Stratovan

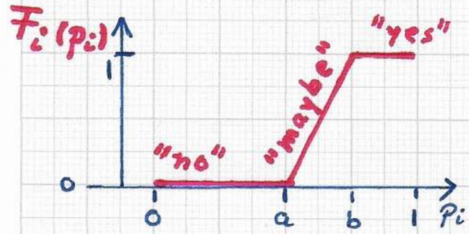
■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks: ...

• Key considerations:

(1) A p_i-value is solely based on a numerical measure of similarity. The measure is similarity of coefficient value histograms for scale i of segments **S** and **s**.

(2) An expert must define when two segments are (i) surely of the same class; (ii) surely not of the same class; and (iii) possibly of the same class – concerning scale i and based on the expert's knowledge how a numerical p_i-value reflects a true, physically correct class-match of two segments.



Example of a simple, allowable scale-specific decider function $F_i(p_i)$. An expert defines the values of a and b, and

$$F_i = \begin{cases} 0, & 0 \le p_i \le a \\ 1, & b \le p_i \le 1 \\ \frac{p_i - a}{b - a}, & a < p_i < b \end{cases}$$

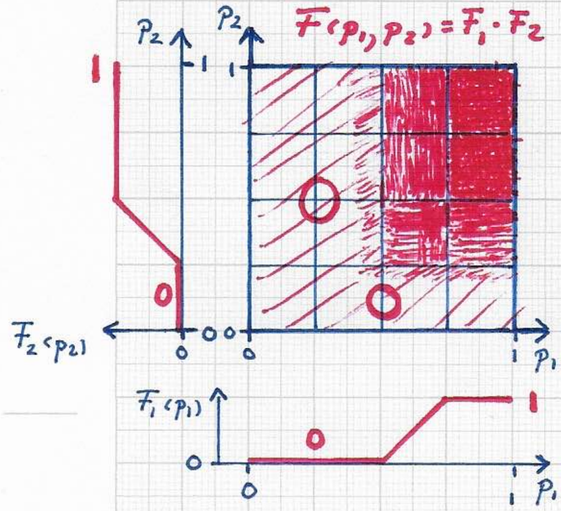
Since a process involving an expert for the definition of a decider function $F(\mathbf{p})$ must be simple and efficient, as far as a user's/expert's time spent on such a definition is concerned, this process must be well-designed. **An expert should have to specify only a MINIMAL AMOUNT OF DATA/INFORMATION with MAXIMAL IMPACT ON THE FINAL CONSTRUCTION of a NEAR-OPTIMAL DECIDER FUNCTION $F(\mathbf{p})$.** Thus, a preliminary design goal should be to establish scale-specific univariate functions $F_1(p_1), \ldots, F_H(p_H)$ that detime to what degree segments S and s are a match purely based on a single scale i. **One should also keep in mind that it might not be necessary or desired to involve all H p-values in the final decider function $F(\mathbf{p})$. For example, it might be sufficient to only consider one low-frequency, one medium-frequency and one high-frequency p-value for F.**

...

■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks:...

• Construction of $F(\mathbb{p})$ as a tensor product:



Example of a simple construction of a multivariate decider function $F(\mathbb{p})$ via the use of a tensor product.

Two individual scale-specific functions $F_1(p_1)$ and $F_2(p_2)$ are given. The tensor product of these functions is

$$F(\mathbb{p}) = F(p_1,p_2) = F_1(p_1)\cdot F_2(p_2), \quad (p_1,p_2)\in[0,1]^2.$$

