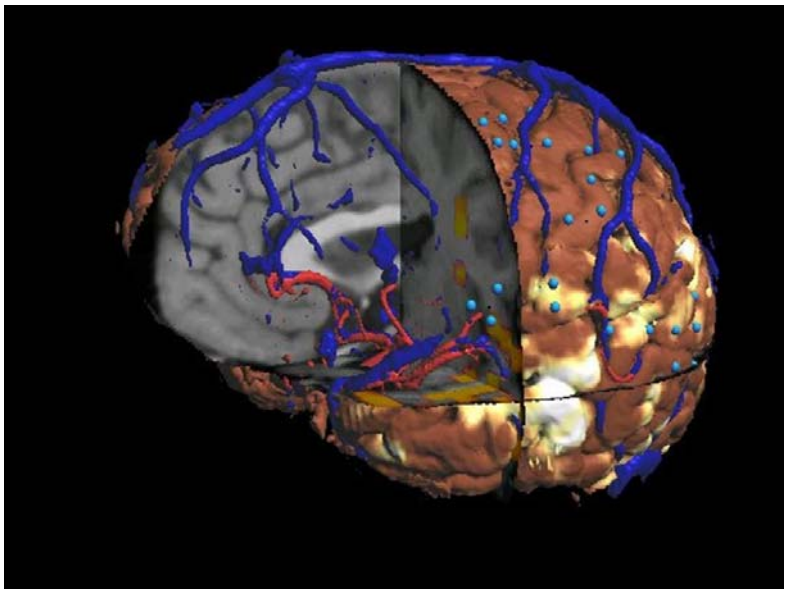
2. Neural network theory has held that promise.
Existence proof: neural nets work in people, insects, etc.

Applications of neural nets: Physics, biology, psychology, engineering, mathematics

3. A basic component of many neural nets: feed-forward neural networks.

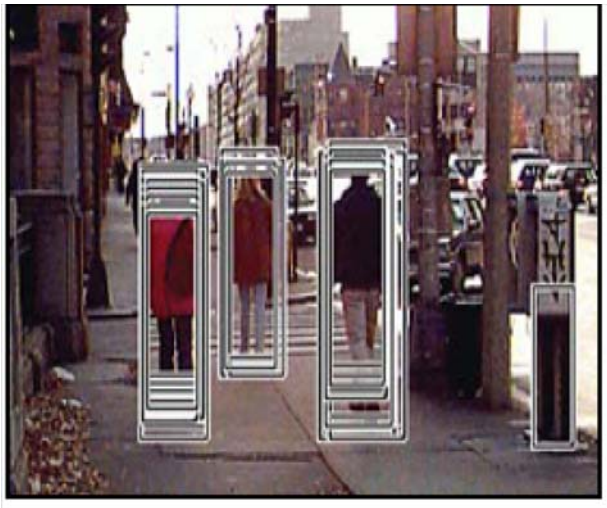
The role of learning theory has grown a great deal in:

- Mathematics
- Statistics
- Computational Biology
- Neurosciences, e.g., theory of plasticity, workings of visual cortex



Source: University of Washington

- Computer science, e.g., vision theory, graphics, speech synthesis

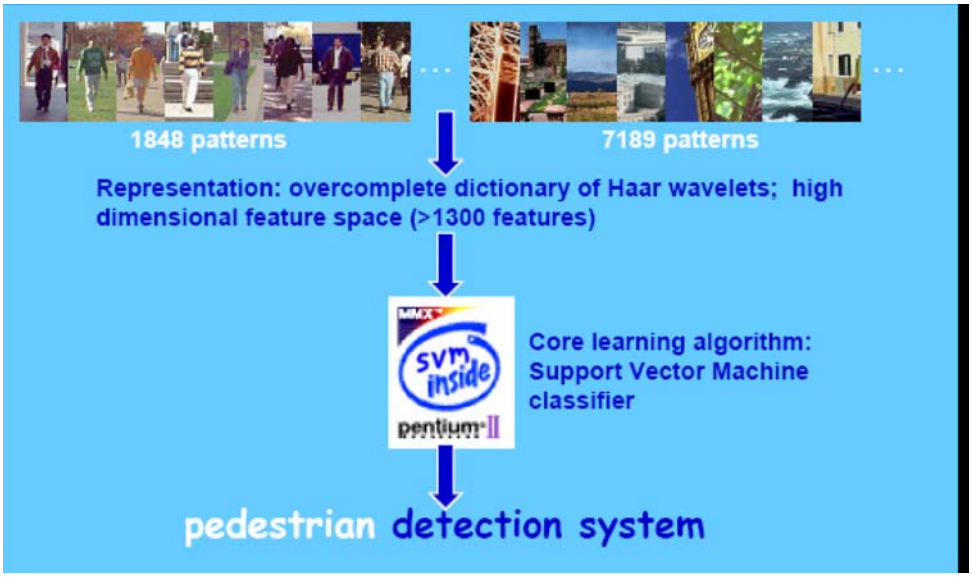


Source: T. Poggio/MIT

Face identification:

