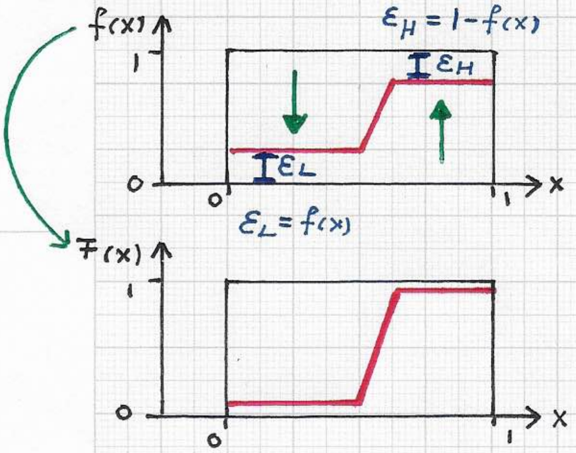
OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks: …

Transforming a function $f(x)$, $x \in [0,1]$, $f \in [0,1]$, with LOW f-values for $x \in [0, 1/2]$ and HIGH f-values for $x \in [1/2, 1]$ to a function $F(x)$ with even LOWER (HIGHER) function values than $f(x)$ in the respective regions:



The function $f(x)$ can be transformed to $F(x)$ via the use of weight functions that emphasize even LOWER (HIGHER) function value behavior in the left (right) part of the x-domain.

Given: $f(x)$

Wanted: $F(x)$

Algorithm: Use linear weight/blending functions to emphasize f's behavior

$$F(x) = (1-x)\,(f(x))^2 + x\,(1-(1-f(x))^2)$$

Similarly, when a HIGH value of a function $F_1(p_1)$ is combined with another HIGH value of a function $F_2(p_2)$, the mathematical operation used to generate the value of the overall decider function F should lead to an "even HIGHER value." In other words: "A HIGH match probability indicated by a HIGH value of $F_1(p_1)$ together with a HIGH match probability indicated by a HIGH value of $F_2(p_2)$ implies an even HIGHER match probability, overall, of F."

Before describing the general method one can employ to construct a multivariate decider function F based on such principles and goals, it is important to first consider a simple example to illustrate the approach. The figure (left) illustrates the objective. The figure shows the differences between $f(x)$ and 0 (ε_L) and 1 (ε_H). The provided formula for $F(x)$ is an example that uses ε_L^2 and $(1-f(x))^2 = \varepsilon_H^2$ for the transformation. …

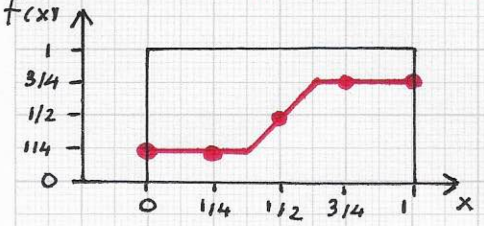
Stratovan

OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

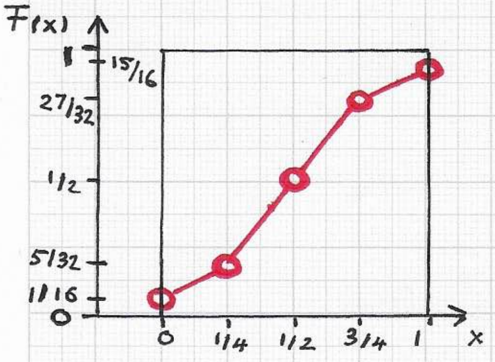
• Laplacian eigenfunctions and neural networks: ...

Numerical example for the transformation

$$F(x) = (1-x)\,(f(x))^2 + x\,(1-(1-f(x))^2)$$



x	f	F
0	1/4	1/16
1/4	1/4	5/32
1/2	1/2	1/2
3/4	3/4	27/32
1	3/4	15/16



Result of transforming f(x) to F(x). The "low left values are even lower", the "high right values are even higher." The value 1/2 remains unchanged.

The numerical example provided (left) demonstrates the effect of the specific transformation.

• **Note.** It is interesting to realize that the "shape of the SIGMOID function" could also serve as a good model to represent individual univariate decider functions $F_{SC}(p_{SC})$.