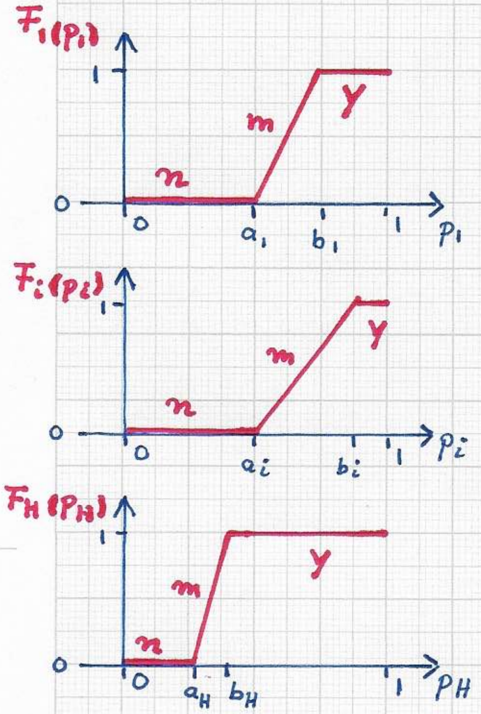
While the definition of this tensor product is simple and effective, one cannot expect that this product leads to the "near-optimal" function $F(\mathbb{p})$.

While it is reasonable to assume that an expert can easily and reliably decide when a p_i-value for segments S and s represents (does not represent) a class-match at scale i, it is substantially more complicated when considering multiple p-values, e.g., p_{i_1} and p_{i_2}. The "definition of class-match" itself becomes difficult, "subjective" and intuitively hard to comprehend when all H computed p-values (or any subset of more than one p-value) are to be used by a decider function $F(p_1,\ldots,p_H)$.

As a consequence one must "construct" a function $F(p_1,\ldots,p_H)$ very carefully due to its multivariate nature. First, even for an expert it is non-intuitive to specify conditions for $F(\mathbb{p})$ in its H-dimensional domain; second an effective and simple visualization of F for a high-dimensional domain is impossible; third, it would be too time-consuming to interactively define conditions in H-dimensional space. ...

■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks: ...



Examples of simple univariate decider functions. In this situation, an expert has specified only **2H** values, i.e., the end points of the H open intervals (a_i, b_i).

The univariate decider functions $F_i(p_i)$ merely consist of two constant and one "linear-ramp" pieces: **n – no match** (value = 0); **y – yes** (match, value = 1); **m – maybe a match** (value increasing linearly from 0 to 1).

• **Note.** Besides considering a human expert, one should also devise a more efficient automated method for defining a_i and b_i.

While the construction of "simple" univariate decider functions $F_i = F_i(p_i)$, $i = 1 \ldots H$, seems to be a rather straightforward task, the selection of a specific function "type" (e.g., tensor product) for creating the needed multivariate decider function $F(\mathbb{p}) = F(p_1, \ldots, p_H)$ is difficult. The "type" selection and subsequent construction process should be guided by optimization principles. The figures (left) show examples of extremely simple univariate decider functions $F_1(p_1), \ldots, F_i(p_i), \ldots, F_H(p_H)$, where an expert has determined the values of a_i and b_i, $i = 1, \ldots, H$. The expert has therefore established the intervals $[0, a_i]$, (a_i, b_i) and $[b_i, 1]$, $i = 1 \ldots H$, where the univariate decider functions return a **NO-CLASS-MATCH (n)**, **MAYBE-CLASS-MATCH (m)** and **YES-CLASS-MATCH (y) value**, respectively. In these examples, only two values must be specified – the values of a_i and b_i – which simplifies the definition and reduces the probability of introducing "expert errors." ...

■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks: ...

One can also interpret the individual decider functions $F_i(p_i)$ as "confidence functions," i.e., as functions producing values that inform us how "confident" one can be that a value of p_i indicates that there is no match **(NO – n)**, potentially a match **(MAYBE – m)** or definitely a match **(YES – y)** — regarding the class of two image segments **when** only considering one scale, i.e., scale **i**. Thus, the multivariate decider function $F(p_1, \ldots, p_H)$, to be constructed from the functions $F_i(p_i)$, for example, can be understood as an "integrative confidence function" that computes one final, overall confidence value from the individual scale-specific "confidence values" $F_i(p_i)$. **Considering the case of only two functions $F_1(p_1)$ and $F_2(p_2)$, the bivariate function $F(p_1, p_2)$ is constructed as a function $F(p_1, p_2) = Op(F_1(p_1), F_2(p_2))$, where 'Op' stands for some operator that combines the two univariate functions.** ...