MIT

People classification or detection:



Poggio/MIT

What is the theory behind such learning algorithms?

2. The problem: Learning theory

Given an unknown function $f(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}$, learn $f(\mathbf{x})$.

Example 1: **x** is retinal activation pattern (i.e., $x_i =$ activation level of retinal neuron i), and $y = f(\mathbf{x}) > 0$ if the retinal pattern is a chair; $y = f(\mathbf{x}) < 0$ otherwise.

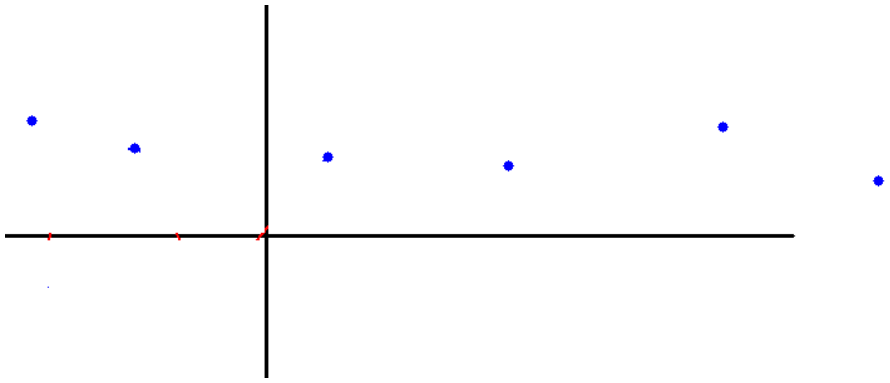
[Thus: want concept of a chair]

Given: examples of chairs (and non-chairs): $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, together with proper outputs $y_1, \ldots, y_n$. This is the information:

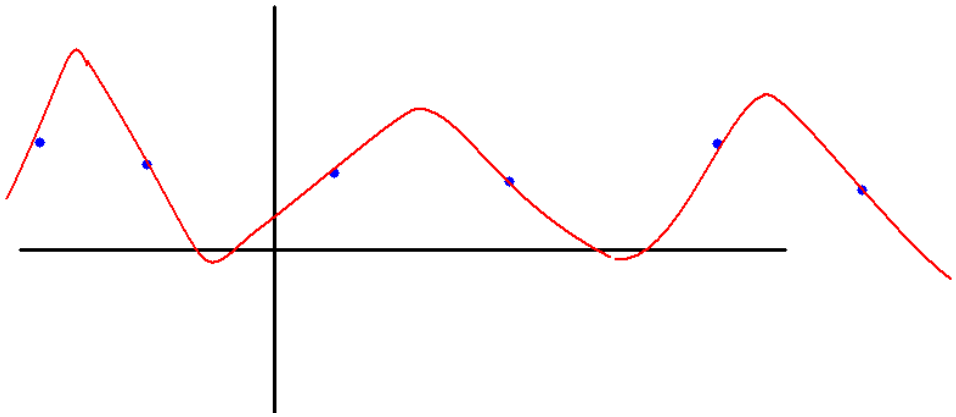
$$Nf = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))$$

Goal: Give best possible estimate of the unknown function f, i.e., try to learn the concept f from the examples Nf.

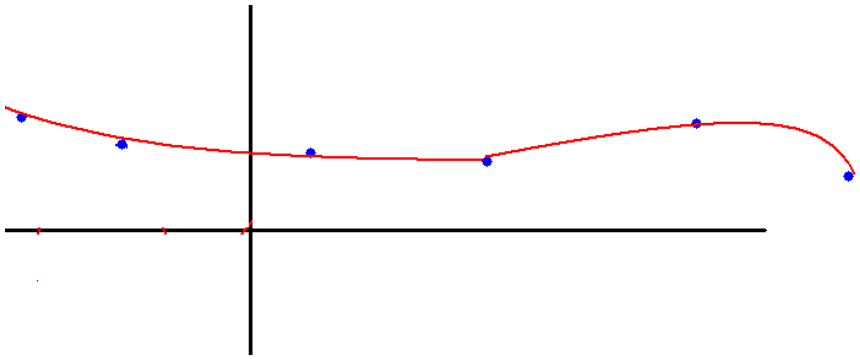
But: given pointwise information about f not sufficient: which is the "right" $f(x)$ given the data Nf below?