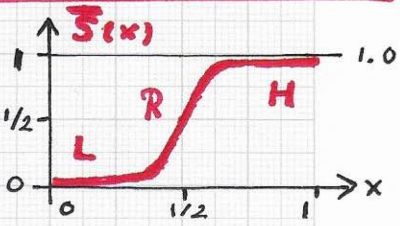
The SIGMOID function is defined as

$$S(x) = 1/(1+e^{-x}) = e^x/(1+e^x).$$

One can use "shape parameters" a and γ to generalize the standard SIGMOID function as follows:

$$\bar{S}(x) = 1/(1+e^{-\gamma(x-a)}) = e^{\gamma(x-a)}/(1+e^{\gamma(x-a)}).$$

The function $\bar{S}(x)$, sketched in the figure (right) is the specific function



$$\bar{S}(x) = 1/(1+e^{-20(x-0.5)}).$$

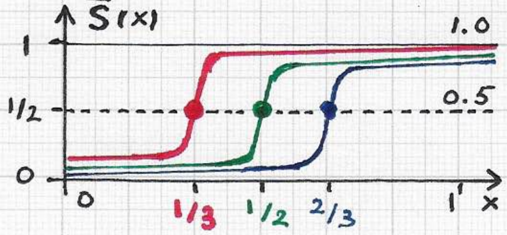
The graph of this function clearly exhibits the three types of behavior we look for in a univariate decider function, i.e., **L-, R- and H-regions.**

...

OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

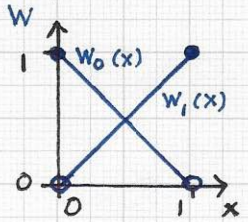
- Laplacian eigenfunctions and neural networks: ...

Generalized SIGMOID functions with large exponential coefficient γ and three different values for the translation/shift parameter a:



Graphs of generalized SIGMOID functions with a-values $1/3$, $1/2$ and $2/3$.

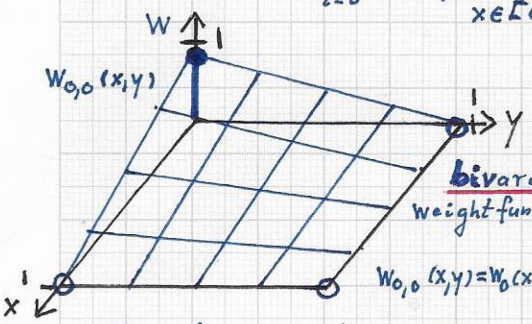
Defining weight functions for combining univariate decider functions via tensor products of univariate weight functions:



Univariate weight functions:

$W_0(x) = 1 - x$; $W_1(x) = x$

$\sum_{i=0}^{1} W_i = 1$; $W_i \geq 0$, $x \in [0,1]$



bivariate weight function: $W_{0,0}(x,y) = W_0(x)\,W_0(y)$

Four bivariate weight functions:

$W_{i,j}(x,y) = W_i(x)\,W_j(y)$, $i,j \in \{0,1\}$.

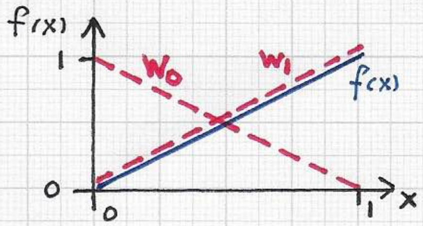
The figure (left) shows how the value of the parameter a of $\bar{S}(x)$ can be used to move the **"RAMP" R** to the left or right. The value of the parameter γ of $\bar{S}(x)$ can be used to model R: Low γ-values produce "wide RAMPs" with low slope values; high γ-values produce "narrow RAMPs" with high slope values. This Note merely serves one purpose: One can choose from several options to model the univariate decider functions, including the generalized SIGMOID function, high-degree Bernstein-Bézier polynomials or a simple piecewise linear function with three linear segments.

Whatever model one uses to represent univariate decider functions $F_{sc}(p_{sc})$, one must define the operation for constructing, for example, $F = F(F_1(p_1), F_2(p_2))$. The choice and use of proper weights/weight functions for blending F_1 and F_2 is crucial, see figures (left, bottom). ...

OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

- Laplacian eigenfunctions and neural networks: …

Guiding principle for combining function values: move values < 1/2 closer to 0; move values > 1/2 closer to 1. The principle is most simply exemplified via the univariate function $f(x)=x$ that is "blended with itself":

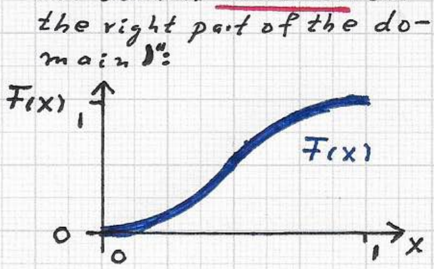


We use the blending functions $w_0(x)=1-x$ and $w_1(x)=x$ to blend two functions that are derived from $f(x)$ to implement the guiding principle:

$$f_0(x) = (f(x))^2 = x^2$$

$$f_1(x) = (1-(1-f(x))^2)$$

"Blend f_0 and f_1 with weight functions w_0 and w_1, thus 'emphasizing' **LOW** (in the left part of the domain) and **HIGH** (in the right part of the domain)":



$$F(x) = w_0(x) f_0(x) + w_1(x) f_1(x) = (1-x)x^2 + x(1-(1-x)^2) = -2x^3+3x^2$$

As it is crucial to keep in mind the overarching principle to be used when combining the values of multiple univariate decider functions, we state it here once again:

COMBINING TWO LOW VALUES (close to 0) SHOULD PRODUCE AN EVEN LOWER VALUE; COMBINING TWO HIGH VALUES (close to 1) SHOULD PRODUCE AN EVEN HIGHER VALUE.

