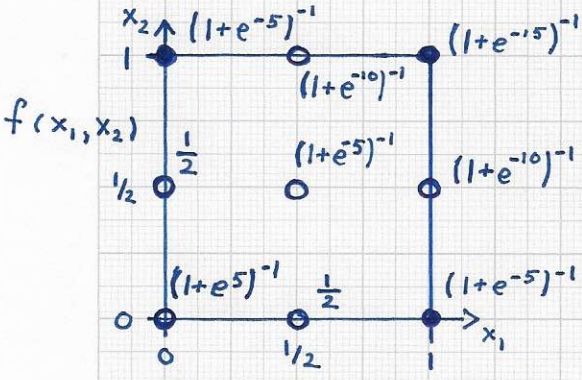
■ OBJECT AND MATERIAL EIGENFUNCTIONS - Cont'd.

• Laplacian eigenfunctions and neural networks:...



Nine values of the activation function

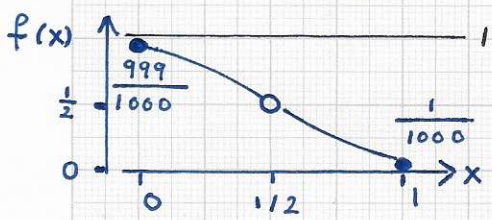
$$f(x_1,x_2) = 1/(1+e^{-(10x_1+10x_2-5)})$$

for (x_1,x_2) tuples in the unit domain square.

This function also has contour lines of the general form

$$x_1 + x_2 = \text{'const'}$$

and represents an implementation of the 'v' operator.



Design requirements for an activation functions that implements the '¬' operator:

$$f(x) = 1/(1+e^{-L(1-2x)}),$$

where $L = \ln(999)$.

iii) Boolean operator 'v':

For the 'v' operator one can design the necessary function $y = b + w_1x_1 + w_2x_2$ in an analagous fashion. We establish the linear equation system slightly differently for the 'v' operator:

$$f(1,1) = 1/(1+e^{-15}) \Rightarrow e^{-(b+w_1+w_2)} = e^{-15} \Rightarrow b + w_1 + w_2 = 15$$

$$f(1,0) = 1/(1+e^{-5}) \Rightarrow e^{-(b+w_1)} = e^{-5} \Rightarrow b + w_1 = 5$$

$$f(0,1) = 1/(1+e^{-5}) \Rightarrow e^{-(b+w_2)} = e^{-5} \Rightarrow b + w_2 = 5$$

The solution of this linear system is:

$$b = -5, \quad w_1 = w_2 = 10.$$

The resulting activation function is

$$f(x_1,x_2) = 1/(1+e^{-(-5+10x_1+10x_2)}).$$

One can generalize this function to

$$f(x_1,x_2) = 1/(1+e^{-L(-1+2x_1+2x_2)}).$$

iv) Boolean operator '¬':

For this operator one can design the univariate linear function by using these two conditions, for example:

$$f(0) = \frac{999}{1000} \Rightarrow 1+e^{-(b+w\cdot 0)} = \frac{1000}{999} \Rightarrow b = \ln(999)$$

$$f(1) = \frac{1}{1000} \Rightarrow 1+e^{-(b+w\cdot 1)} = \frac{1000}{1} \Rightarrow b+w = -\ln(999)$$

...

■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks: ...

• Specific activation function:

$$f(x_1,\dots,x_n) = 1/(1+e^{-l(x_1,\dots,x_n)}),$$

where

$$l(x_1,\dots,x_n) = b + \sum_{i=1}^{n} w_i x_i$$

⇒ contours of f:

$$f(x_1,\dots,x_n) = c$$

$$\Rightarrow 1+e^{-l(x_1,\dots,x_n)} = \frac{1}{c}$$

$$\Rightarrow e^{-l(x_1,\dots,x_n)} = \frac{1-c}{c}$$

$$\Rightarrow l(x_1,\dots,x_n) = \ln\left(\frac{c}{1-c}\right) = \text{'const'}$$

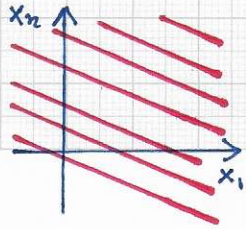
⇒ The contours of f are hyper-planes.