(a)



(b)



[How to decide?]

3. The neural network approach to learning:

Feed-forward neural network:

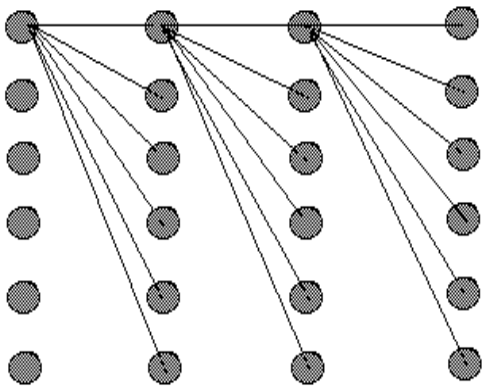


fig. 58

Layer = vertical row of neurons

Neurons in first layer influence neurons in second layer.
Neurons in second layer influence neurons in third layer.
Etc.

First layer contains “input", i.e., we control activations of its neurons.

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 and Hecht-Nielsen 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:

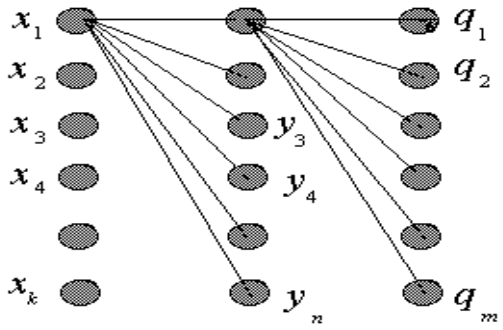


fig. 59

Now consider only 3 layer networks.

x_i = activation level (either chemical or electrical potential) of i^{th} neuron in first layer

y_i = activation level of i^{th} neuron in second layer

q_i = activation level of i^{th} neuron in third layer

ν_{ij} = strength of connection (weight) from j^{th} neuron in layer 1 to i^{th} neuron in layer 2.

w_{ij} = strength of connection (weight) from j^{th} neuron in layer 2 to i^{th} neuron in layer 3.

Example: First layer is retina and is the illumination level at the neuron x_i. This is input layer (light shines on retina and activates it).

Last layer is speech center (neurons ultimately connected to mouth), and its pattern q_i of neuron activations corresponds to verbal description about to be delivered of what is seen in first layer.

2. Neuron interaction rule

Neurons in one layer influence those in next layer in almost a linear way:

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

i.e., activation y_i is a linear function of activations x_j in previous layer, aside from function H.

here θ_i = constant for each i.

The function H is a sigmoid:

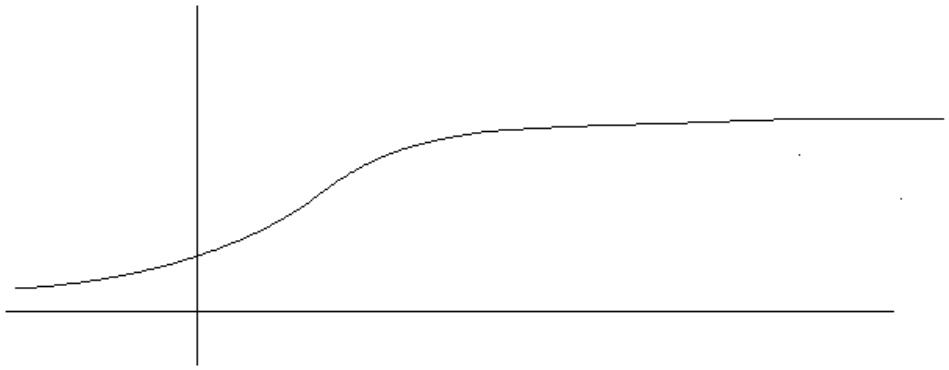


fig 60

Note that H has an upper bound, so response cannot exceed some constant.