(This principle effectively strengthens a potential mis-match (0) and a potential match (1) decision.)

The blending method illustrated in the figures (left) could be considered as an option for combining two functions $f(x)=x$ and $f(y)=y$ in a standard tensor product approach:

$f_0(x)=x^2$, $f_1(x)=(1-(1-x)^2)$; $f_0(y)=y^2$, $f_1(y)=(1-(1-y)^2)$.

$w_0(x)=(1-x)$, $w_1(x)=x$; $w_0(y)=(1-y)$, $w_1(y)=y$.

$\Rightarrow w_{0,0}(x,y)=w_0(x)w_0(y)$, $w_{1,0}(x,y)=w_1(x)w_0(y)$,

$w_{0,1}(x,y)=w_0(x)w_1(y)$, $w_{1,1}(x,y)=w_1(x)w_1(y)$.

…

■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks: ... ⇒

$$F(x,y) = w_{0,0}(x,y)\, f_0(x) f_0(y) + w_{1,0}(x,y)\, f_1(x) f_0(y) + w_{0,1}(x,y)\, f_0(x) f_1(y) + w_{1,1}(x,y)\, f_1(x) f_1(y)$$

$$= (1-x)(1-y)\, x^2 y^2 + x(1-y)\,(1-(1-x)^2)\, y^2 + (1-x)\, y\, x^2\, (1-(1-y)^2) + x\, y\, (1-(1-x)^2)(1-(1-y)^2)$$

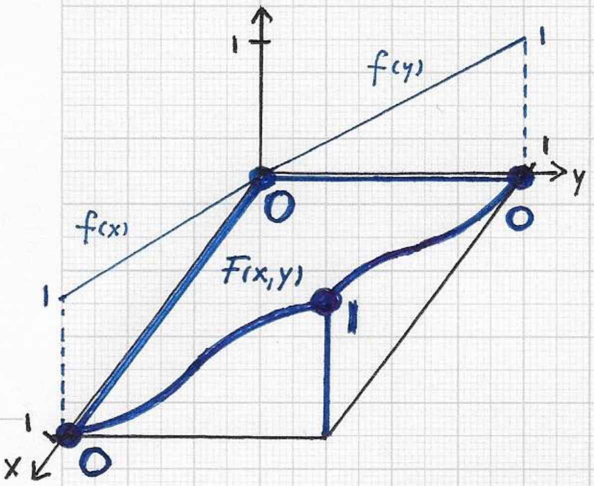
$$= (1-y) y^2 \{ (1-x) x^2 + x(1-(1-x)^2) \} + y(1-(1-y)^2) \{ (1-x) x^2 + x(1-(1-x)^2) \}$$

$$= (1-y) y^2 F(x) + y(1-(1-y)^2) F(x)$$

$$= F(x) \{ (1-y) y^2 + y(1-(1-y)^2) \}$$

$$= F(x)\, F(y)$$

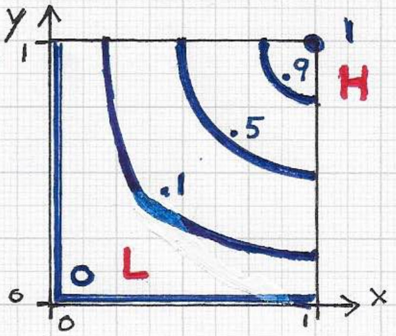
Tensor product method for combining $f(x)=x$ and $f(y)=y$:



Sketch of the graph of the function

$F(x,y) = F(x)\, F(y)$.

This function is one of the four standard bicubic Hermite polynomials.



Sketch of contours (iso-lines) of $F(x,y) = (-2x^3+2x^3) \cdot (-2y^3+3y^2)$. The contours indicate the regions of low, close-to-0 (L) and high, close-to-1 (H) values.

• TENSOR PRODUCT $F(x)\,F(y)$ = REALIZATION OF A "FUZZY AND":

$L \wedge x = L$

$H \wedge H = H$

Since $F(x)$ and $F(y)$ are the cubic Hermite polynomials that satisfy the conditions $F(0)=0$, $F(1)=1$, $F'(0)=F'(1)=0$, the tensor product $F(x,y)=F(x)\,F(y)$ is a bicubic Hermite polynomial. WHILE THIS BICUBIC POLYNOMIAL HAS SOME OF THE PROPERTIES EXPECTED FROM AN OVERALL DECIDER FUNCTION CANDIDATE, IT MIGHT NOT BE "IDEAL" SINCE: (i) the fact that $F(x,0)=F(0,y)=0$ and (ii) the "lack of a desirable symmetry behavior of $F(x,y)$" are shortcomings.

...