(E.g., $l(x_1,x_2) = \text{'const'}$ ⇒ lines ;

$l(x_1,x_2,x_3) = \text{'const'}$ ⇒ planes)

• Note. The value of $\ln\left(\frac{c}{1-c}\right)$ must be defined. Thus,

$$0 < c < 1 \Rightarrow 0 < \frac{c}{1-c} < \infty.$$



A hyper-plane contour of the function f can separate the $(x_1,\dots,x_n)$ domain only with a "hyper-plane separator."

The solution of this system is:

$$b = L \quad , \quad w = -2L \quad , \text{ where } L = \ln(999).$$

The univariate activation function is

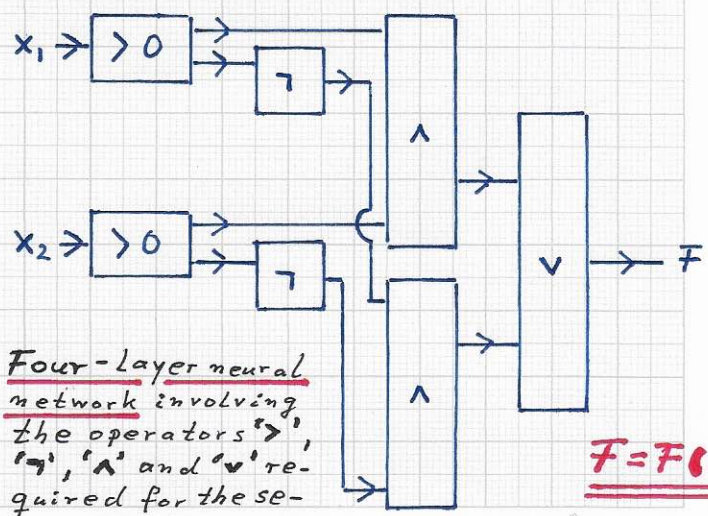
$$f(x) = 1/(1+e^{-L(1-2x)}) \quad .$$

It is sketched in the figure on the previous page (left, bottom).

The motivating example for the design of these neural network building blocks was the goal of devising a multi-layer network for the computation of the value of an expression involving relational and Boolean operators:

$$F(x_1,x_2) = \big(((x_1>0) \wedge (x_2>0)) \vee ((\neg(x_1>0)) \wedge (\neg(x_2>0))) \big) .$$

The network layers reflect the computations:



Four-layer neural network involving the operators '>', '¬', '∧' and '∨' required for the sequential, "layered" computation of $F(x_1,x_2)$.

$F = F(x_1,x_2)$

...

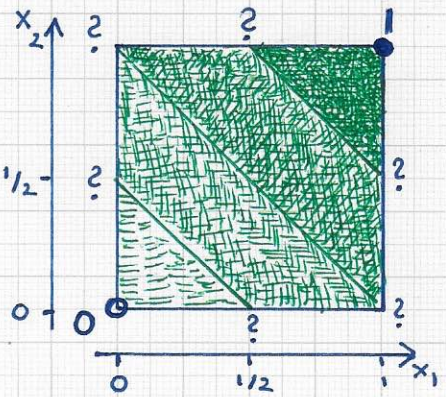
■ OBJECT AND MATERIAL EIGENFUNCTIONS - Cont'd.

•Laplacian eigenfunctions and neural networks: ...

• Boolean algebra — exact and fuzzy computation:

i) An implementation of an exact logical computation uses as input values that correctly represent the value 0 or 1; the computed output value also are precisely 0 or 1.

ii) An implementation of an approximate or fuzzy logical computation uses as input real values that represent value ranges for false and true; the computed output is also a real value representing false or true.



Fuzzy implementation of '∧' operator. The input range for x_1 and x_2 is the interval [0,1]. The fuzzy computation over the domain square generates values between 0 (false) and 1 (true). ?-values designed.