Activation in third layer:

$$q_i = \sum_{j=1}^{n} w_{ij}\, y_j$$

$=$ linear function of the y_j's

Goal: show 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.

Vector notation:

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

$=$ vector of neuron activations in layer 1.

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

$=$ vector of connection weights from neurons in first layer to *i*th neuron in the second layer

Now: activation y_i in second layer is:

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

Activation q_i in the third layer is:

$$q_i = \sum_{j=1}^{n} w_{ij}\, y_j = W^i \cdot y.$$

Goal: to show that the pattern of activations

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

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

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

on the first layer.

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

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

Thus question is: if $q = f(x)$ is defined by (1) (i.e., input determines output through a neural network equation), is it possible to approximate any function in this form?

Ex. If first layer = retina, then can require that if $x =$ visual image of chair (vector of pixel intensities corresponding to chair),

then $q =$ neural pattern of intensities corresponding to articulation of the words "this is a chair"

Equivalently: Given any function $f(x) : \mathbb{R}^k \to \mathbb{R}^k$, can we approximate $f(x)$ by a function of the form (1) in

(a) $C(\mathbb{R}^k)$ norm?
(b) L^1 norm?
(c) L^2 norm?

Can show that these questions are equivalent to case where there is only one q:

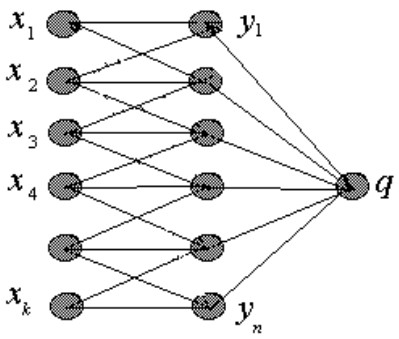


fig 61

Then have from (1):

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

Question: Can any function $f(x) : \mathbb{R}^k \to \mathbb{R}$ be approximately represented in this form?

Partial Answer: Hilbert's 13th problem.

1957: Kolmogorov solved 13th problem by proving that any 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, 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.

3. Some results

1987: Hecht-Nielsen remarks that if we have 4 layers

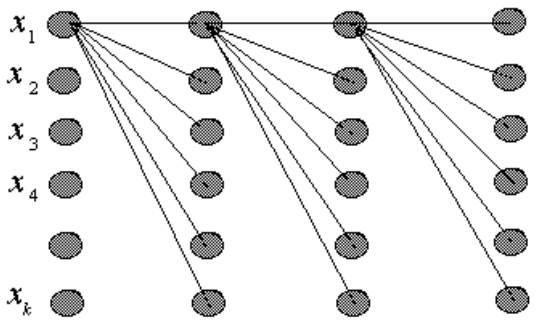


fig 62

any function $f(x)$ can be approximated within ϵ in $C(\mathbb{R}^d)$ norm by such a network.

Caveat: we don't know how many neurons it will take in the middle.

1989: Funahashi 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:

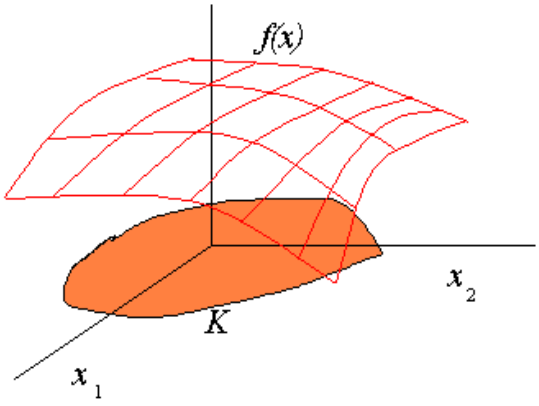


fig 63

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

$$\overline{f}(x) = \sum_{j=1}^{n} w_j \, H(V^j \cdot x - \theta_j) \qquad (2)$$

satisfies

$$\|\overline{f}(x) - f(x)\|_\infty \leq \epsilon.$$

Thus functions of the form (2) are dense in the Banach space $C(K)$, defined in the $\| \cdot \|_\infty$ norm.

Corollary: Functions of the form (3) are dense in $L^p(K)$ for all $p, 1 \leq p < \infty$.

That is, given any such p, and an i-o (input-output) function $f(x) \in L^p(K)$ and $\epsilon > 0$, there exists an $\overline{f}$ of the form (3) such that $\| f - \overline{f} \|_p \leq \epsilon$

Caveat: may need very large *hidden layer* (middle layer).