• Possibilities for the construction of a bivariate decider function from two univariate decider functions:

Given: $F_1(p_1)$, $F_2(p_2)$; $p_1, p_2, F_1, F_2 \in [0,1]$

Wanted: $F(p_1, p_2) = F(F_1(p_1), F_2(p_2))$; $F \in [0,1]$

i) Multiplication (tensor product)

$$F(p_1, p_2) = F_1(p_1)\, F_2(p_2)$$

ii) Average

$$F(p_1, p_2) = (F_1(p_1) + F_2(p_2))/2$$

iii) Convex Combination

$$F(p_1, p_2) = \frac{w_1 F_1(p_1) + w_2 F_2(p_2)}{w_1 + w_2};$$

$w_1 + w_2 = 1$; $w_1, w_2 \geq 0$

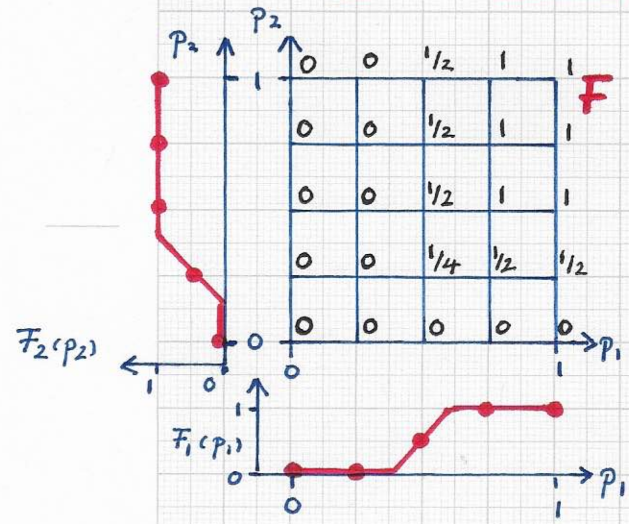
iv) "More general" combination

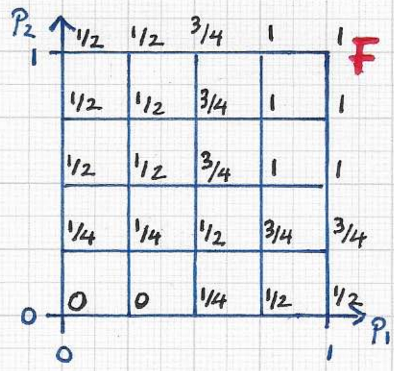
$$F(p_1, p_2) = Op\,(F_1(p_1), F_2(p_2))$$

■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks:...

• Kolmogorov's superposition theorem concerns the deconstruction of a given multivariate function into univariate functions. Our goal is the opposite, i.e., constructing a multivariate decider function:





Simple, straightforward constructions of the bivariate decider function (25 function values shown):

top: $F = F_1 \cdot F_2$

bottom: $F = (F_1 + F_2)/2$

• **Note.** A famous theorem by Kolmogorov (1956/1957) is related to our problem of constructing a multivariate decider function from simple univariate functions. In contrast to our "constructive" setting, Kolmogorov's theorem concerns "deconstruction"; it states:

There exist functions $f_{i,j}(x)$, $x \in [0,1]$, so that the multivariate function $f(x_1, \ldots, x_n)$, $x_i \in [0,1]$, can be written as

$$f(x_1, \ldots, x_n) = \sum_{j=1}^{2n+1} g_j \left(\sum_{i=1}^{n} f_{i,j}(x_i) \right).$$

!

In other words, a multivariate function can be deconstructed to an algebraic representation only involving univariate functions. This theorem has become of interest in the context of "analyzing, verifying and understanding" the behavior of certain multi-layer neural networks that are also multivariate as their input is a multi-dimensional feature vector/tuple $(x_1, \ldots, x_n)$.

...