The example above — computing (a numerical approximation of) the value of the Boolean function $F(x_1, x_2)$ — demonstrates how the "nested layers of a Boolean expression" define corresponding layers of a multi-layer neural network. A Boolean variable can take on only the value 0 (false) or 1 (true); it is binary, discrete. The sigmoid function used to implement logical (and relational) operators depends on real numbers as input and generates a real number as output. Thus, the definition of false and true applies to a neural computing setting only in a "fuzzy" sense: a numerical value represents false (true) when it is close to zero (one).

In the most general context, one would want to "convert" a given logical circuit to a corresponding neural network — or vice versa — and demonstrate that they are equivalent, i.e., they generate the same output for the same input. ...

■ OBJECT AND MATERIAL EIGENFUNCTIONS - Cont'd.

• Laplacian eigenfunctions and neural networks: ...

• Relationship to material classification:

i) An unclassified material segment is characterized via H coefficient value histograms resulting from many local eigenfunction expansions of the segment, using H eigenfunction basis functions.

ii) Every classified material segment is characterized via the same information. The classified material segments' coefficient value histograms are "stored in a database" of classified samples.

iii) One can compute (up to) H scale-specific (eigenfunction-specific) real p-values, with $0 \le p \le 1$, serving as measures for the "match strength" or "degrees of similarity" between scales of the unclassified segment's expansion and a classified segment's expansion.

iv) One can view the (up to) H p-values as input to a Boolean logical circuit to perform classification.

The example described on the previous pagen - devising a multi-layer neural network for the evaluation of a Boolean (specific) expression - has demonstrated that the original "precise and binary-value-based computation" can be translated into a design of a "fuzzy, probabilistic, real-value-based multi-layer neural network." We can now establish the relationship to our overarching material classification problem involving coefficient histograms of eigenfunction expansions. We introduced the concept of a **decider function $F = F(p_1, \ldots, p_H)$** that uses (some of the available) p_i-values as input, with $0 \le p_i \le 1$, and returns as output a real number indicating the "match strength" between a new unclassified material sample and one, multiple or zero classified material samples "stored in the database." **A multi-layer neural network implementation of F must effectively perform/emulate the Boolean logic computations defining a "match."** ...

■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks: ...

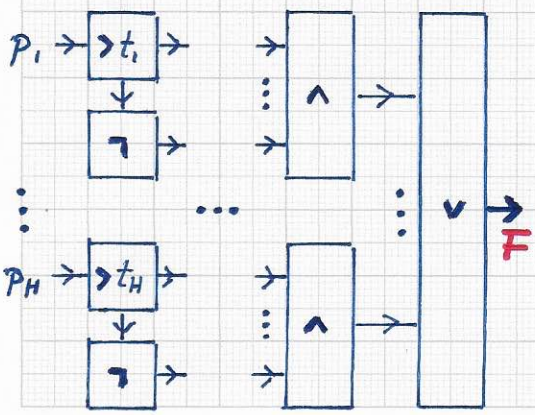
• Conceptual view of a purely logic-based approach to determining whether unclassified segment S belongs to class represented via a classified sample s stored in database:

Input: $p_1, \ldots, p_H$
$(0 \leq p_i \leq 1)$

Output: $F(p_1, \ldots, p_H)$

Algorithm:

```
/* If a complex con-*/
/* dition involving p1, */
/* ..., pH is 'satisfied', */
/* then F will return */
/* a 'high probabitity' */
/* for a match between */
/* S and s, as far as */
/* class membership */
/* is concerned; a */
/* 'low probability' */
/* will result other- */
/* wise. */
```



One can now view the classification problem as follows:

• The database of stored normalized histogram data for classified material samples consists of approximately 10^6 histograms when 100 classes, 100 samples per class (on average) and $125 = 5^3 = H$ eigenfunctions (5^3 being convolution mask resolution) are used.

• An unclassified image/material segment is represented by H histograms. This segment is called S.

• If the essential classification step was the computation of a "match strength" between S and the approximately $100 \cdot 100 = 10^4$ stored segments, classified samples, an expert-informed and-defined 'exhaustive logical classification method' could be done as follows:

– FOR all classified segments s DO
– compute $F(p_1, \ldots, p_H)$ for S and s;
– IF output of F indicates "match"
THEN S belongs to class of s;

Example. The H p-values are real input values; they are larger (or not) than t-values. F yields a real result.

...