Important question: How large will the hidden layer need to be to get an approximation within ϵ (i.e., how complex is it to build a neural network which recognizes a chair)?

4. Newer activation functions:

Recall $H(x)$ is assumed to be a sigmoid function:

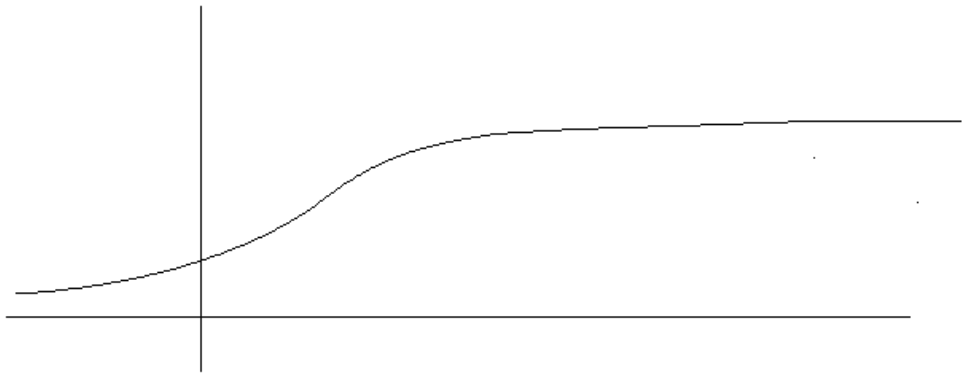


fig 64

Reason: biological plausibility.

Newer idea: how about a localized $H(x)$

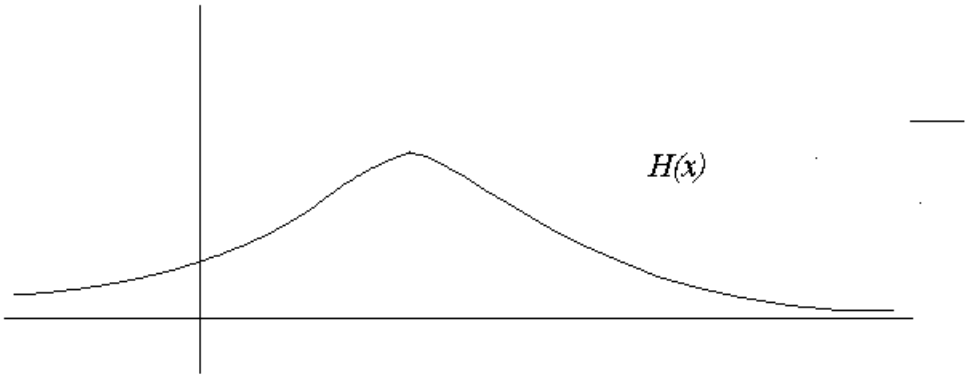


fig 65

Not as biologically plausible, but may work better. E.G., H could be a wavelet?

Poggio, Girosi, others pointed out: if $H(x) =$ cos x on $[0, \infty)$, get

$$\overline{f}(x) = \sum_{j=1}^{n} w_j \cos\,(V^j \cdot x - \theta_j). \qquad (3)$$

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

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

Now if

$K = \{(x_1, x_2, \dots, x_k) | \; -\pi \leq x_i \leq \pi$ if $i = 2, 3, \dots$ and $0 \leq x_1 \leq \pi\}$,

then this is just a multivariate Fourier cosine series in x. We know that continuous functions can be approximated by multivariate Fourier series, and we know how to find the w_j very easily:

$$w_j = \left(\tfrac{2}{\pi^k} \right) \int f(x) \cos\, m \cdot x \, dx.$$

We can build the network immediately, since we know what the weights need to be if we know the $i - o$ function. Very powerful.

Notice that $H(x)$ here is:

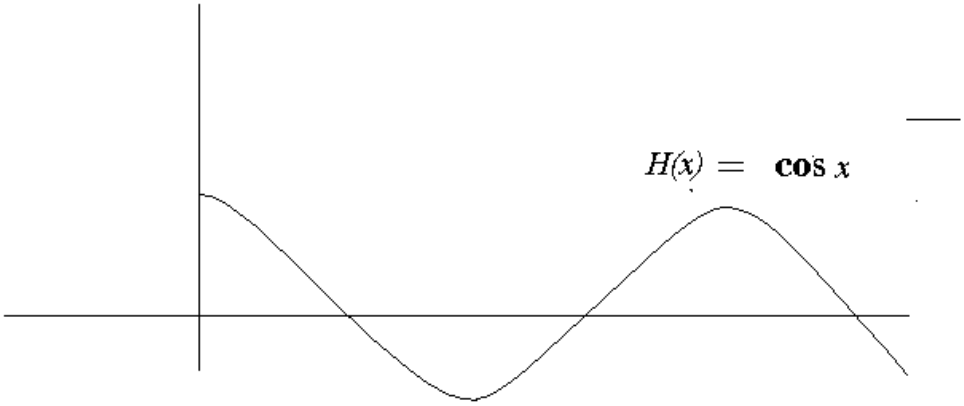


fig. 66

nothing like a sigmoid.

Note: questions of stability however - make a small mistake in x, and cos $m \cdot x$ may vary wildly.

However, in machine tasks this may not be as crucial.

Why does this not solve the approximation problem once and for all? It ignores learning. The learning problem is better solved by:

5. Radial basis functions

Had:

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

Now consider newer families of activation functions, and neural network protocols:

Instead of each neuron in hidden layer summing its inputs, perhaps it can take more complicated functions of inputs:

Assume now that K is a fixed function:

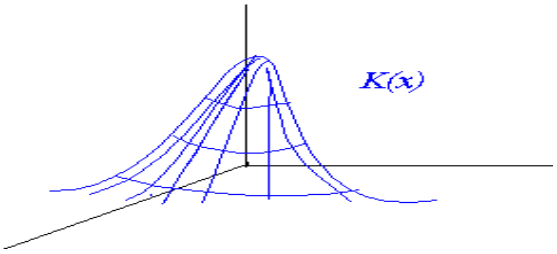


fig 67

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

$$y_1 = y_1(x) = K\left(\frac{x - z_1}{\sigma}\right)$$

$$y_2 = y_2(x) = K\left(\frac{x - z_2}{\sigma}\right)$$

in general

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

Now write (again)

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

Goal now is to represent $i - o$ function f as a sum of bump functions:

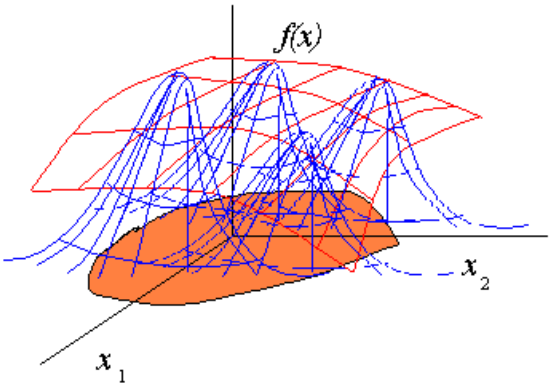


fig 68

Functions K are called *radial basis functions*.

Neural networks and approximation people are interested in these.

Idea behind radial basis functions:

Each neuron y_i in hidden layer has given activation function $y_i(x)$ which depends on activations $x = (x_1, \ldots x_k)$ in first layer.

Weights w_i connect middle layer to output layer (single neuron)

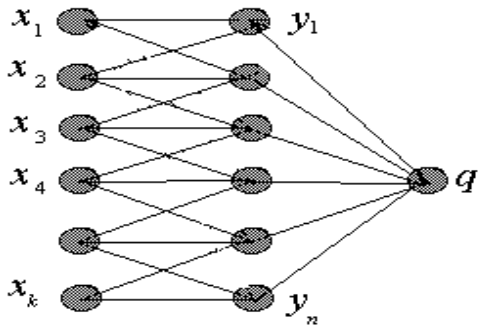


fig 69

Output is:

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

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

Can check:

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

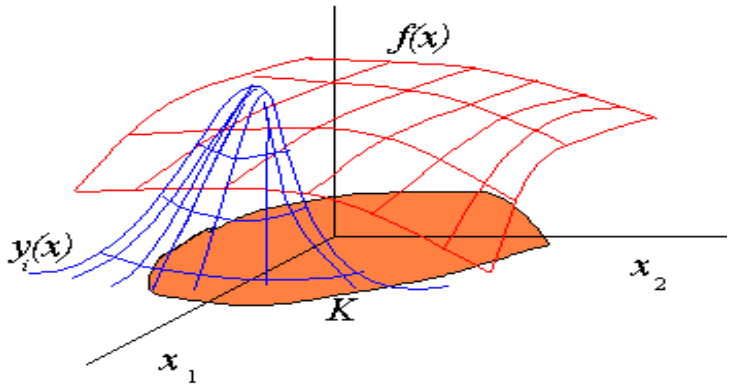


fig 70

Thus w_i measures "overlap" between $f(x)$ and 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 have weight 0.

Here we have that each neuron provides a weight w_i according to the "degree of matching" of neuron with desired $i-o$ function $f(x)$.

Poggio: "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 (facial recognition; motor tasks in cerebellum).

6. Mathematical analysis of RBF networks:

Mathematically, class of functions we obtain has the form:

$$q(x) = \sum_{i=1}^{k} w_i \, K\left(\frac{x - z_i}{\sigma}\right), \tag{4}$$

where K is a fixed function and $\{z_i, \sigma\}$ are constants which may vary.

Note: class of functions $q(x)$ of this form will be called $S_0(K)$.

Again: what functions $f(x)$ can be approximated by $\overline{f}(x)$?

Park and Sandberg (1993) answered this question (other versions previously):

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

That is, any i-o function $f(x)$ in L^1 (i.e., an integrable function) can be approximated to arbitrary degree of accuracy by a function of the form (5) in L^1 norm.

Proof: Assume that $\int K \neq 0$.

Let $\kappa =$ continuous compactly supported functions on $\mathbb{R}^k$.

Then any L^1 function can be approximated arbitrarily well in L^1 by functions in κ, i.e., κ is dense in L^1.

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 functions in κ can be well approximated in L^1 by functions in S_0.

Choose $\epsilon_1 > 0$ and a function $K_c \in \kappa$ such that

$$\|K - K_c\|_1 < \epsilon_1 \ .$$

Let the constant $a = \frac{1}{\int K_c(x)dx}$.

Define $\phi(x) = a\,K_c(x)$, so that

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

Define $\phi_\sigma(x) = \frac{1}{\sigma^k} \cdot \phi\,(x/\sigma)$.

Basic Lemma: Let $f_c \in \kappa$. Then

$$\|f_c - \phi_\sigma * f_c\|_1 \underset{\sigma \to 0}{\longrightarrow} 0$$

(Here $*$ denotes convolution)

Thus functions of the form f_c can be arbitrarily well approximated by $\phi_\sigma * f_c$;

therefore sufficient to show $\phi_\sigma *\ f_c$ can be approximated by functions in S_0 arbitrarily well.

Now write (for T below sufficiently large):

$$(\phi_\sigma * f_c\,)\,(\alpha) = \int_{[-T,T]^r} \phi_\sigma(\alpha - x) f_c\,(x) dx$$

$$\approx v_n(\alpha) \equiv \sum_{i=1}^{n^k} \phi_\sigma\,(\alpha - \alpha_i) f_c\,(\alpha_i) \left(\frac{2T}{n}\right)^k, \qquad (5)$$

where α_i are points of the form

$$[-T + \tfrac{2i_1T}{n},\ -T + \tfrac{2i_2T}{n}, \ldots,\ -T + \tfrac{2i_kT}{n}]$$

[one point in each sub-cube of size $2T/n$].

Riemann sum implies that $\nu_n \underset{n \to \infty}{\longrightarrow} \phi_\sigma * f_c$ pointwise; then can use dominated convergence theorem to show that convergence is also in L^1.

Thus we can approximate $\phi_\sigma * f_c$ by ν_n. Now need to show ν_n can be approximated by something in S_0. By (5)

$$\nu_n(\alpha) = \frac{1}{n^k} \sum_0^{n^k} \frac{f_c(\alpha_i)(2T)^k}{\int K_c(\alpha)\, d\alpha \cdot \sigma^k} K_c\left(\frac{\alpha - \alpha_i}{\sigma}\right).$$

Now replace K_c by K which can be made arbitrarily close; then have something in S_0.

Converse of the theorem (only if) is easy.

Second theorem for L^2 density:

Define

$$S_1 = \left\{ \sum_i a_i K((x - z_i)/\sigma_i) \,\middle|\, a_i, \sigma_i \in \mathbb{R};\ z_i \in \mathbb{R}^d \right\}$$

(variable scale σ_i which can depend on i added).

Theorem: Assuming that K is square integrable, then $S_1(K)$ is dense in $L^2(\mathbb{R}^d)$ iff K is non-zero a.e.

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